Test-Driven Development in C

a Yahtzee game kata

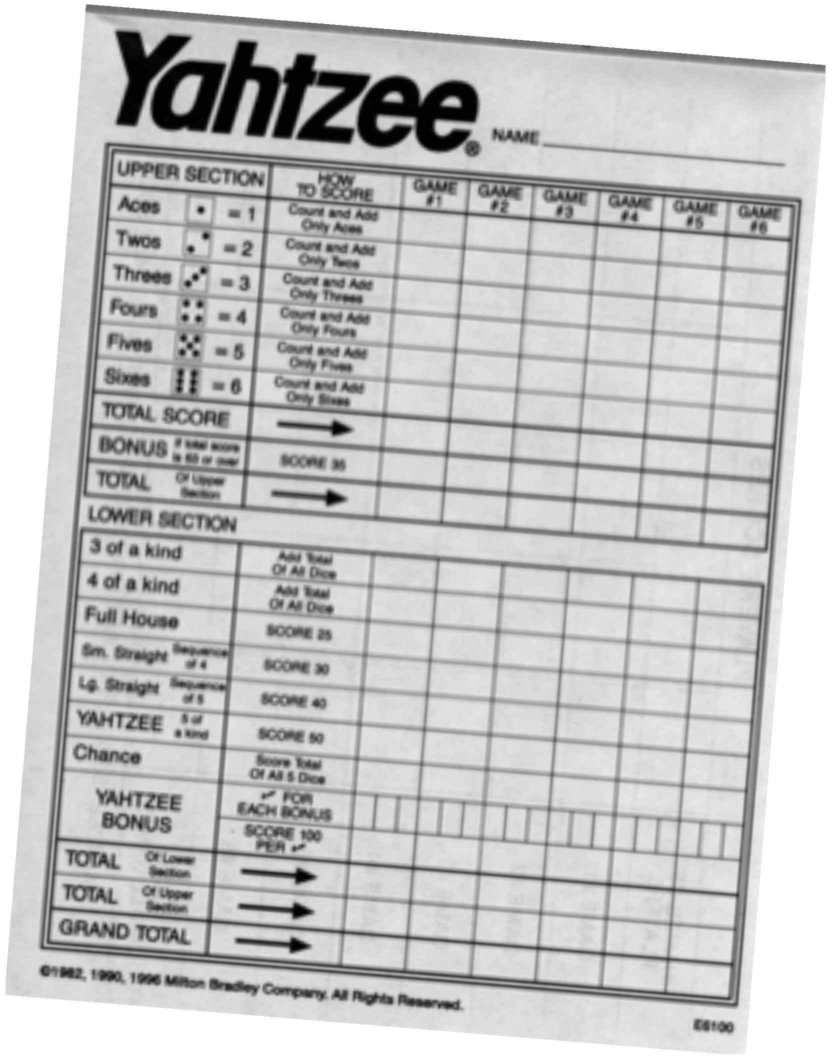
Olve Maudal, Cisco Systems Norway

During the last decade Test-Driven Development has become an established practice for developing software in the industry. In this lightning talk I will demonstrate Test-Driven Development in C.

A 15 minute lightning talk
dagen@IFI, October 27, 2011

The requirement

Write a C library that can score the lower section of a game of yahtzee.

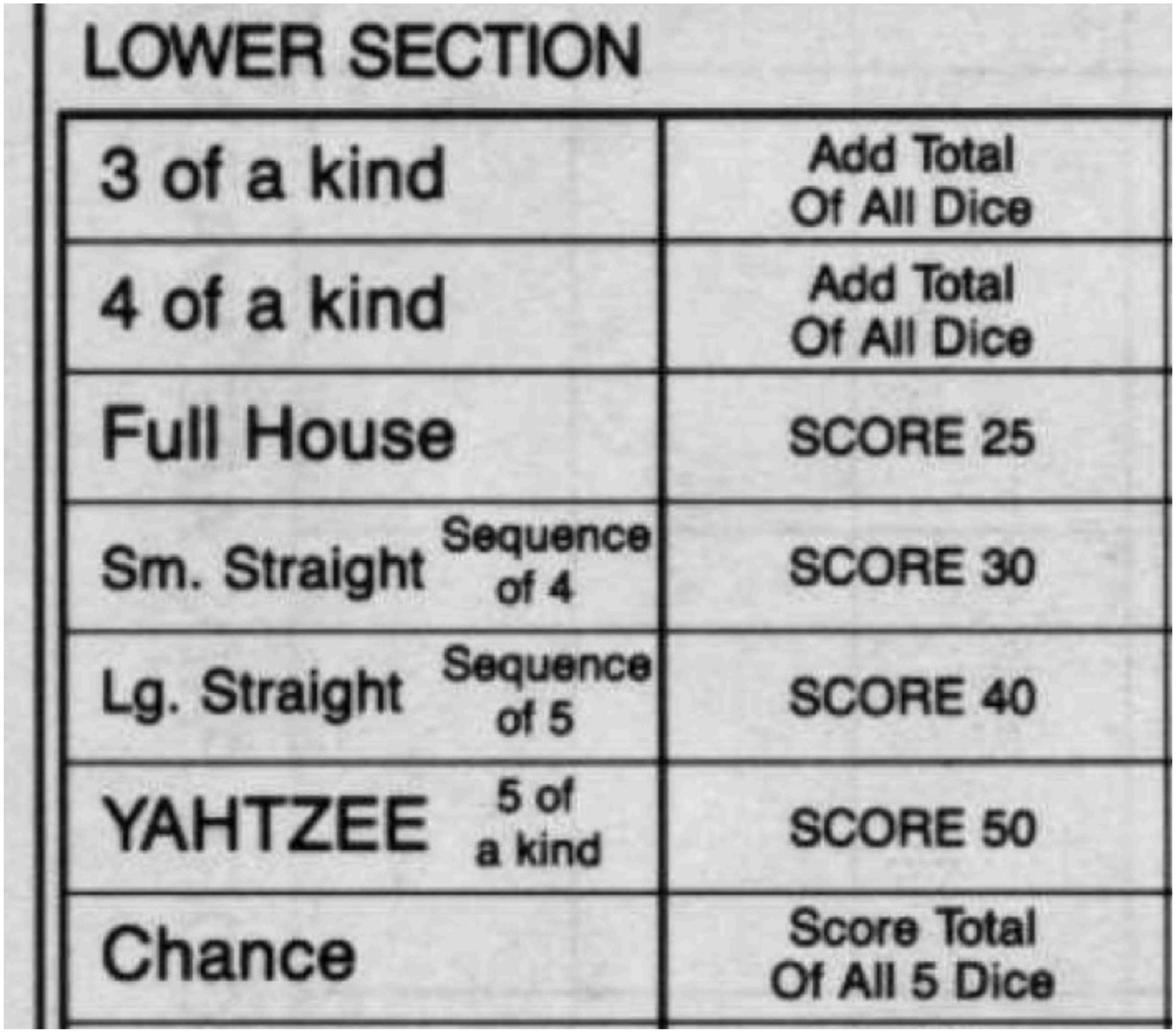


Yahtzee® NAME ____________

UPPER SECTION			HOW TO SCORE	GAME #1	GAME #2	GAME #3	GAME #4	GAME #5	GAME #6
Aces	⚀	= 1	Count and Add Only Aces						
Twos	⚁	= 2	Count and Add Only Twos						
Threes	⚂	= 3	Count and Add Only Threes						
Fours	⚃	= 4	Count and Add Only Fours						
Fives	⚄	= 5	Count and Add Only Fives						
Sixes	⚅	= 6	Count and Add Only Sixes						
TOTAL SCORE			→						
BONUS	If total score is 63 or over		SCORE 35						
TOTAL	Of Upper Section		→						
LOWER SECTION									
3 of a kind			Add Total Of All Dice						
4 of a kind			Add Total Of All Dice						
Full House			SCORE 25						
Sm. Straight	Sequence of 4		SCORE 30						
Lg. Straight	Sequence of 5		SCORE 40						
YAHTZEE	5 of a kind		SCORE 50						
Chance			Score Total Of All 5 Dice						
YAHTZEE BONUS			✓ FOR EACH BONUS						
			SCORE 100 PER ✓						
TOTAL	Of Lower Section		→						
TOTAL	Of Upper Section		→						
GRAND TOTAL			→						

©1982, 1990, 1996 Milton Bradley Company. All Rights Reserved.

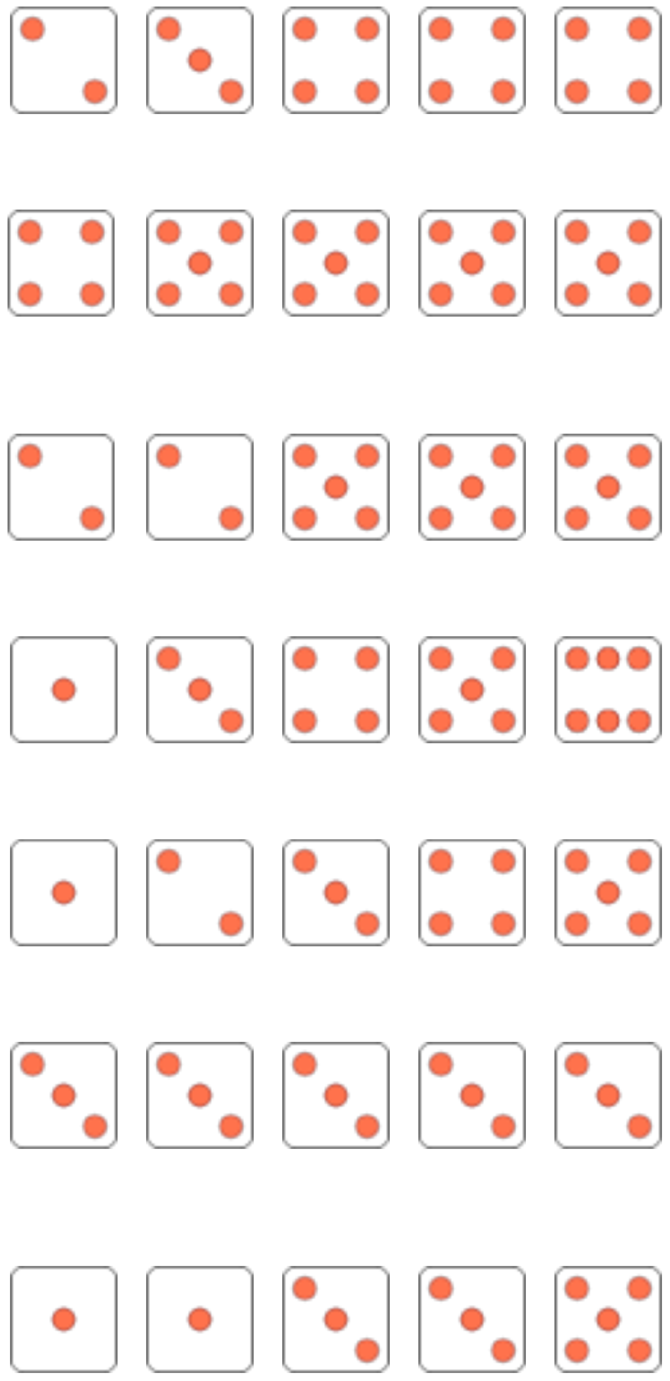
E6100



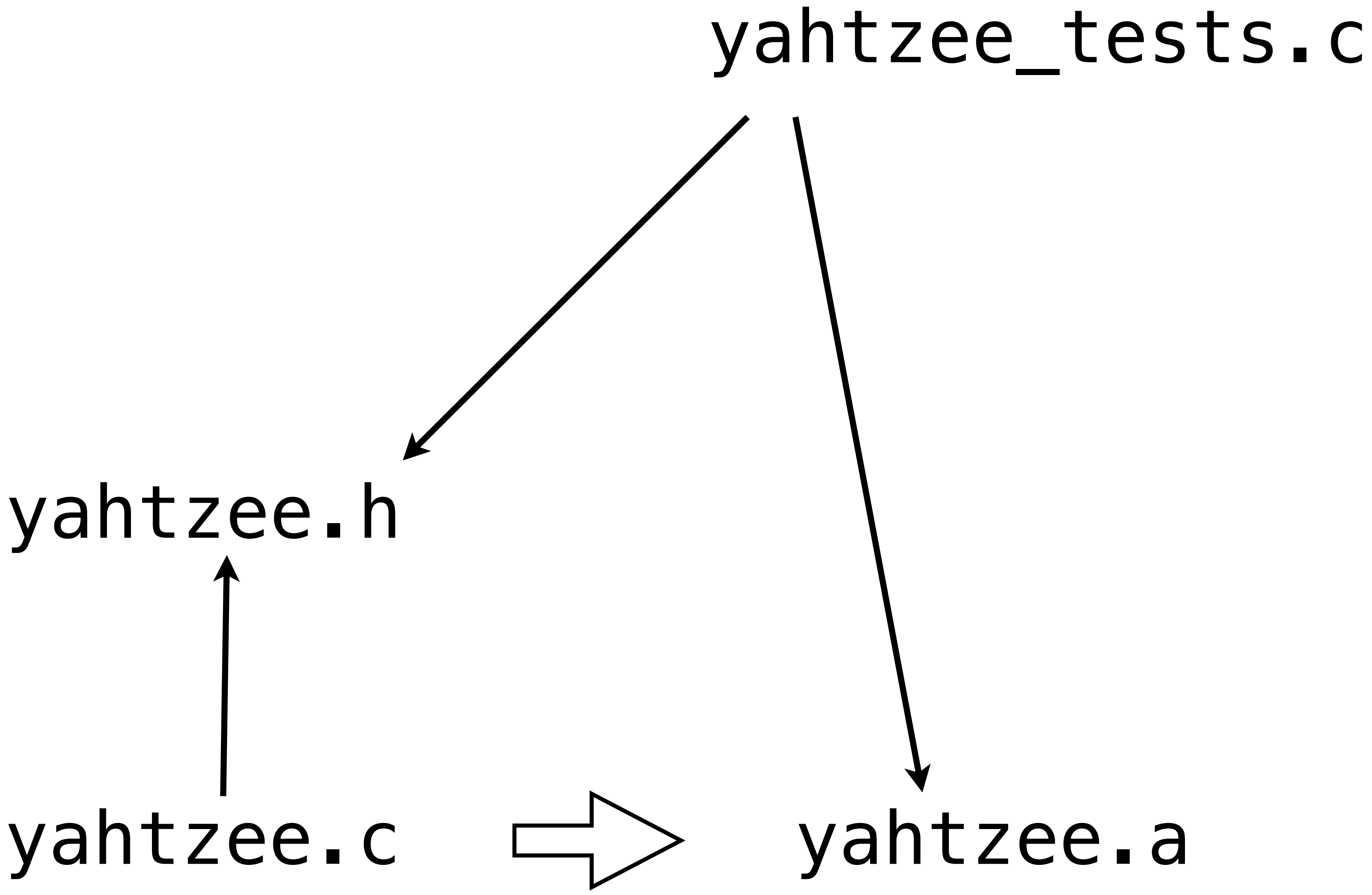
LOWER SECTION

3 of a kind	Add Total Of All Dice
4 of a kind	Add Total Of All Dice
Full House	SCORE 25
Sm. Straight Sequence of 4	SCORE 30
Lg. Straight Sequence of 5	SCORE 40
YAHTZEE 5 of a kind	SCORE 50
Chance	Score Total Of All 5 Dice

Examples



(see also http://en.wikipedia.org/wiki/Yahtzee)


yahtzee_tests.c
yahtzee.h
yahtzee.c
yahtzee.a

Makefile

```
CC=gcc
CFLAGS=-std=c99 -O -Wall -Wextra -pedantic
LD=gcc

all: yahtzee.a

yahtzee.a: yahtzee.o
	ar -rcs $@ $^

check: yahtzee_tests
	./yahtzee_tests

yahtzee.o: yahtzee.c yahtzee.h

yahtzee_tests.o: yahtzee_tests.c yahtzee.h

yahtzee_tests: yahtzee_tests.o yahtzee.a
	$(LD) -o $@ $^

clean:
	rm -f *.o *.a yahtzee_tests
```

yahtzee_test.c

yahtzee_test.c

```
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

```
$ make check
```

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

```
$ make check
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee_tests.o yahtzee_tests.c
```

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

```
$ make check
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee_tests.o yahtzee_tests.c
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee.o yahtzee.c
```

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

```
$ make check
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee_tests.o yahtzee_tests.c
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee.o yahtzee.c
ar -rcs yahtzee.a yahtzee.o
```

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

```
$ make check
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee_tests.o yahtzee_tests.c
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee.o yahtzee.c
ar -rcs yahtzee.a yahtzee.o
gcc -o yahtzee_tests yahtzee_tests.o yahtzee.a
```

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

```
$ make check
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee_tests.o yahtzee_tests.c
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee.o yahtzee.c
ar -rcs yahtzee.a yahtzee.o
gcc -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
```

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

```
$ make check
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee_tests.o yahtzee_tests.c
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee.o yahtzee.c
ar -rcs yahtzee.a yahtzee.o
gcc -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
Assertion failed: (6*9 == 42), function main, file yahtzee_tests.c, line 6.
```

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

```
$ make check
gcc -std=c99 -O -Wa                                         tzee_tests.c
gcc -std=c99 -O -Wa
ar -rcs yahtzee.a    htzee.o
gcc -o yahtzee_te  ts yahtzee_tests.o yahtzee.a
./yahtzee_tests
Assertion failed: (6*9 == 42), function main, file yahtzee_tests.c, line 6.
```

Our first unit test failed. This is good! Exactly what we want. The rythm of TDD is : fail, fix, pass... fail, fix, pass

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

```
$ make check
gcc -std=c99 -O -Wa                                             tzee_tests.c
        :99 -O -Wa
         ahtzee.a      htzee.o
gcc -o y htzee_te ts yahtzee_tests.o yahtzee.a
./yahtze _tests
Assertion failed: (6*9 == 42), function main, file yahtzee_tests.c, line 6.
```

Fail - Fix - Pass

Our first unit test failed. This is good! Exactly what we want. The rythm of TDD is : fail, fix, pass... fail, fix, pass

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*9 == 42);

    puts("Yahtzee tests OK");
}
```

```
$ make check
gcc -std=c99 -O -Wa                                         tzee_tests.c
        :99 -O -Wa
         ahtzee.a       tzee.o
gcc -o y htzee_te ts yahtzee_tests.o yahtzee.a
./yahtze _tests
Assertion failed: (6*9 == 42), function main, file yahtzee_tests.c, line 6.
```

Our first unit test failed. This is good! Exactly what we want. The rythm of TDD is : fail, fix, pass... fail, fix, pass

Fail - Fix - Pass

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*7 == 42);

    puts("Yahtzee tests OK");
}
```

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*7 == 42);

    puts("Yahtzee tests OK");
}
```

Fail - **Fix** - Pass

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*7 == 42);

    puts("Yahtzee tests OK");
}
```

Fail - **Fix** - Pass

```
$ make check
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee_tests.o yahtzee_tests.c
gcc -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
Yahtzee tests OK
```

yahtzee_test.c

```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*7 == 42);

    puts("Yahtzee tests OK");
}
```

Fail - **Fix** - Pass

Fail - Fix - **Pass**

```
$ make check
gcc -std=c99 -O -Wall -Wextra -pedantic   -c -o yahtzee_tests.o yahtzee_tests.c
gcc -o yahtzee_tests yahtzee_tests.o yahtzee.a
./yahtzee_tests
Yahtzee tests OK
```

yahtzee_test.c

```
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*7 == 42);

    puts("Yahtzee tests OK");
}
```

Fail - **Fix** - Pass

Fail - Fix - **Pass**

```
$ make check
gcc -std=c99 -O -                    -o yahtzee_tests.o yahtzee_tests.c
gcc -o yahtzee_te                    .a
./yahtzee_tests
Yahtzee tests OK
```

All tests are OK!

Let’s write a proper unit test

yahtzee_test.c

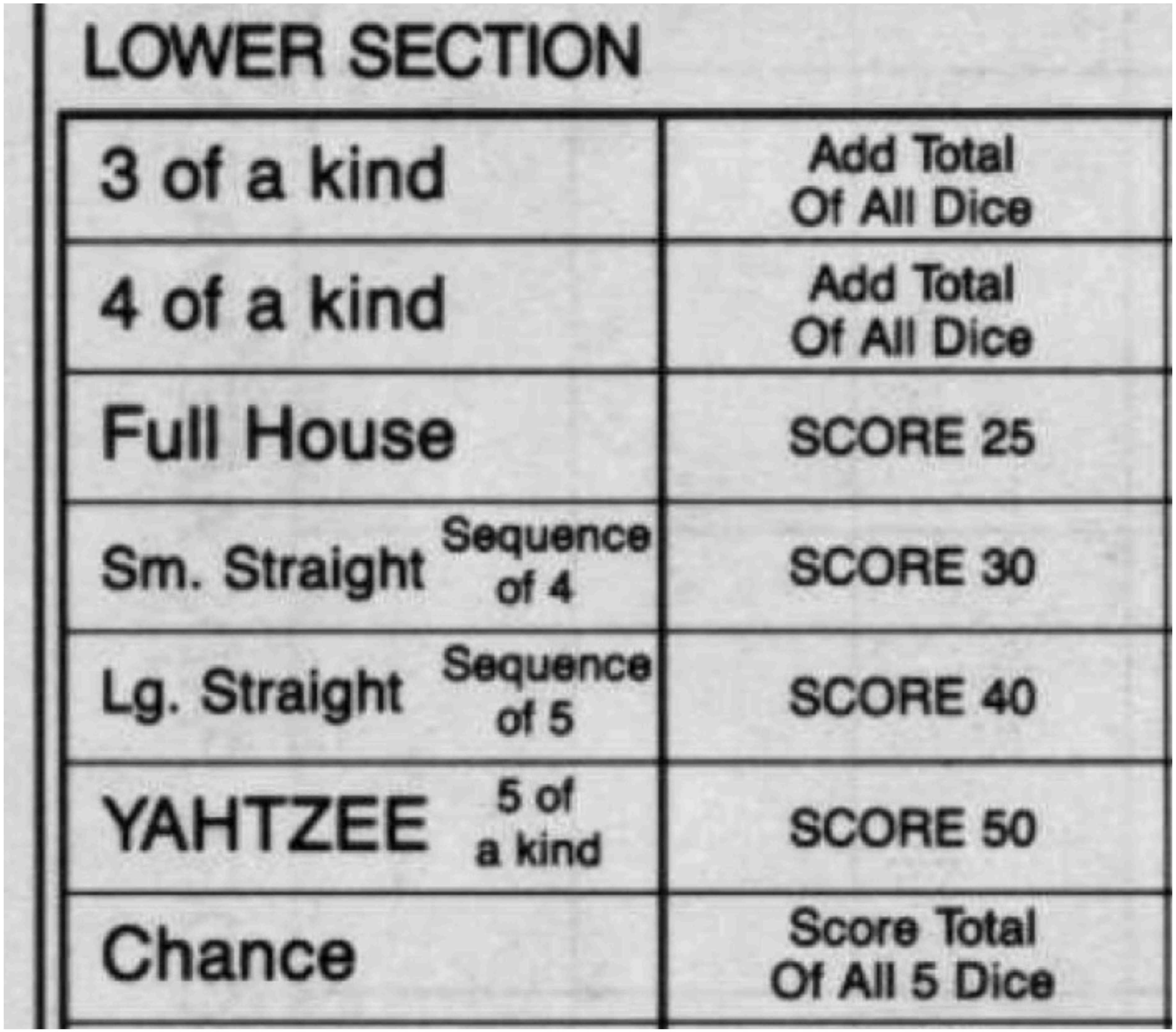
```c
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(6*7 == 42);

    puts("Yahtzee tests OK");
}
```

Fail - **Fix** - Pass

Fail - Fix - **Pass**

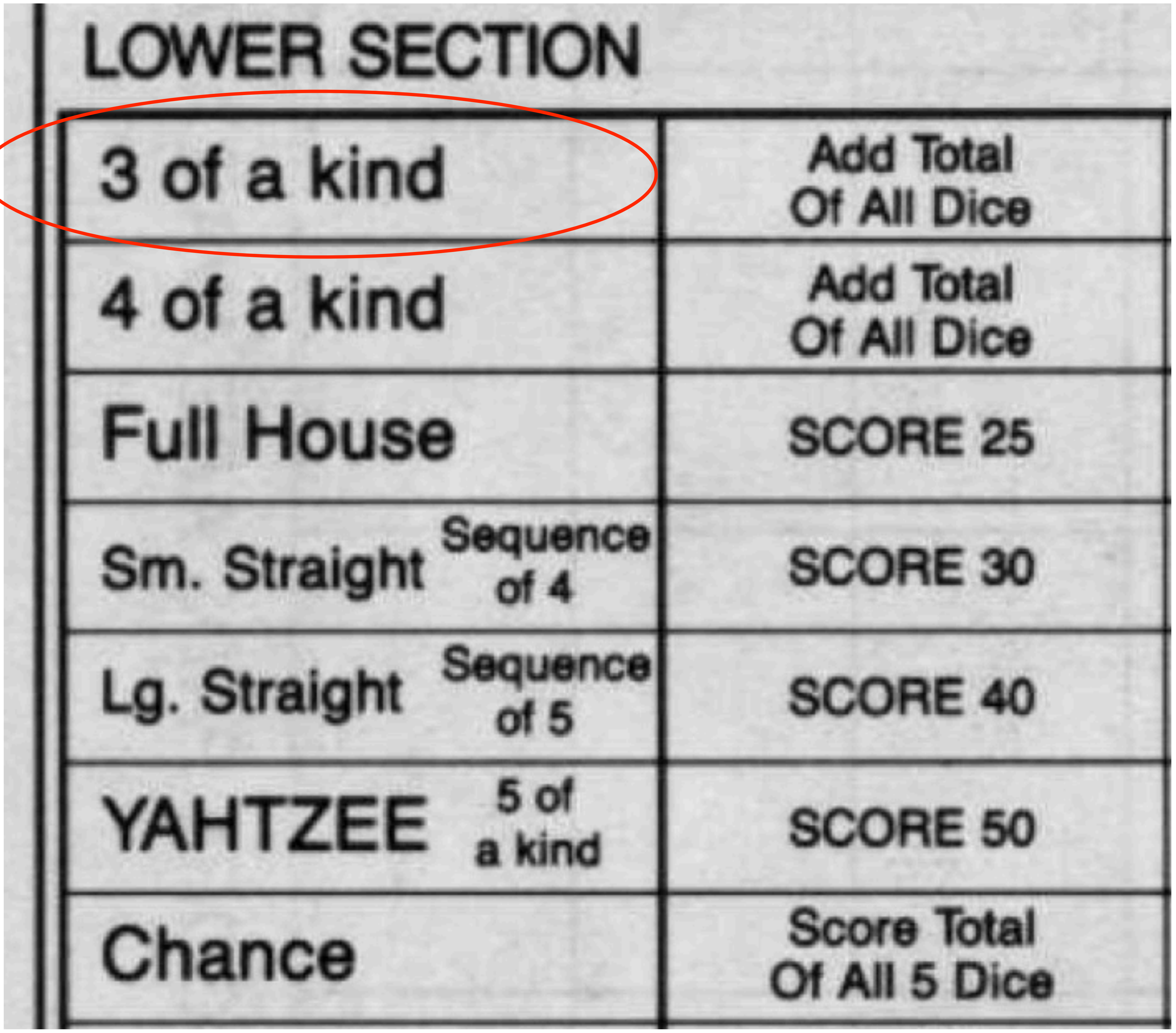
```
$ make check
gcc -std=c99 -O -                          -o yahtzee_tests.o yahtzee_tests.c
gcc -o yahtzee_te                          .a
./yahtzee_tests
Yahtzee tests OK
```

All tests are OK!



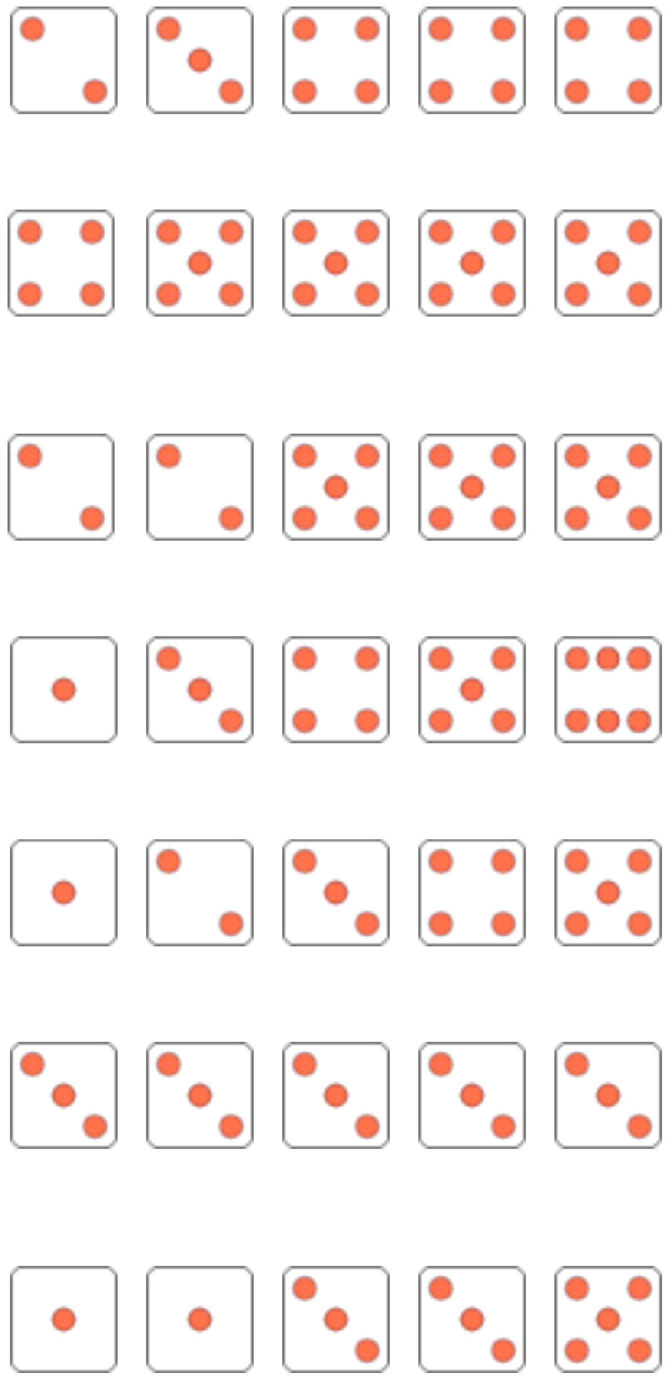
LOWER SECTION

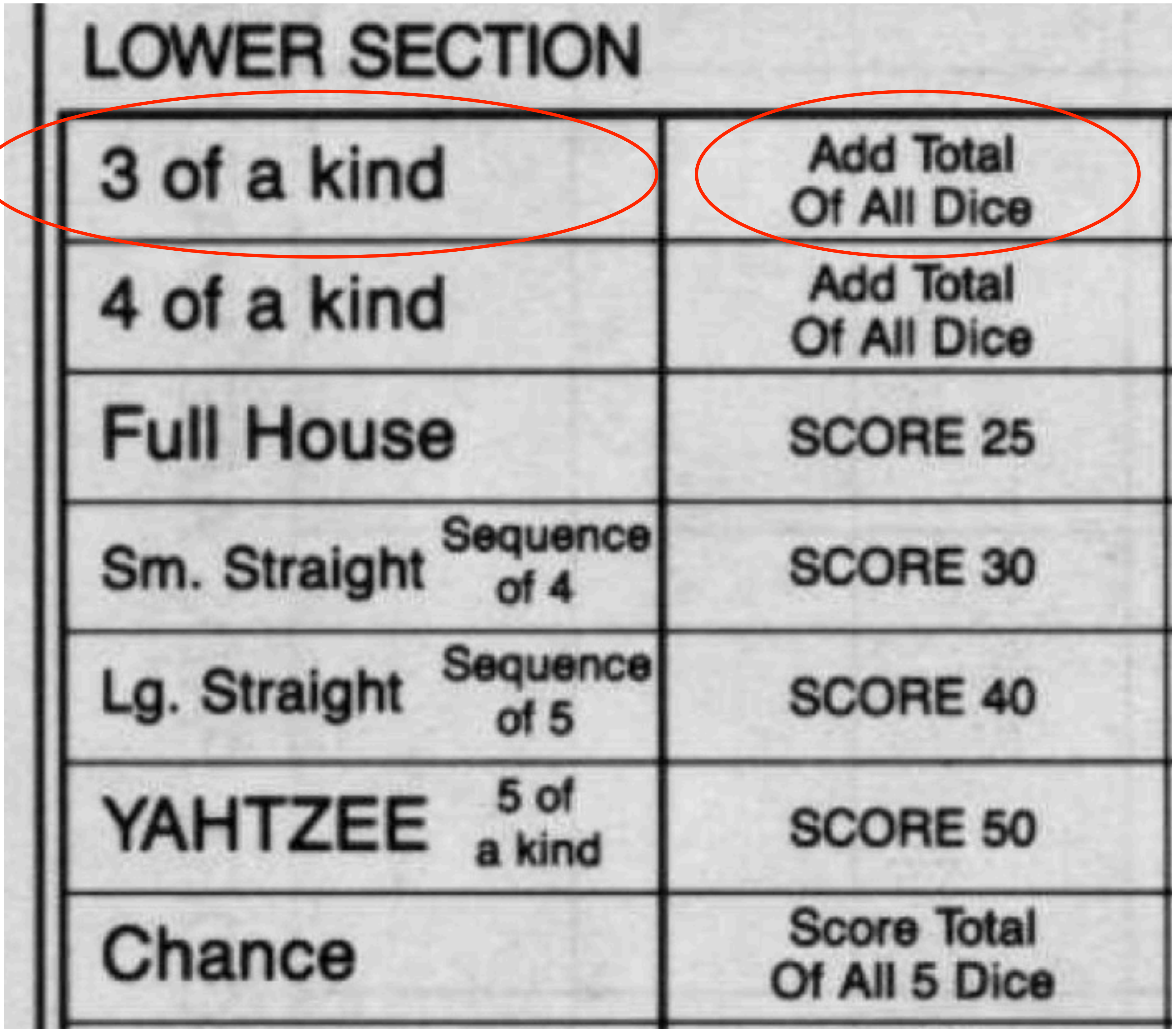
3 of a kind	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



LOWER SECTION

3 of a kind	Add Total Of All Dice
4 of a kind	Add Total Of All Dice
Full House	SCORE 25
Sm. Straight Sequence of 4	SCORE 30
Lg. Straight Sequence of 5	SCORE 40
YAHTZEE 5 of a kind	SCORE 50
Chance	Score Total Of All 5 Dice





LOWER SECTION

3 of a kind	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄

yahtzee_test.c

```
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{


    puts("Yahtzee tests OK");
}
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{



    puts("Yahtzee tests OK");
}
```

yahtzee_test.c

```
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return ??
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

what should we return?

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return ??
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

```
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return ??
}
```

what should we return?

Remember fail-fix-pass? Return something that you know will fail.

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

```
Assertion failed: (score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2)
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - Pass

```
Assertion failed: (score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2)
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - Pass

```
Assertion failed: (score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2)
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

```
Yahtzee tests OK
```

yahtzee.c

```
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

Fail - **Fix** - Pass

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

Congratulation. We have completed our first proper fail-fix-pass cycle.

Returning 7 is a minimal change to make it pass. This is OK because what we are concerned about now is just to make sure that the "wiring" of the test is OK. Is the test really being called and is it testing the right function?

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

Congratulation. We have completed our first proper fail-fix-pass cycle.

Returning 7 is a minimal change to make it pass. This is OK because what we are concerned about now is just to make sure that the "wiring" of the test is OK. Is the test really being called and is it testing the right function?

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

Let's add another unit test.

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

Congratulation. We have completed our first proper fail-fix-pass cycle.

Returning 7 is a minimal change to make it pass. This is OK because what we are concerned about now is just to make sure that the "wiring" of the test is OK. Is the test really being called and is it testing the right function?

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);

    puts("Yahtzee tests OK");
}
```

Let's add another unit test.

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);


    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);

    puts("Yahtzee tests OK");
}
```

```
Assertion failed: (score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4)
```

yahtzee.c

```
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - Pass

```
Assertion failed: (score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4)
```

yahtzee.c

```
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

Should we "cheat" again and check for the last dice, if 4 then return 10 otherwise 7?

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - Pass

```
Assertion failed: (score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4)
```

yahtzee.c

```
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

Should we "cheat" again and check for the last dice, if 4 then return 10 otherwise 7?

No! Another principle of TDD is that while you are supposed to do simple and "naive" increments you are not allowed to do "obviously stupid" stuff.

yahtzee.h

```
int score_t
```

yahtzee_test.c

```
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - Pass

```
Assertion failed: (score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4)
```

yahtzee.c

```
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

Should we "cheat" again and check for the last dice, if 4 then return 10 otherwise 7?

No! Another principle of TDD is that while you are supposed to do simple and "naive" increments you are not allowed to do "obviously stupid" stuff.

yahtzee.h

```
int score_t
```

A simple and naive thing to do here is to just **sum the dice** and return the value. That would satisfy all the tests and we know the functionality will eventually be needed.

```
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - Pass

```
Assertion failed: (score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4)
```

yahtzee.c

```
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

yahtzee.c

```
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return 7;
}
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```
#include "yahtzee.h"









int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (int die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```
#include "yahtzee.h"

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (int die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

Fail - **Fix** - Pass

yahtzee.c

```
#include "yahtzee.h"

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (int die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

Fail - **Fix** - Pass

```
Yahtzee tests OK
```

yahtzee.c

```
#include "yahtzee.h"

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (int die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

Fail - **Fix** - Pass

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```
#include "yahtzee.h"

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (int die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

But wait a minute. When indexing an array in C you should really use **size_t**. Let's fix it now as all tests are OK

Fail - **Fix** - Pass

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```c
#include "yahtzee.h"

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (int die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (int die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```
#include "yahtzee.h"


static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (int die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (int die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (int die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

```
Yahtzee tests OK
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

Nice. Let's start a new fail-fix-pass cycle

```
Yahtzee tests OK
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

Nice. Let's start a new fail-fix-pass cycle

```
Yahtzee tests OK
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);


    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);


    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);

    puts("Yahtzee tests OK");
}
```

```
Assertion failed: (score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0)
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - Pass

```
Assertion failed: (score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0)
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

So how do we fix this?

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - Pass

```
Assertion failed: (score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0)
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

So how do we fix this?

We need to check that we really have three of a kind.

yahtzee_test.c

```c
#include "yahtzee.h"
#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);

    puts("Yahtzee tests OK");
}
```

Fail - Fix - Pass

```
Assertion failed: (score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0)
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
        return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
        return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);

}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);

}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}








int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}









static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}










static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
#include "yahtzee.h"
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>



static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>
#include <stdbool.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>
#include <stdbool.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

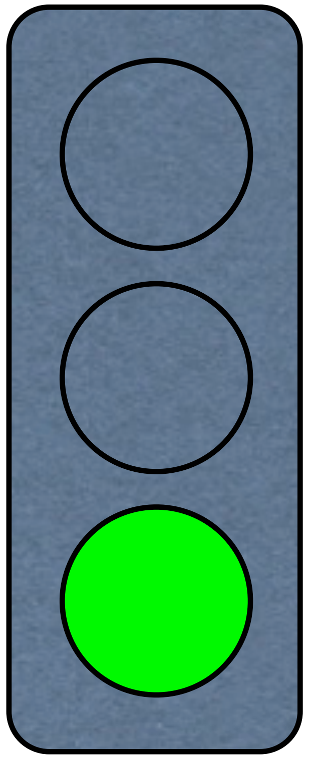
yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>
#include <stdbool.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

```
Yahtzee tests OK
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stddef.h>
#include <stdbool.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```



```
Yahtzee tests OK
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
```

```
Yahtzee tests OK
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
```

```
Yahtzee tests OK
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
```

```
Yahtzee tests OK
```

Looking good! Looking good!

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
```

```
Yahtzee tests OK
```

Looking good! Looking good!

yahtzee_test.c

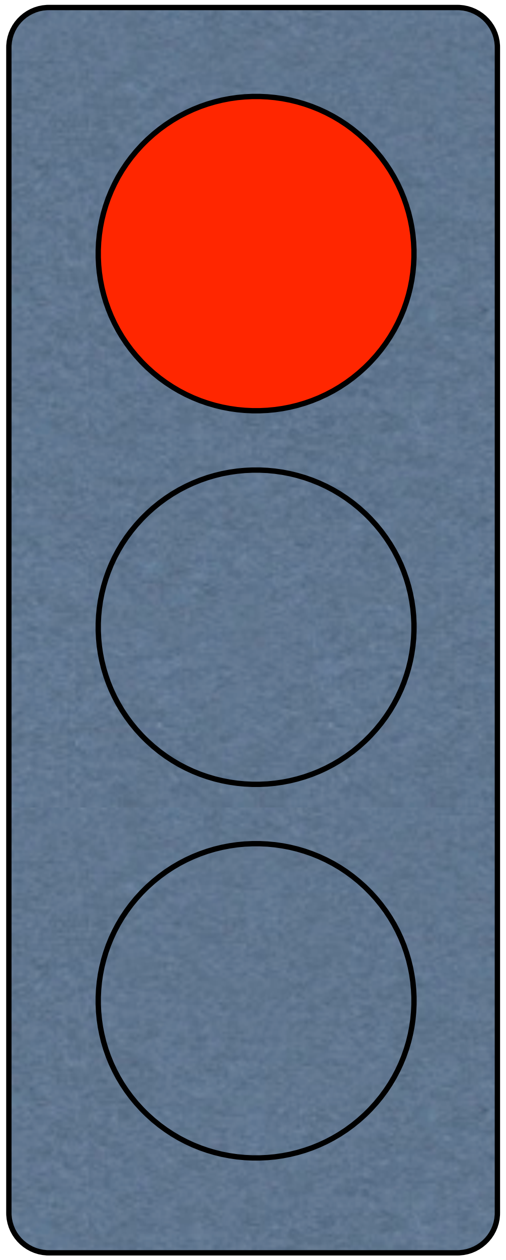
```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
```

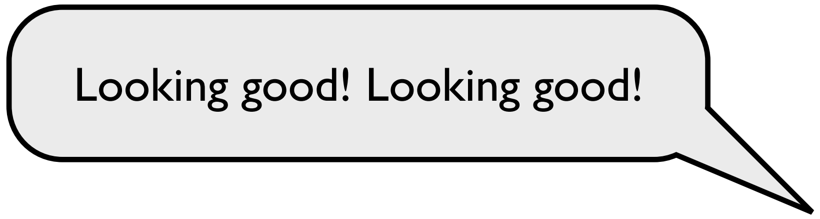
```
Yahtzee tests OK
```

Looking good! Looking good!

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);
```

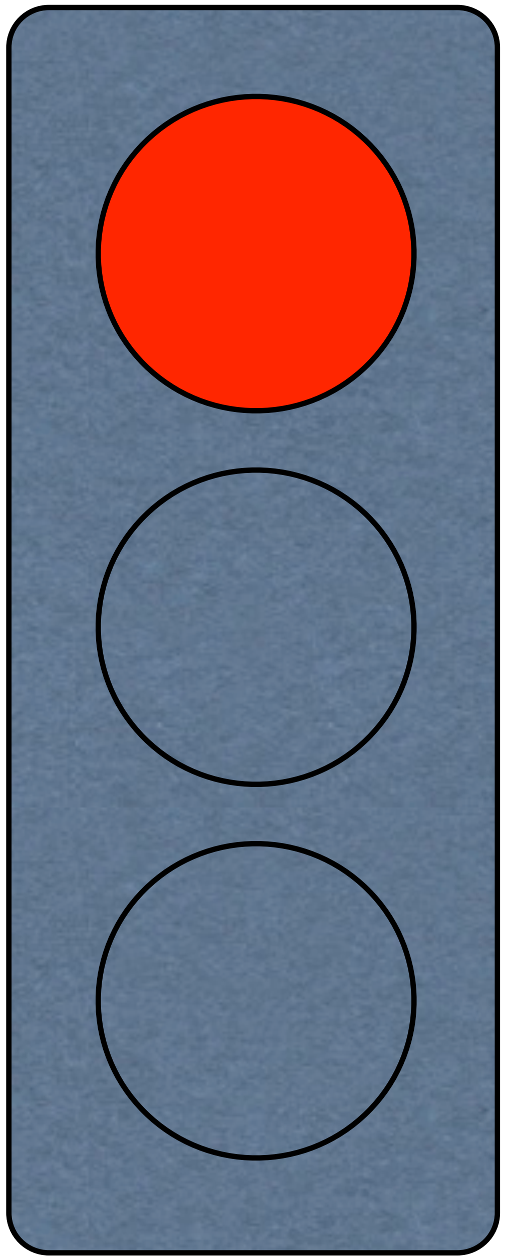




yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);
```

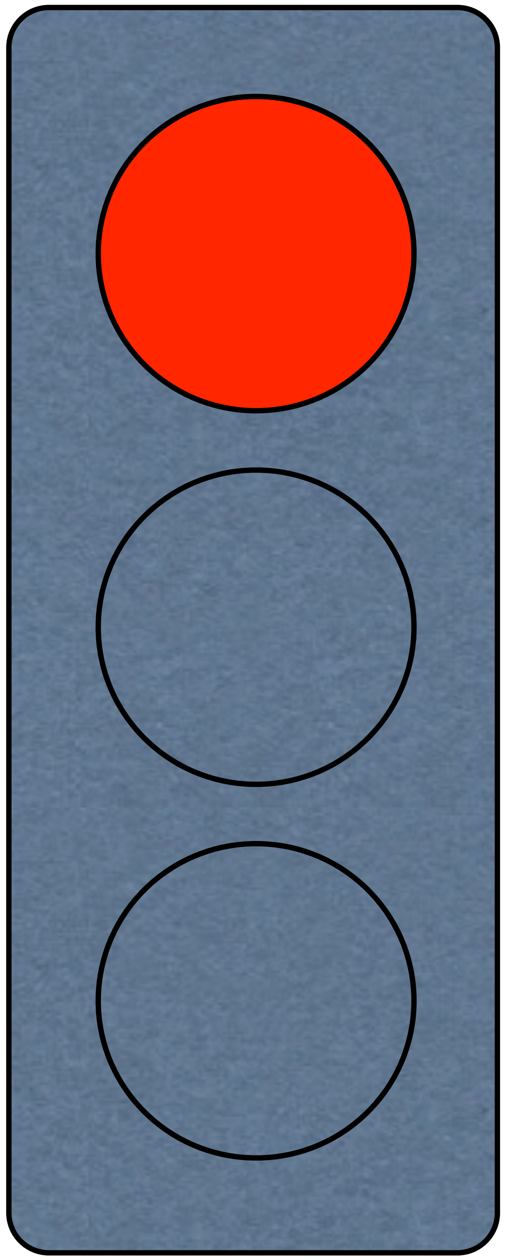
```
Assertion failed: (score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7)
```



yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);
```

```
Assertion failed: (score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7)
```



yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);
```

Oh no... what happened?

```
Assertion failed: (score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7)
```

Oh no... what happened?

```
Assertion failed: (score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7)
```

yahtzee.c

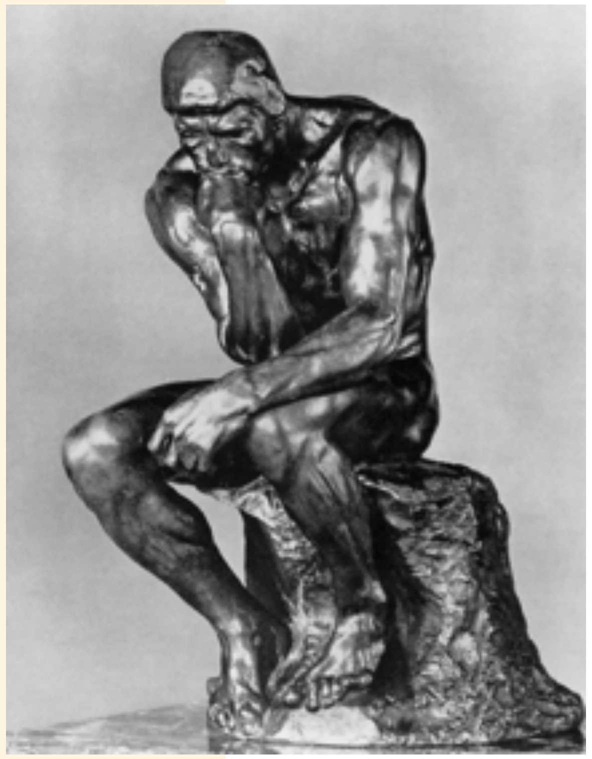
```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

Oh no... what happened?

```
Assertion failed: (score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7)
```

yahtzee.c

```c
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```



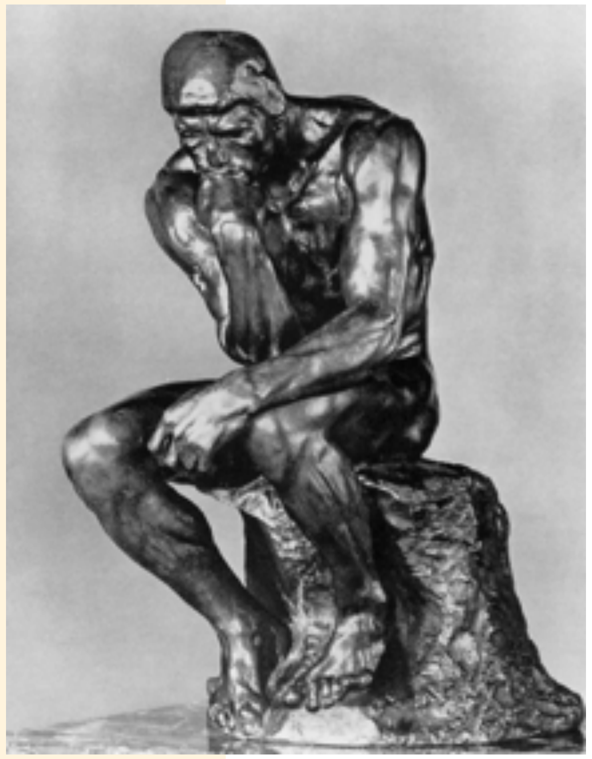
Oh no... what happened?

```
Assertion failed: (score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7)
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

But of course...



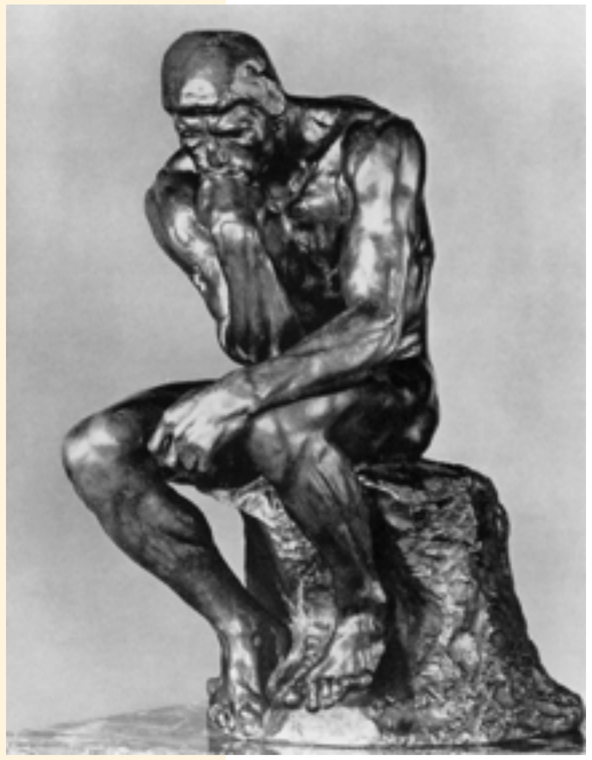
Oh no... what happened?

```
Assertion failed: (score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7)
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

But of course...



Oh no... what happened?

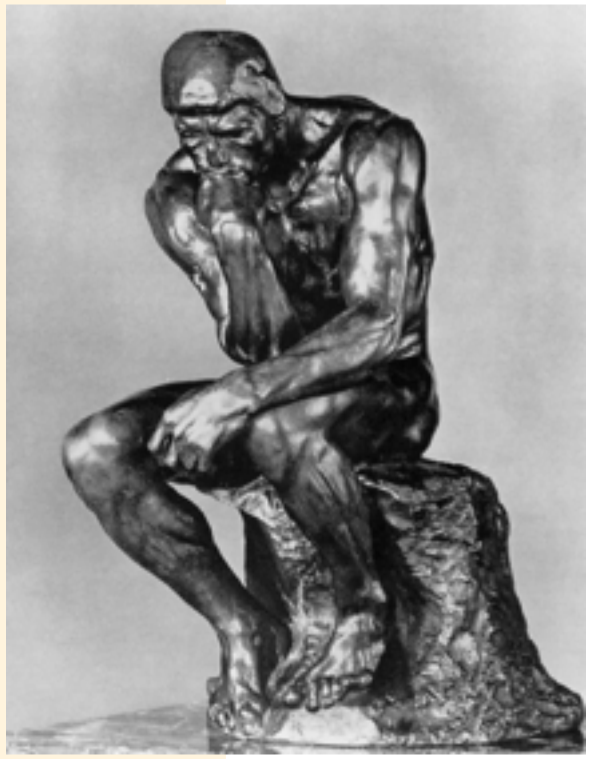
```
Assertion failed: (score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7)
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

But of course...

it should be ***at least*** three...



Oh no... what happened?

```
Assertion failed: (score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7)
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

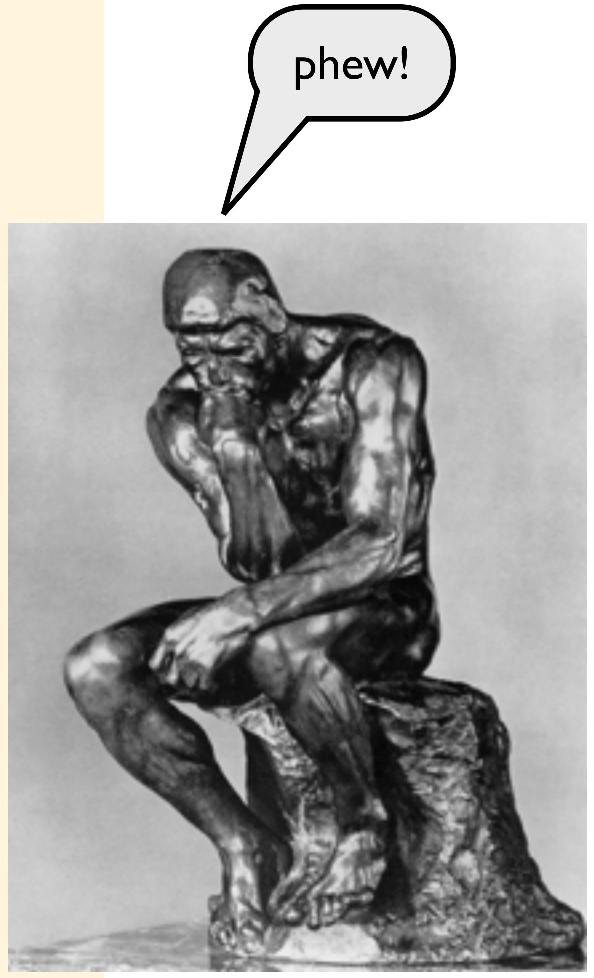
```c
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

```
Yahtzee tests OK
```

yahtzee.c

```c
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

```
Yahtzee tests OK
```



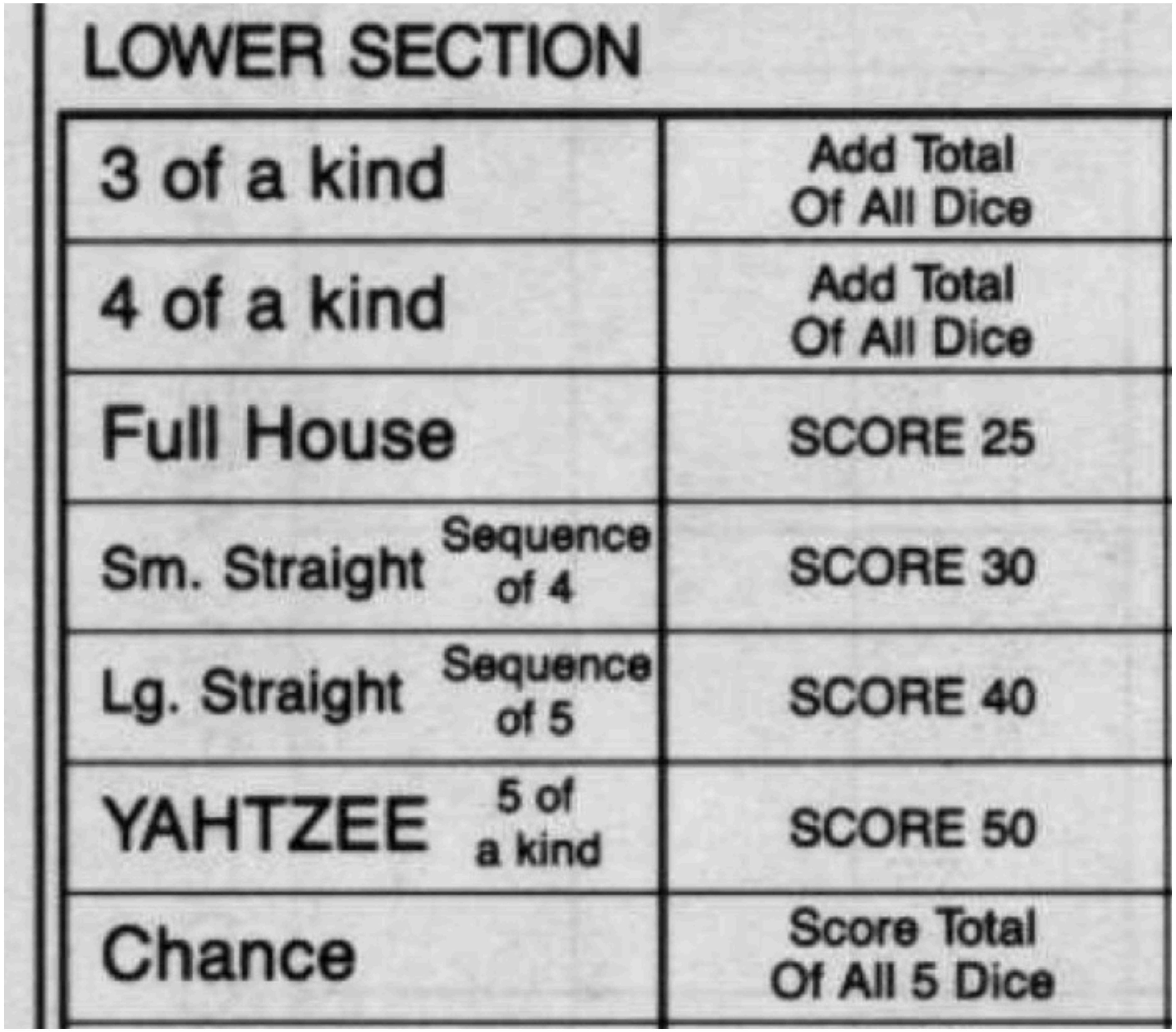
yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);
```

Now we have a decent implementation of three of a kind

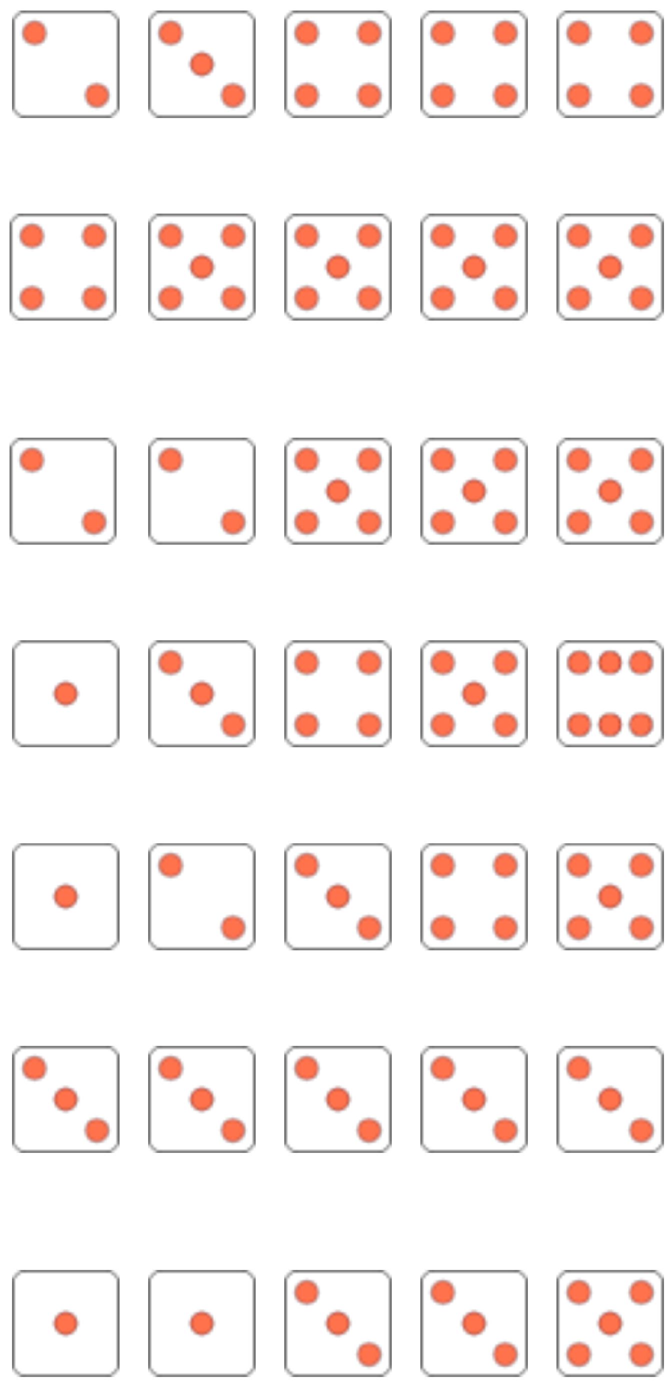
yahtzee_test.c

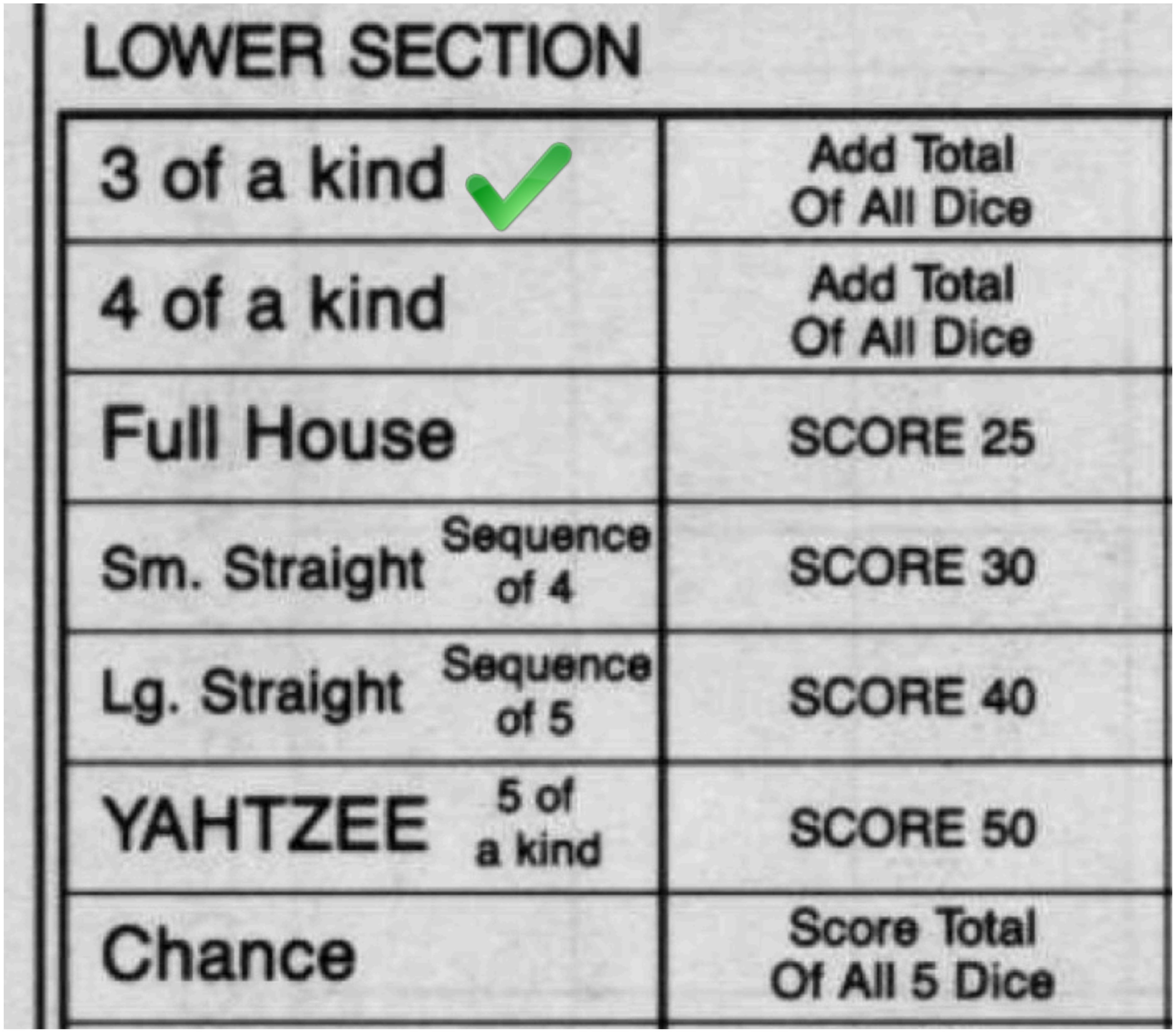
```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);
```



LOWER SECTION

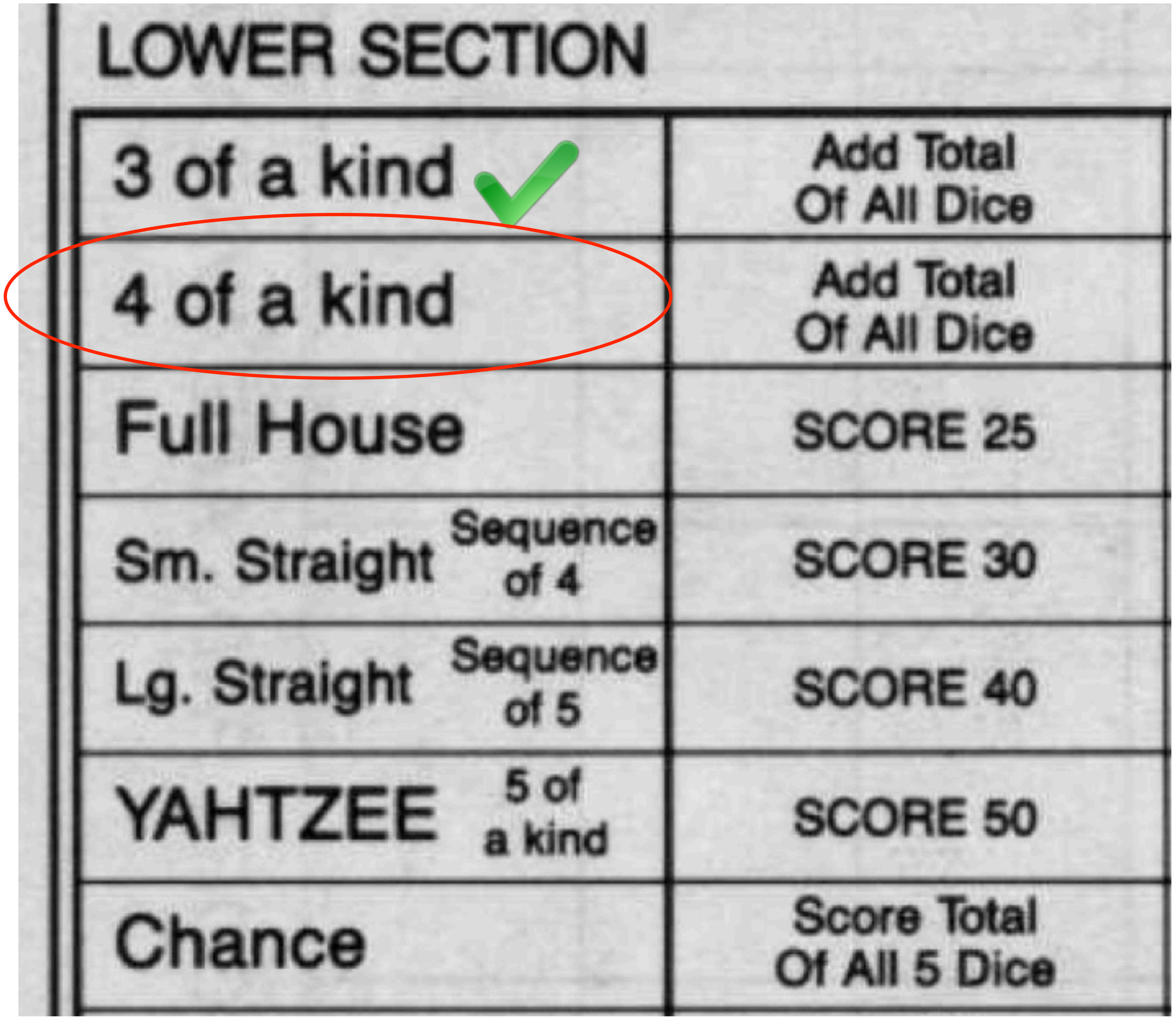
3 of a kind	Add Total Of All Dice
4 of a kind	Add Total Of All Dice
Full House	SCORE 25
Sm. Straight Sequence of 4	SCORE 30
Lg. Straight Sequence of 5	SCORE 40
YAHTZEE 5 of a kind	SCORE 50
Chance	Score Total Of All 5 Dice





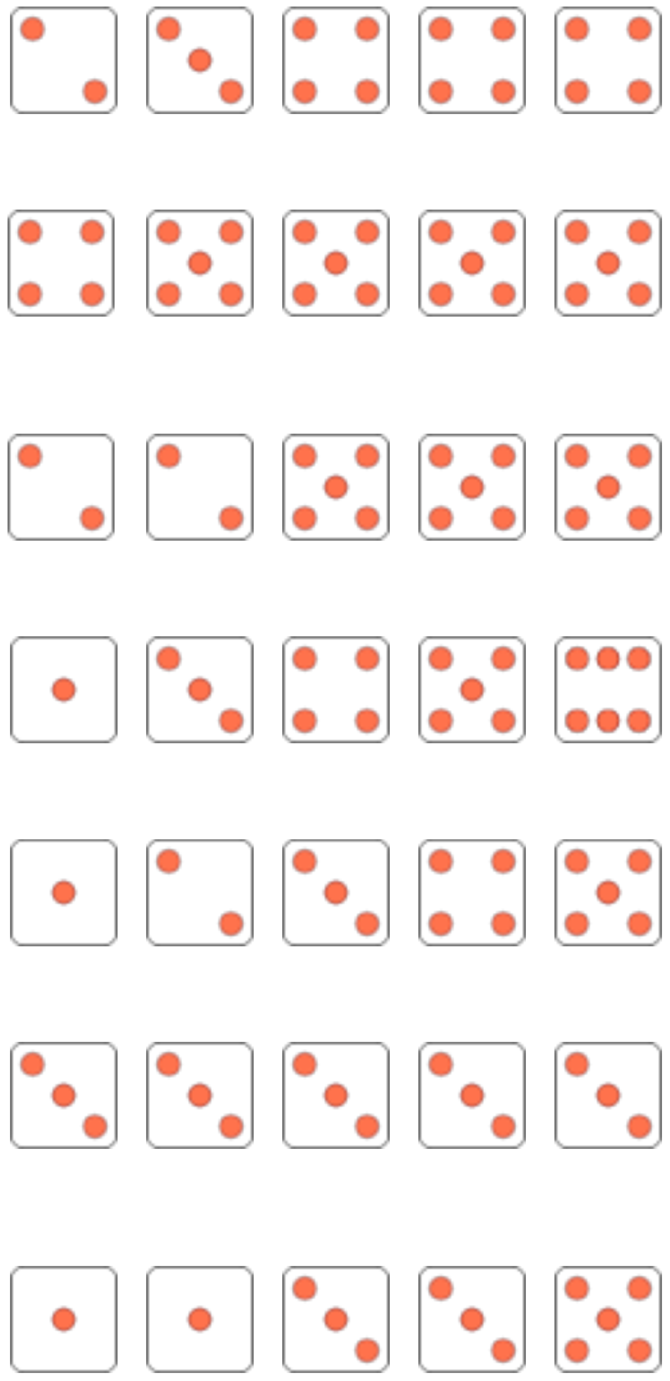
LOWER SECTION

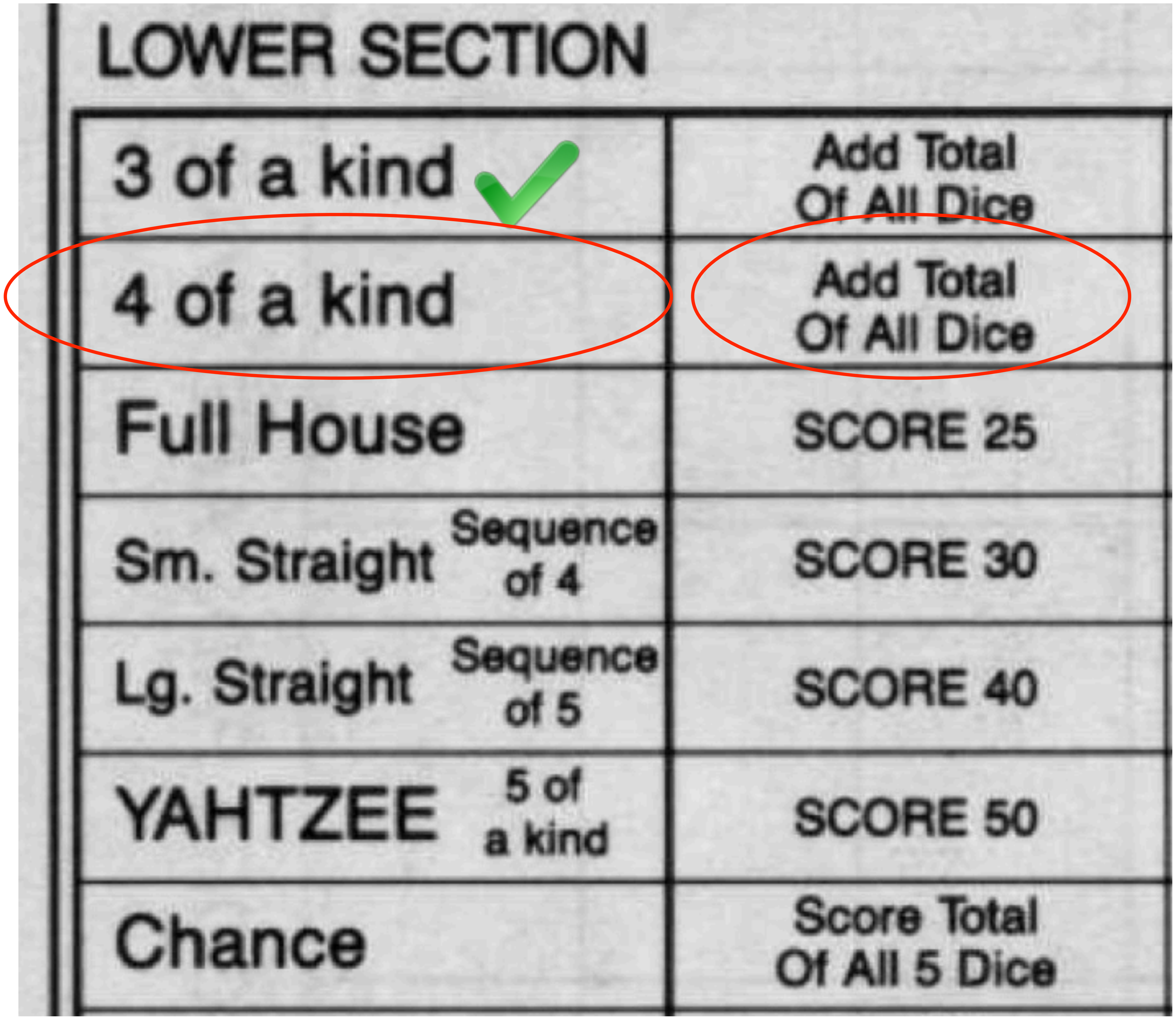
3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



LOWER SECTION

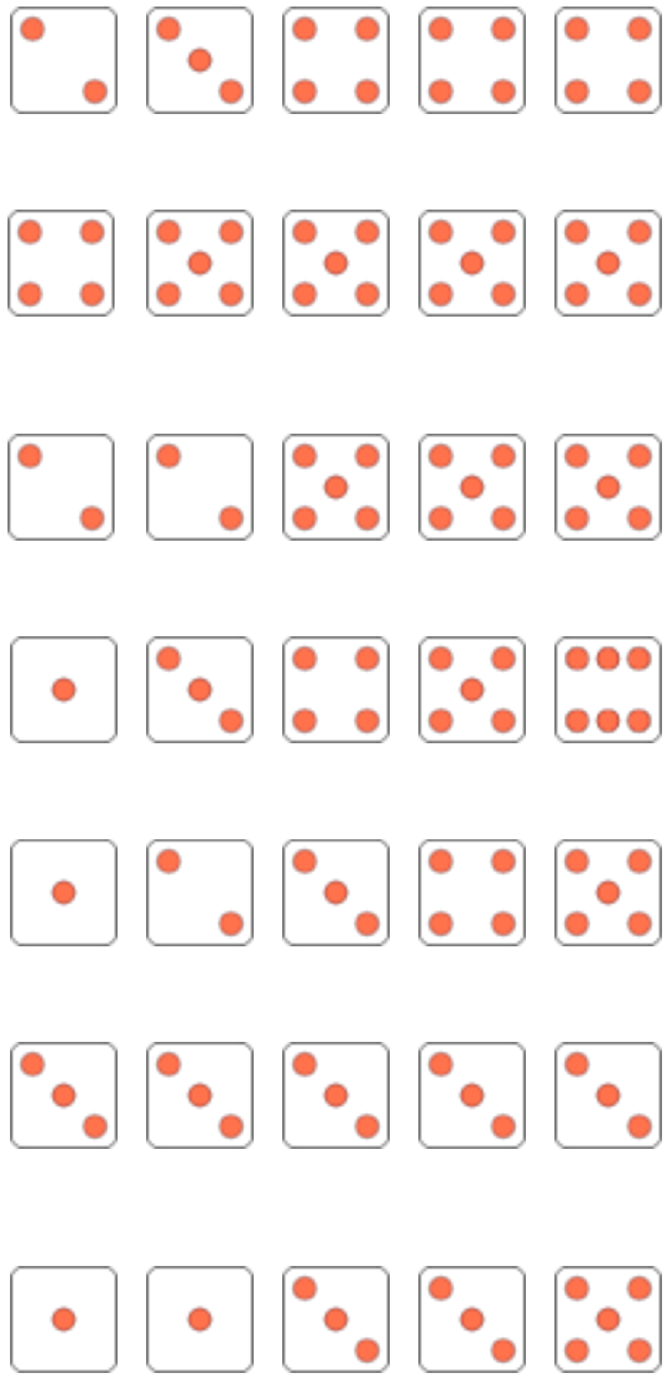
3 of a kind ✓	Add Total Of All Dice
4 of a kind	Add Total Of All Dice
Full House	SCORE 25
Sm. Straight Sequence of 4	SCORE 30
Lg. Straight Sequence of 5	SCORE 40
YAHTZEE 5 of a kind	SCORE 50
Chance	Score Total Of All 5 Dice





LOWER SECTION

3 of a kind ✓	Add Total Of All Dice
4 of a kind	Add Total Of All Dice
Full House	SCORE 25
Sm. Straight Sequence of 4	SCORE 30
Lg. Straight Sequence of 5	SCORE 40
YAHTZEE 5 of a kind	SCORE 50
Chance	Score Total Of All 5 Dice



yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return ??
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return ??
}
```

what should we return?

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return ??
}
```

what should we return?

Remember fail-fix-pass? First we want a failing test

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

```
Assertion failed: (score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9)
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

Fail - Fix - Pass

```
Assertion failed: (score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9)
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

Fail - Fix - Pass

```
Assertion failed: (score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9)
```

yahtzee.c

```c

int score_four_of_a_kind(const int dice[5])
{
    return 9;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 9;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 9;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

```
Yahtzee tests OK
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return 9;
}
```

Fail - **Fix** - Pass

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 9;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
```

yahtzee.c

```c

int score_four_of_a_kind(const int dice[5])
{
    return 9;
}

```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);

```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);

```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 9;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return 9;
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{



}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_fou



}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_fou



}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

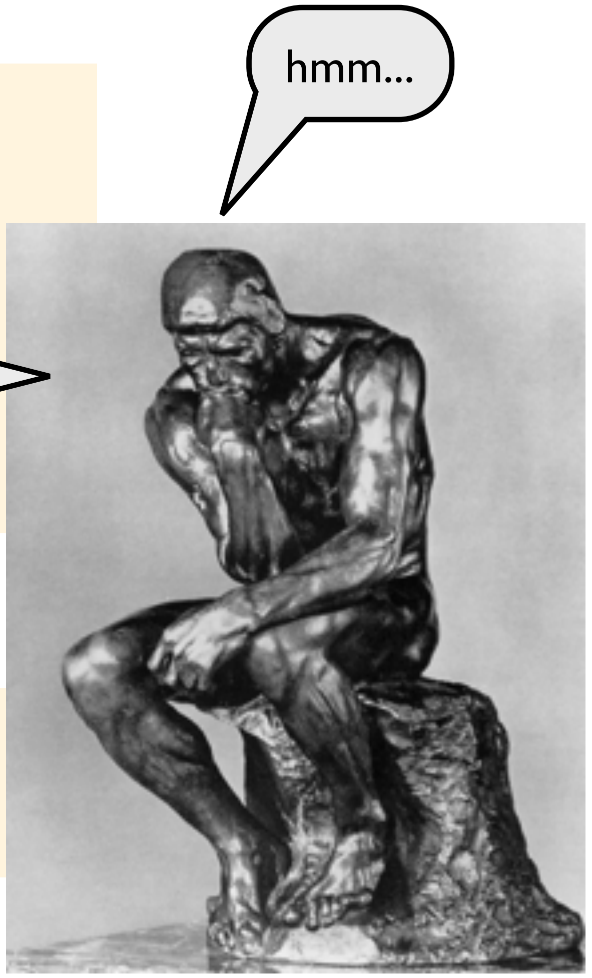
yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_fou


}
```

Perhaps we need a have_at_least_n_of_a_kind()?



yahtzee.h

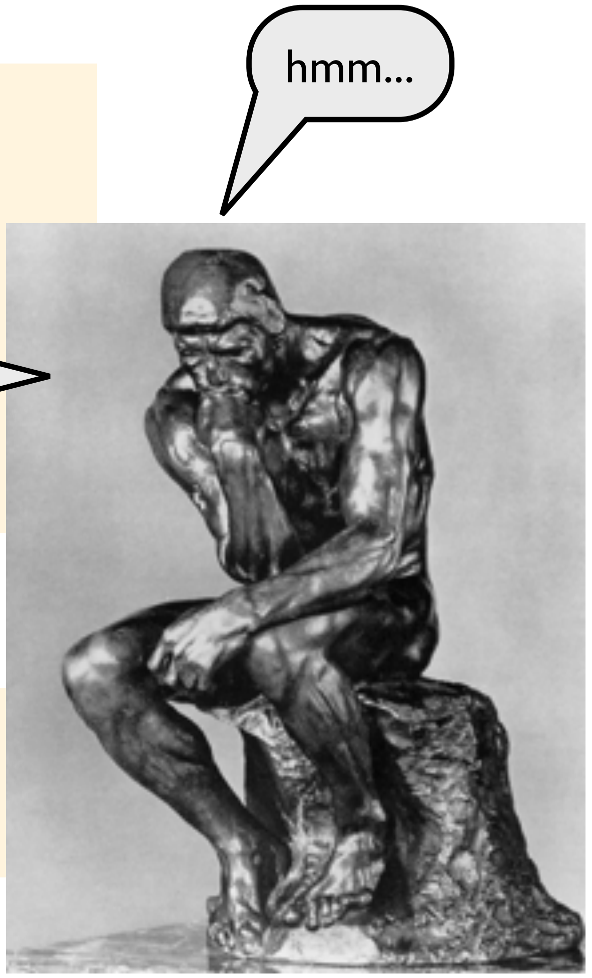
```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{



}
```



yahtzee.h

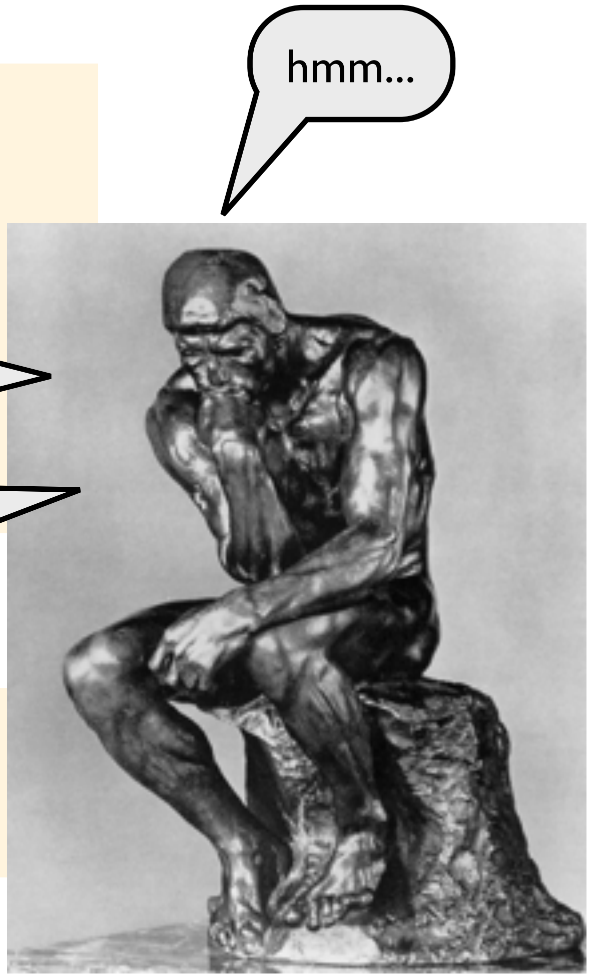
```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{



}
```



yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{



}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 9;



}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return 9;



}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return 9;


}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    //assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return 9;


}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    //assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

```
Yahtzee tests OK
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 9;


}
```

And now that we are at green we can do some **refactoring**

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    //assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

```
Yahtzee tests OK
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_three_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_three_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_n_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_n_of_a_kind(const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= 3)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_n_of_a_kind(int n, const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= n)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(3, dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.c

```c
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_n_of_a_kind(int n, const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= n)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(3, dice))
        return sum_of_dice(dice);
    return 0;
}
```

```
Yahtzee tests OK
```

yahtzee.c

```
...
static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_n_of_a_kind(int n, const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= n)
            return true;
    return false;
}

int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(3, dice))
        return sum_of_dice(dice);
    return 0;
}
```

And now we have prepared the ground for implementing score_four_of_a_kind()

```
Yahtzee tests OK
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return 9;


}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    //assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

```
Yahtzee tests OK
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    return 9;


}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    //assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

```
Yahtzee tests OK
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 9;


}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 9;


}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

```
Assertion failed: (score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9)
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 9;



}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

Fail - Fix - Pass

```
Assertion failed: (score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9)
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    return 9;


}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

Fail - Fix - Pass

```
Assertion failed: (score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9)
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(4, dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(4, dice))
        return sum_of_dice(dice);
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(4, dice))
        return sum_of_dice(dice);
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

```
Yahtzee tests OK
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(4, dice))
        return sum_of_dice(dice);
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(4, dice))
        return sum_of_dice(dice);
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
```

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```c
int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(4, dice))
        return sum_of_dice(dice);
    return 0;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
    assert(score_four_of_a_kind((int[5]){1,1,1,1,1}) == 5);
```

yahtzee.c

```
int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(4, dice))
        return sum_of_dice(dice);
    return 0;
}
```

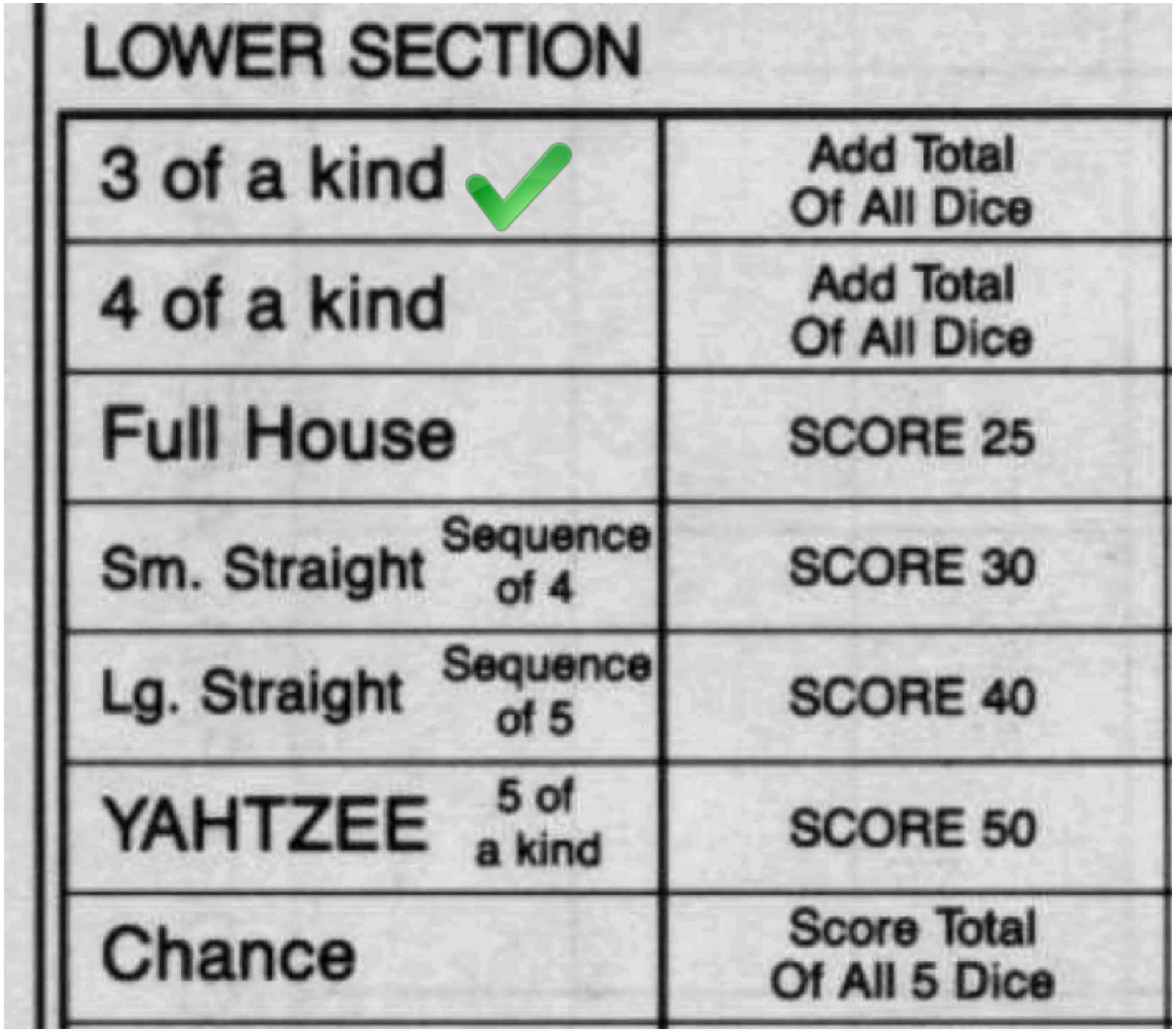
yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

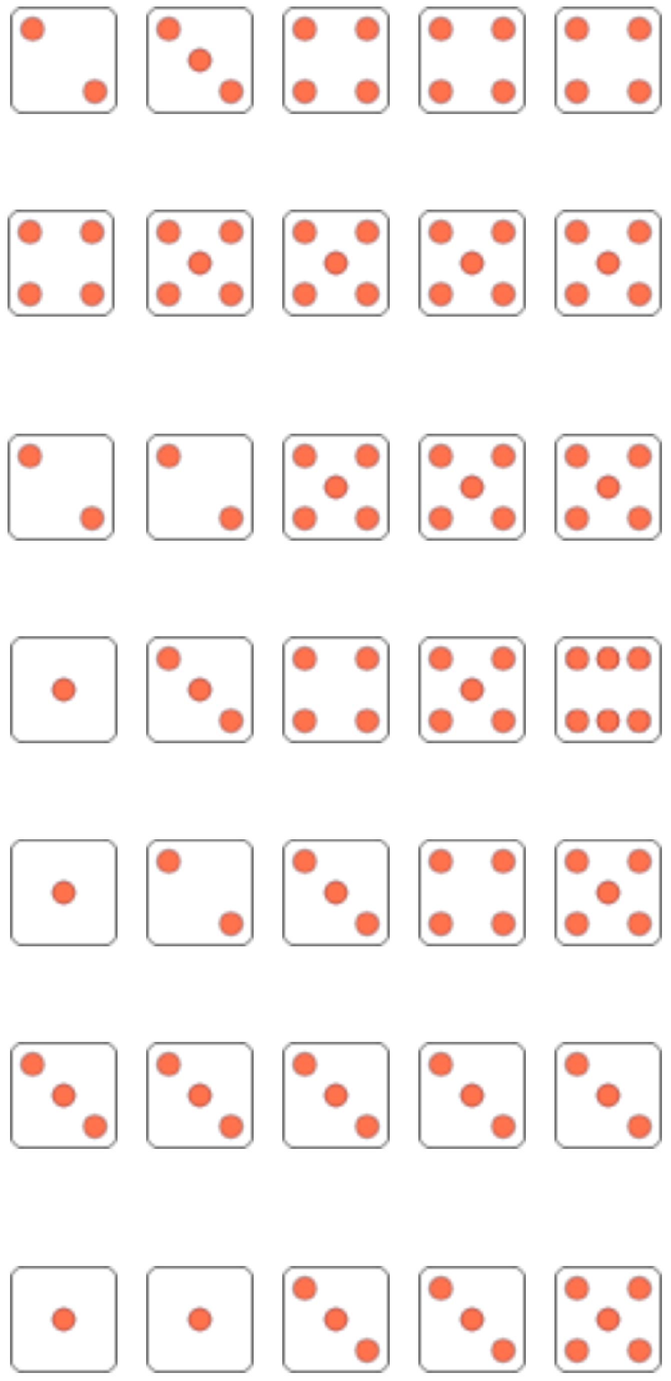
```
...
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,3,4}) == 3+3+4);
    assert(score_three_of_a_kind((int[5]){1,2,3,4,5}) == 0);
    assert(score_three_of_a_kind((int[5]){5,3,5,5,2}) == 15+5);
    assert(score_three_of_a_kind((int[5]){1,1,6,6,6}) == 18+2);
    assert(score_three_of_a_kind((int[5]){6,1,6,6,6}) == 18+7);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
    assert(score_four_of_a_kind((int[5]){1,1,1,1,1}) == 5);
```

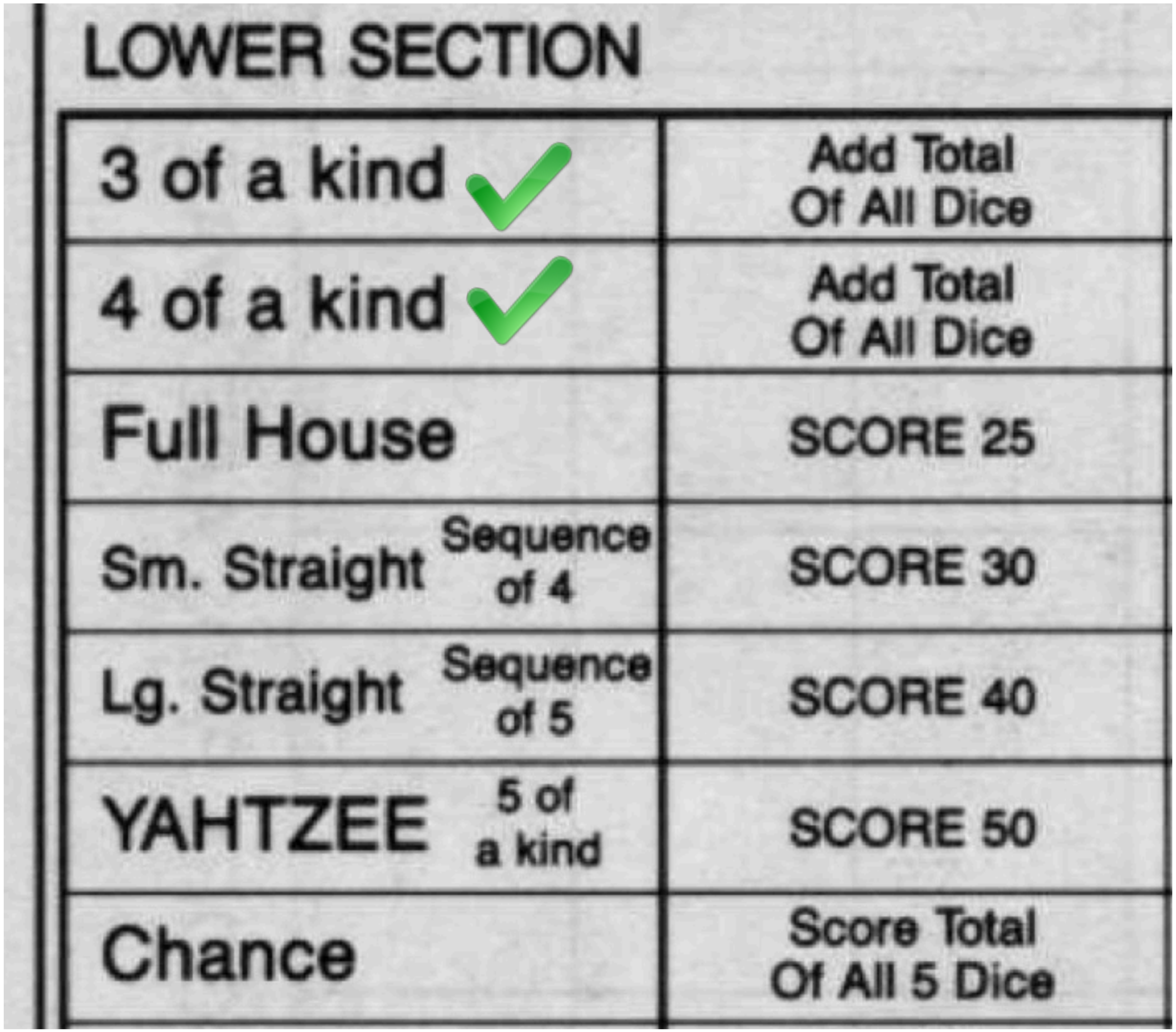
```
Yahtzee tests OK
```



LOWER SECTION

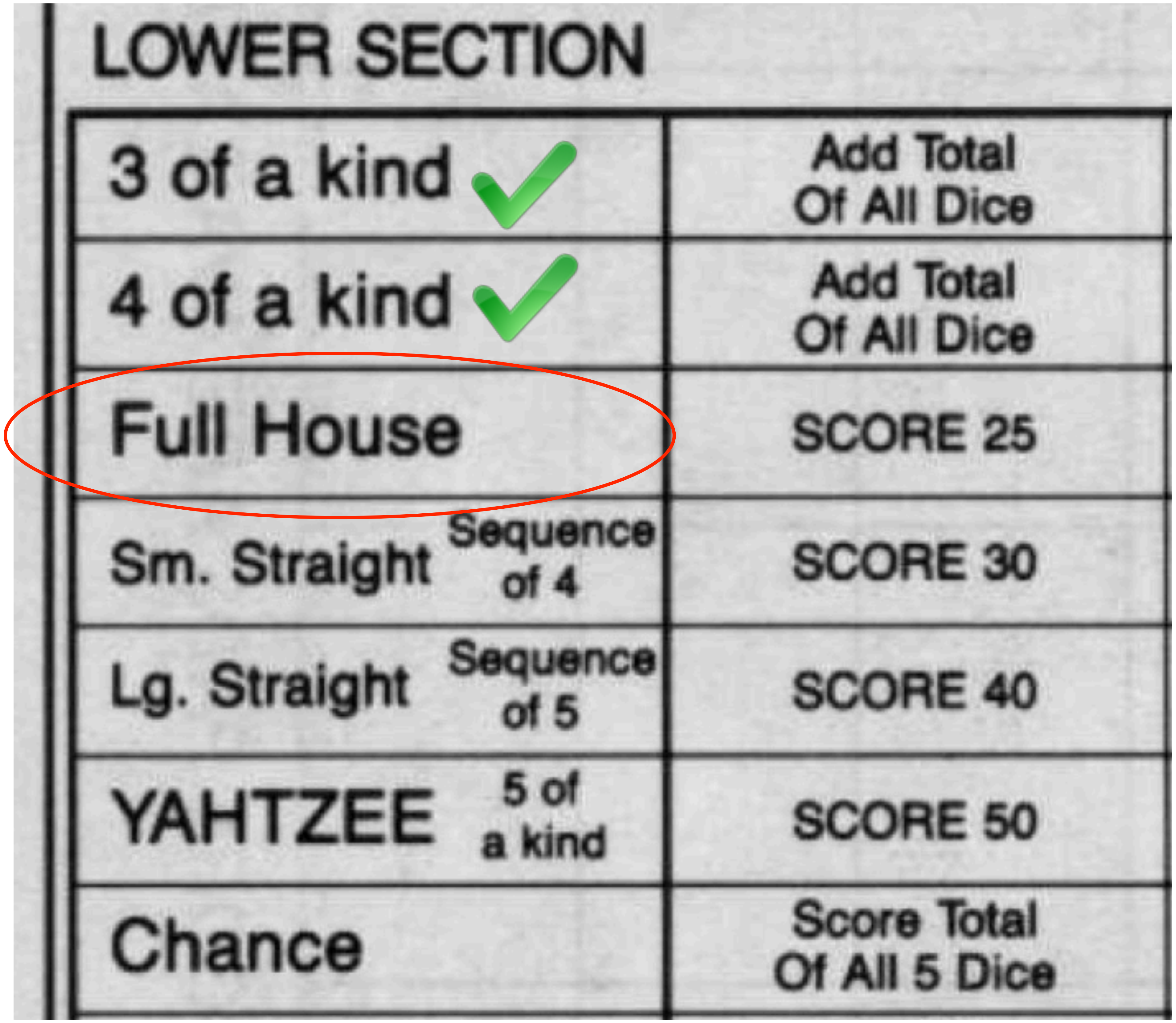
3 of a kind ✓	Add Total Of All Dice
4 of a kind	Add Total Of All Dice
Full House	SCORE 25
Sm. Straight Sequence of 4	SCORE 30
Lg. Straight Sequence of 5	SCORE 40
YAHTZEE 5 of a kind	SCORE 50
Chance	Score Total Of All 5 Dice





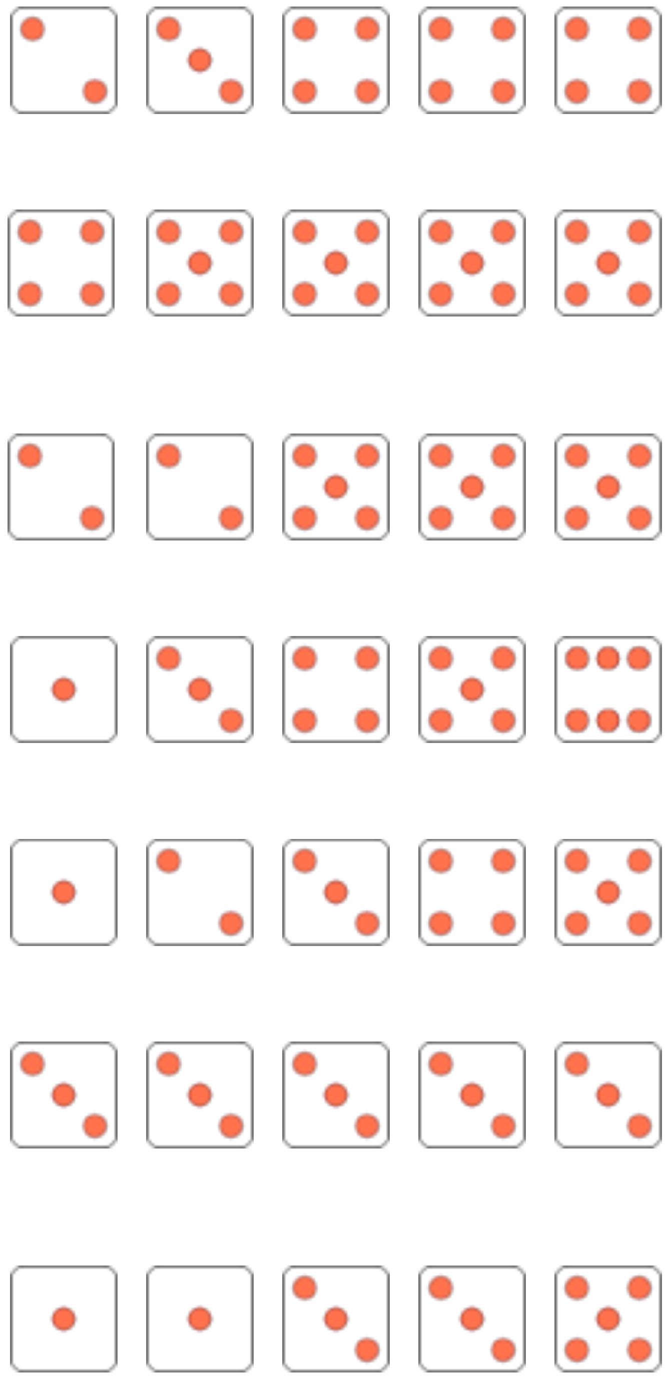
LOWER SECTION

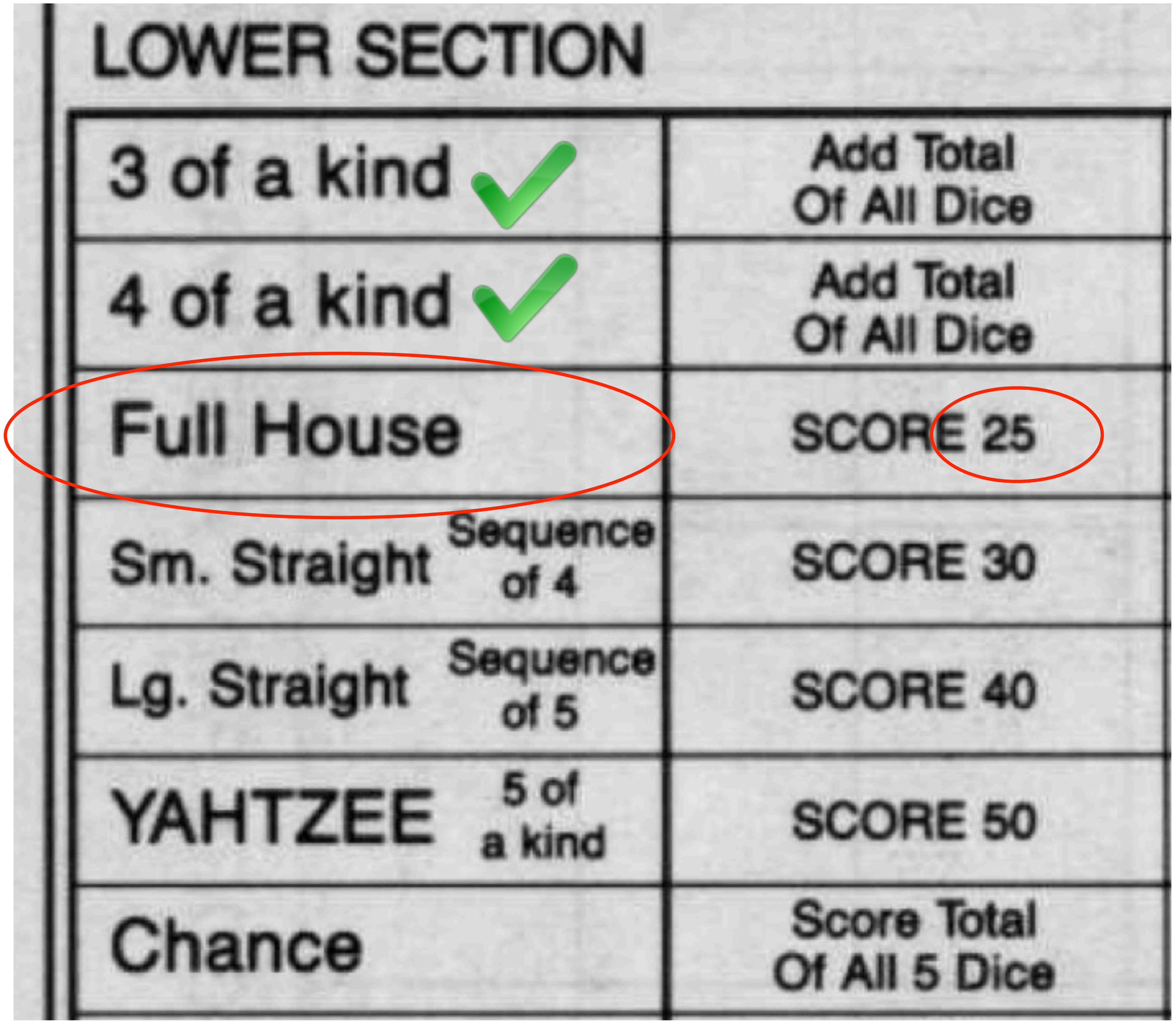
3 of a kind ✓	Add Total Of All Dice		⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice		⚃ ⚄ ⚄ ⚄ ⚄
Full House	SCORE 25		⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4	SCORE 30		⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40		⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50		⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice		⚀ ⚀ ⚂ ⚂ ⚄



LOWER SECTION

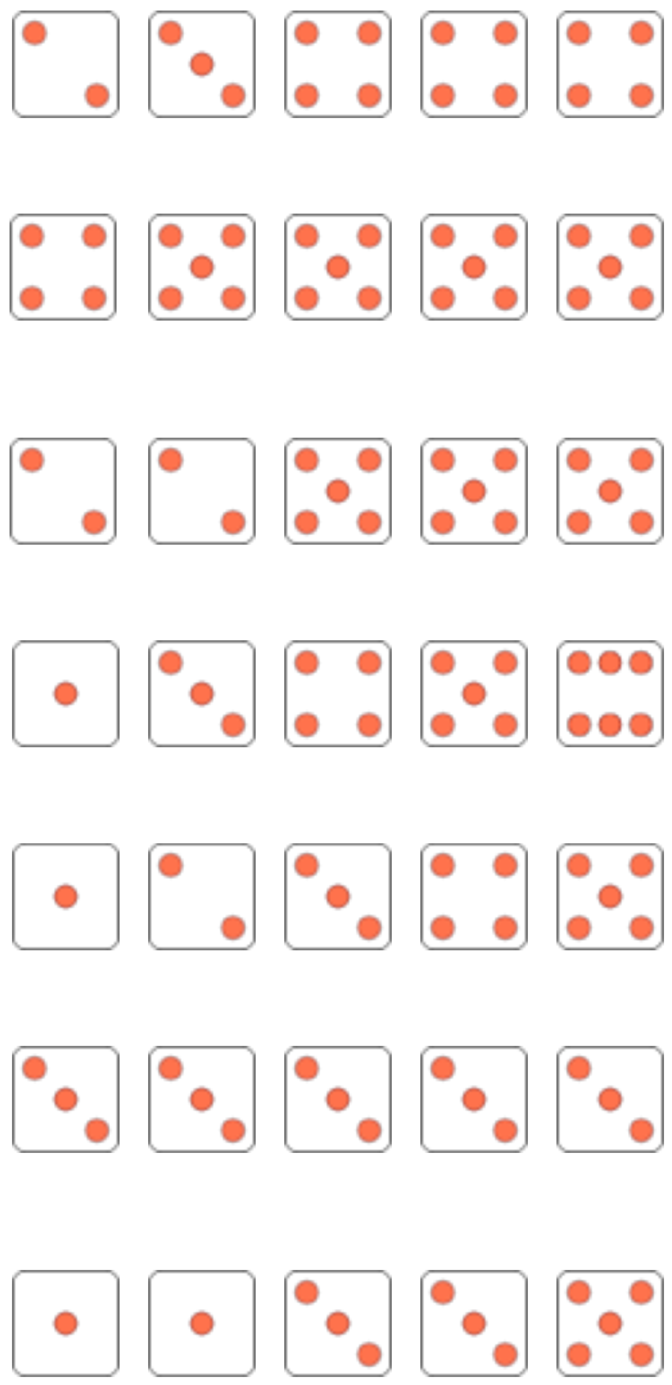
3 of a kind ✓		Add Total Of All Dice
4 of a kind ✓		Add Total Of All Dice
Full House		SCORE 25
Sm. Straight	Sequence of 4	SCORE 30
Lg. Straight	Sequence of 5	SCORE 40
YAHTZEE	5 of a kind	SCORE 50
Chance		Score Total Of All 5 Dice





LOWER SECTION

3 of a kind ✓	Add Total Of All Dice
4 of a kind ✓	Add Total Of All Dice
Full House	SCORE 25
Sm. Straight Sequence of 4	SCORE 30
Lg. Straight Sequence of 5	SCORE 40
YAHTZEE 5 of a kind	SCORE 50
Chance	Score Total Of All 5 Dice



yahtzee_test.c

yahtzee_test.c

yahtzee_test.c

```
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);

```

yahtzee_test.c

```c

    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);


```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c

    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);

```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

```
Assertion failed: (score_full_house((int[5]){3,3,3,5,5}) == 25)
```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

Fail - Fix - Pass

```
Assertion failed: (score_full_house((int[5]){3,3,3,5,5}) == 25)
```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 0;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

Fail - Fix - Pass

```
Assertion failed: (score_full_house((int[5]){3,3,3,5,5}) == 25)
```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 25;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 25;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 25;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

```
Yahtzee tests OK
```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 25;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 25;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
```

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 25;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);
```

yahtzee.c

```c
int score_full_house(const int dice[5])
{
    return 25;
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);
```

```
Assertion failed: (score_full_house((int[5]){3,3,1,5,5}) == 0)
```

yahtzee.c

```
int score_full_house(const int dice[5])
{
    return 25;
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);
```

Fail - Fix - Pass

```
Assertion failed: (score_full_house((int[5]){3,3,1,5,5}) == 0)
```

yahtzee.c

```
int score_full_house(const int dice[5])
{
    return 25;
}
```

Fix?

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);
```

Fail - Fix - Pass

```
Assertion failed: (score_full_house((int[5]){3,3,1,5,5}) == 0)
```

yahtzee.c

```
int score_full_house(const int dice[5])
{
    return 25;
}
```

Fix?

We know how to do this!

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);
```

Fail - Fix - Pass

```
Assertion failed: (score_full_house((int[5]){3,3,1,5,5}) == 0)
```

yahtzee.c

```
int score_full_house(const int dice[5])
{
    return 25;
}
```

Fix?

We know how to do this!

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);
```

Fail - Fix - Pass

```
Assertion failed: (score_full_house((int[5]){3,3,1,5,5}) == 0)
```

yahtzee.c

```
static bool have_exactly_n_of_a_kind(int n, const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == n)
            return true;
    return false;
}

int score_full_house(const int dice[5])
{
    if (have_exactly_n_of_a_kind(3, dice) &&
        have_exactly_n_of_a_kind(2, dice))
        return 25;
    return 0;
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);
```

yahtzee.c

```c
static bool have_exactly_n_of_a_kind(int n, const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == n)
            return true;
    return false;
}

int score_full_house(const int dice[5])
{
    if (have_exactly_n_of_a_kind(3, dice) &&
        have_exactly_n_of_a_kind(2, dice))
        return 25;
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);
```

yahtzee.c

```c
static bool have_exactly_n_of_a_kind(int n, const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == n)
            return true;
    return false;
}

int score_full_house(const int dice[5])
{
    if (have_exactly_n_of_a_kind(3, dice) &&
        have_exactly_n_of_a_kind(2, dice))
        return 25;
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);
```

```
Yahtzee tests OK
```

yahtzee.c

```
static bool have_exactly_n_of_a_kind(int n, const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == n)
            return true;
    return false;
}

int score_full_house(const int dice[5])
{
    if (have_exactly_n_of_a_kind(3, dice) &&
        have_exactly_n_of_a_kind(2, dice))
        return 25;
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee.h

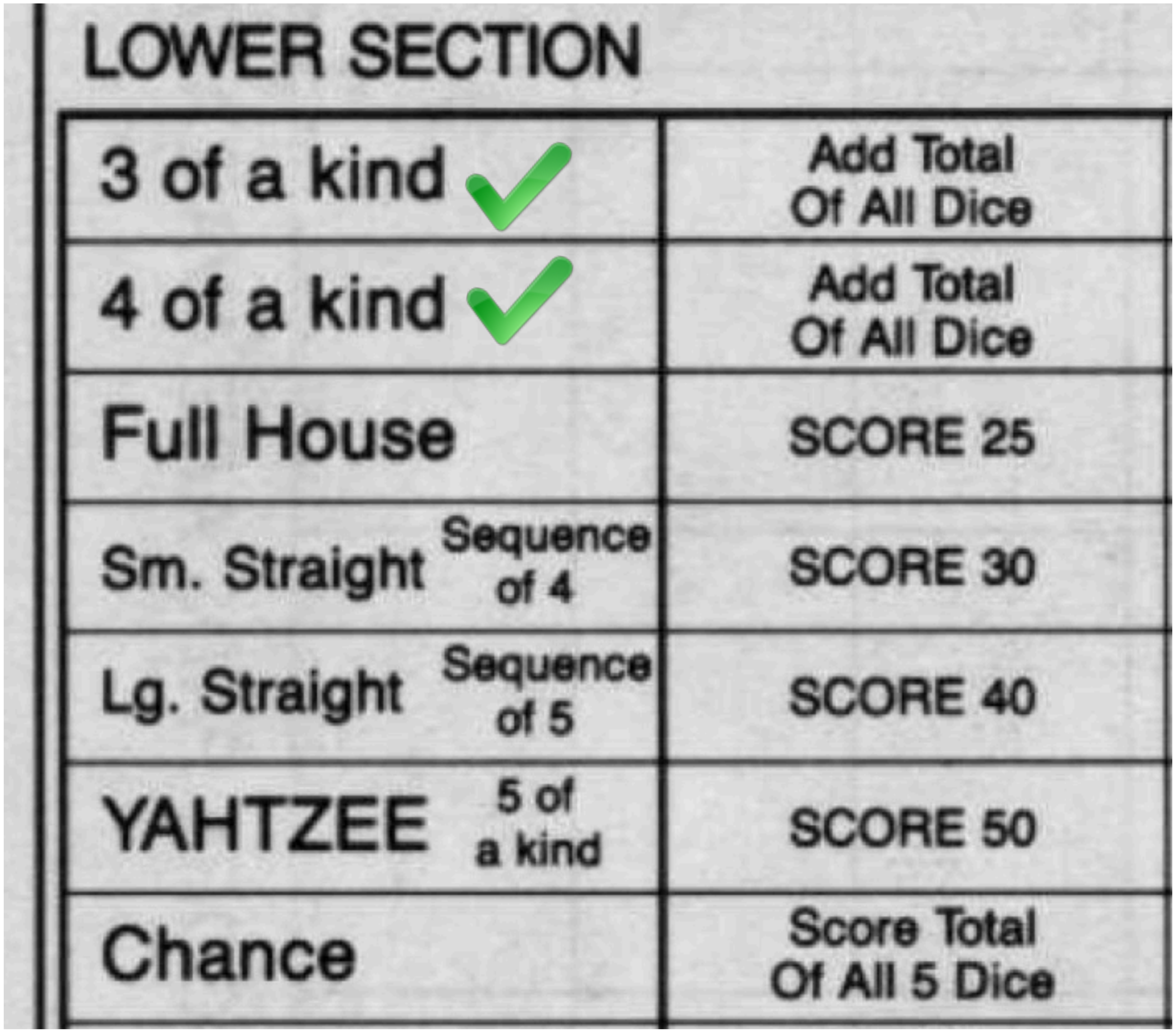
```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
```

yahtzee_test.c

```
    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);
```

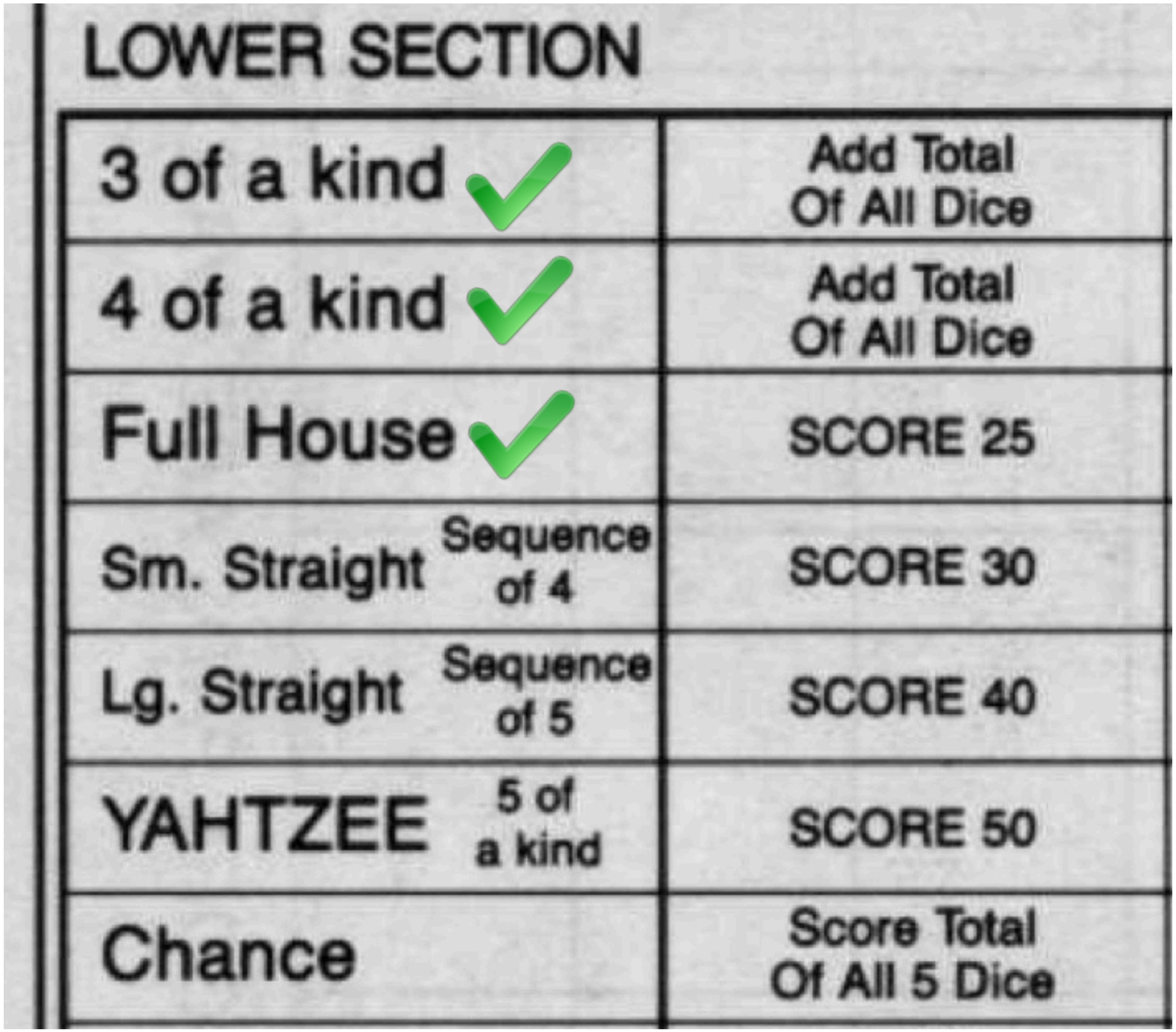
Fail - Fix - **Pass**

```
Yahtzee tests OK
```



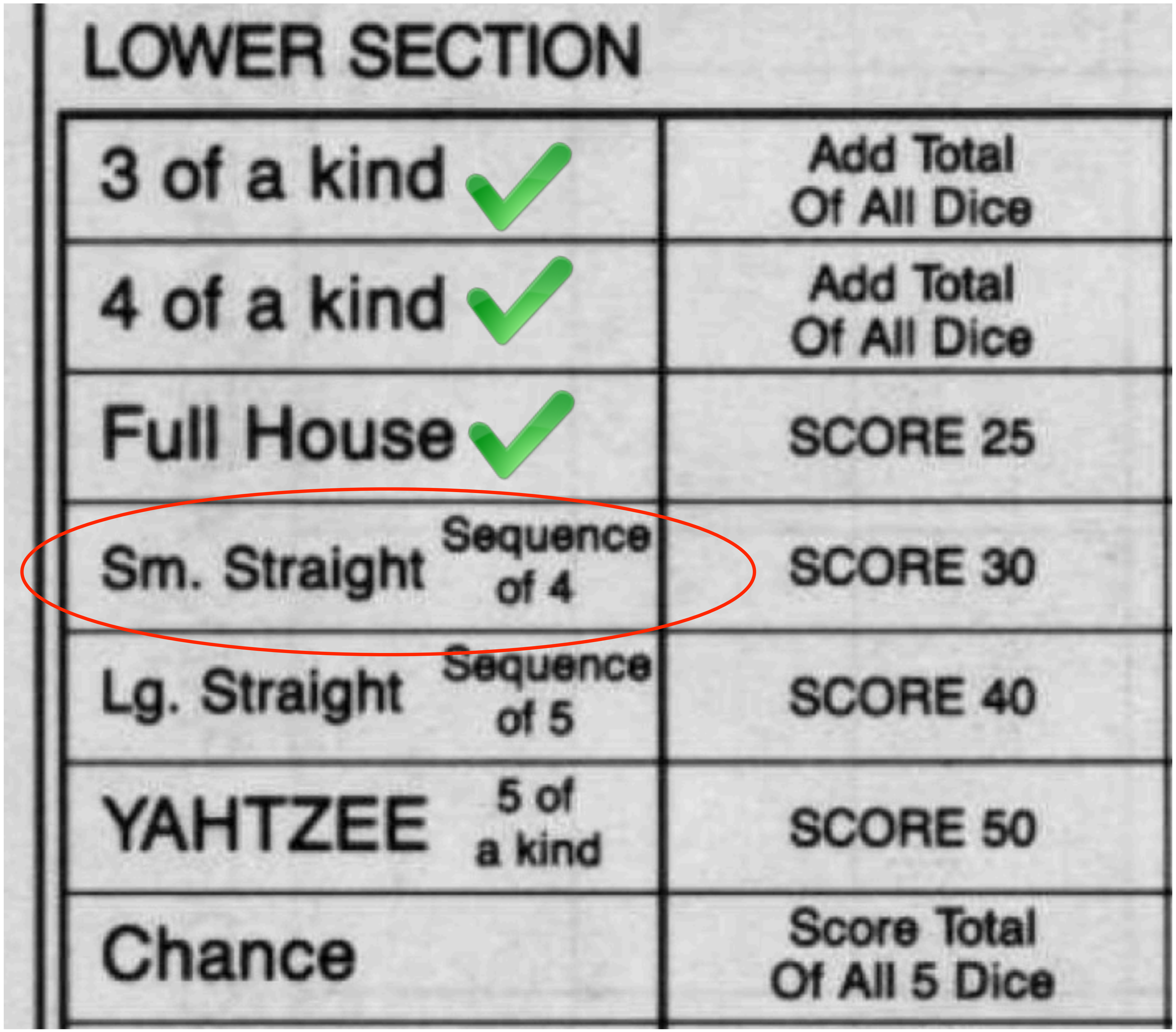
LOWER SECTION

3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



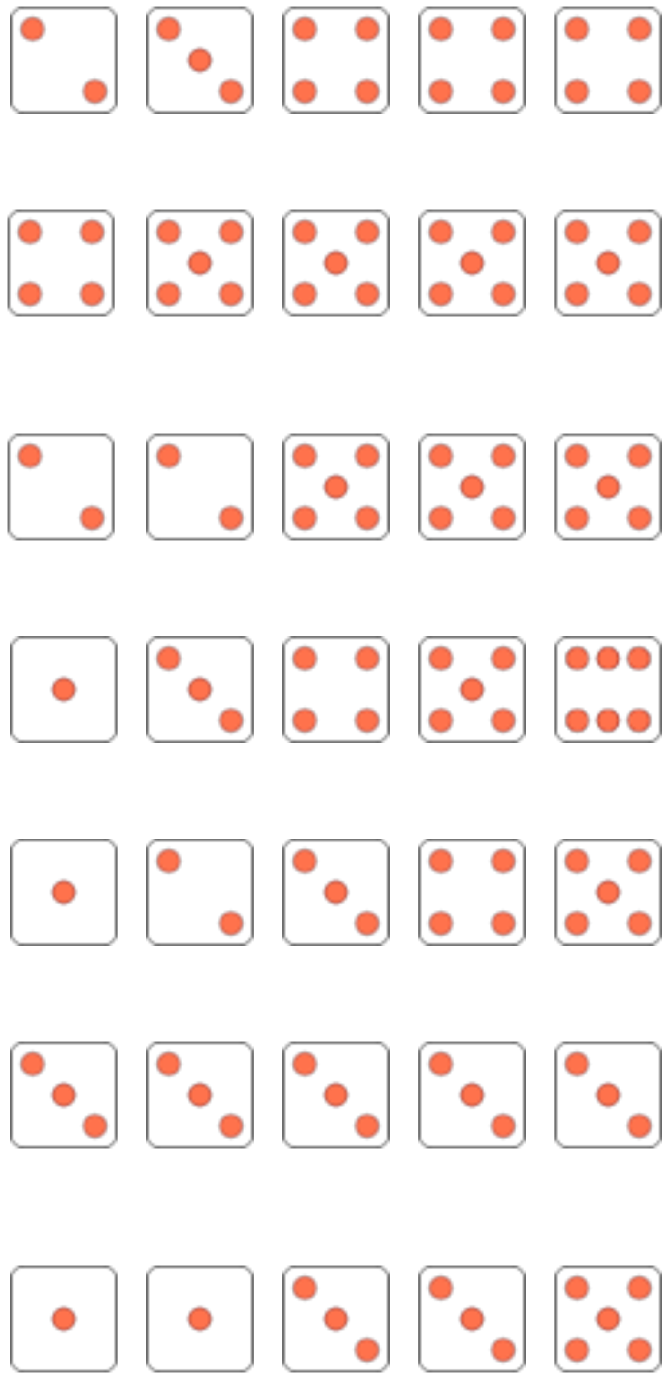
LOWER SECTION

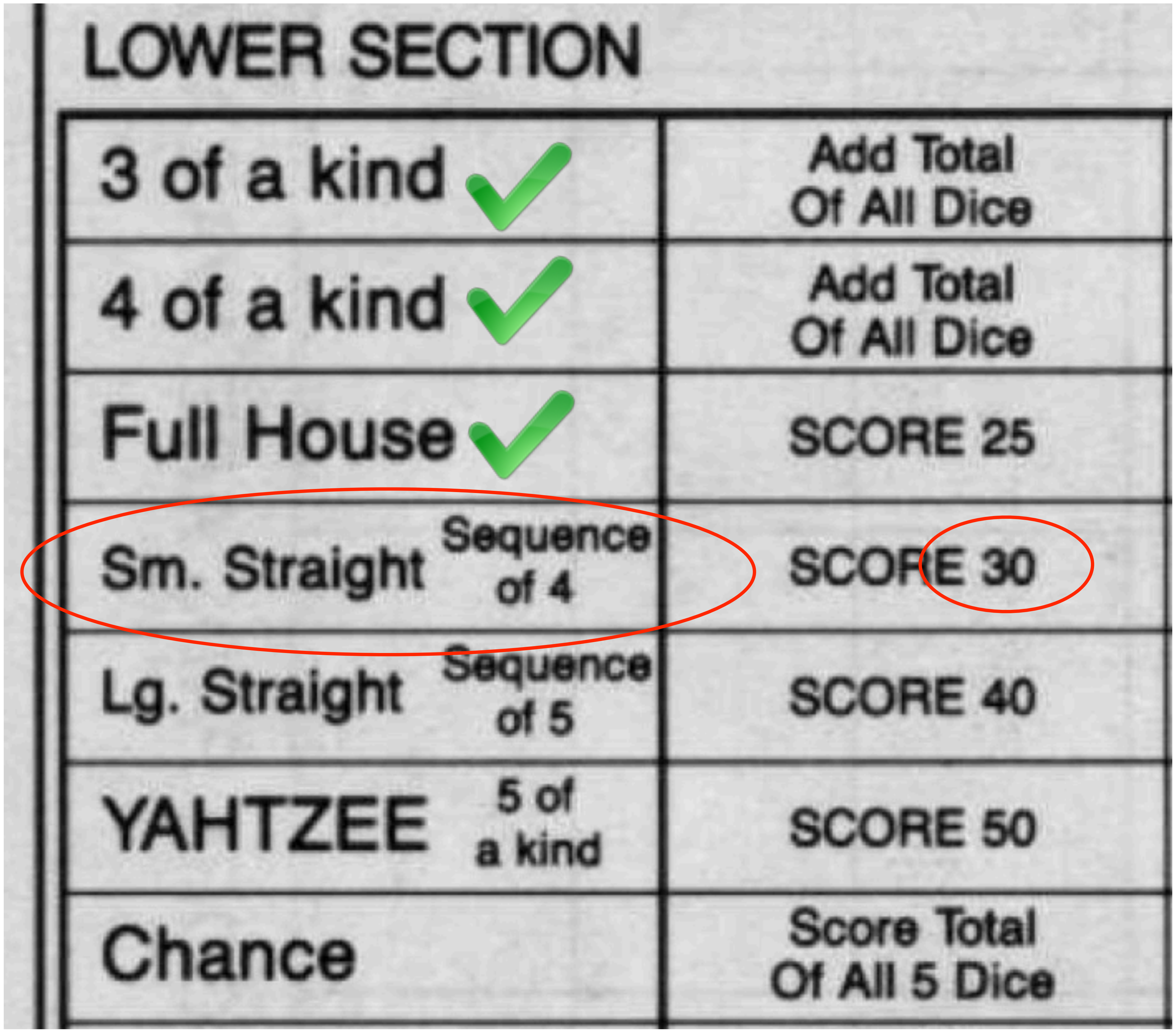
3 of a kind ✓	Add Total Of All Dice		⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice		⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓	SCORE 25		⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4	SCORE 30		⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40		⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50		⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice		⚀ ⚀ ⚂ ⚂ ⚄



LOWER SECTION

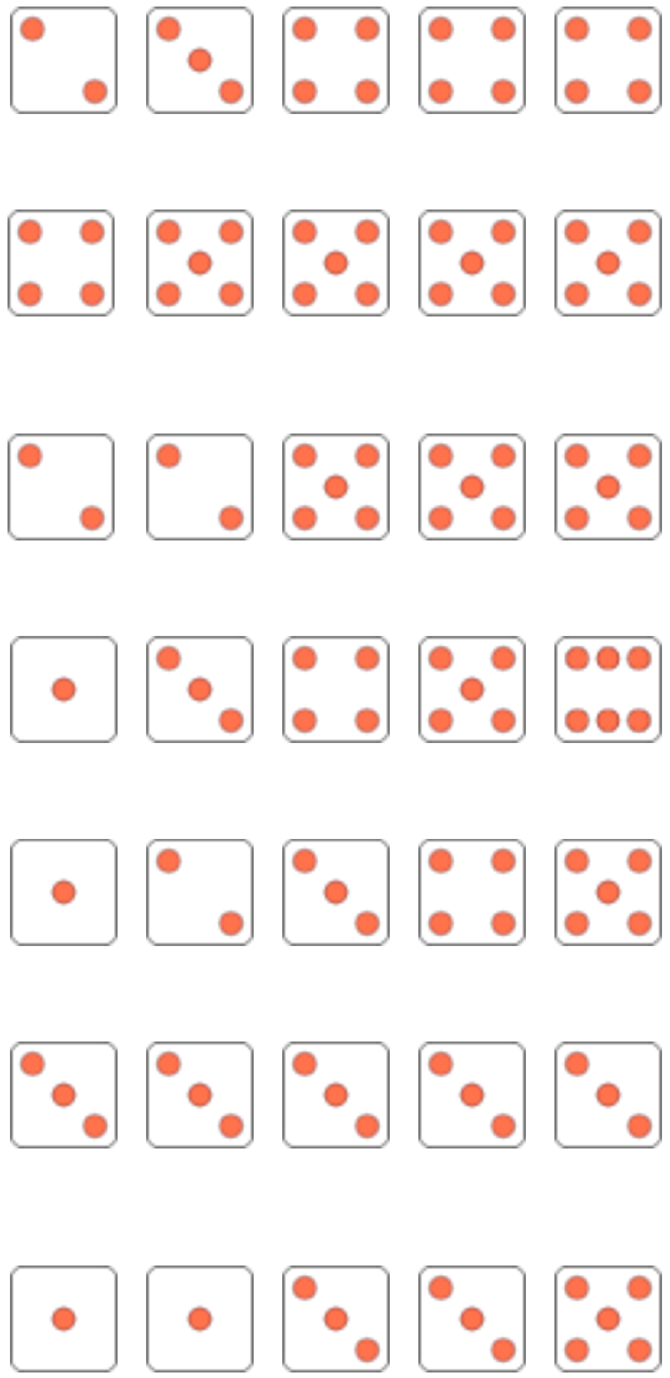
3 of a kind ✓	Add Total Of All Dice
4 of a kind ✓	Add Total Of All Dice
Full House ✓	SCORE 25
Sm. Straight Sequence of 4	SCORE 30
Lg. Straight Sequence of 5	SCORE 40
YAHTZEE 5 of a kind	SCORE 50
Chance	Score Total Of All 5 Dice





LOWER SECTION

3 of a kind ✓	Add Total Of All Dice
4 of a kind ✓	Add Total Of All Dice
Full House ✓	SCORE 25
Sm. Straight Sequence of 4	SCORE 30
Lg. Straight Sequence of 5	SCORE 40
YAHTZEE 5 of a kind	SCORE 50
Chance	Score Total Of All 5 Dice



yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

yahtzee.c

```c
int score_small_straight(const int dice[5])
{
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

yahtzee.c

```c
int score_small_straight(const int dice[5])
{
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

```
Assertion failed: (score_small_straight((int[5]){1,2,3,4,1}) == 30)
```

yahtzee.c

```
int score_small_straight(const int dice[5])
{
    return 0;
}
```

It is OK to plan ahead when implenting a fix... so...

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

```
Assertion failed: (score_small_straight((int[5]){1,2,3,4,1}) == 30)
```

yahtzee.c

```
int score_small_straight(const int dice[5])
{
    return 0;
}
```

It is OK to plan ahead when implenting a fix... so...

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

```
Assertion failed: (score_small_straight((int[5]){1,2,3,4,1}) == 30)
```

yahtzee.c

```
int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

It is OK to plan ahead when implenting a fix... so...

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

yahtzee.c

```
int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

It is OK to plan ahead when implenting a fix... so...

we know that we need **five in a row** soon. So therefor we do **n in a row** now

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

yahtzee.c

```
int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

It is OK to plan ahead when implenting a fix... so...

we know that we need **five in a row** soon. So therefor we do **n in a row** now

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

```
Assertion failed: (score_small_straight((int[5]){1,2,3,4,1}) == 30)
```

yahtzee.c

```c
int have_at_least_n_in_a_row(int n, const int dice[5])
{











}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

yahtzee.c

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{



      ?



}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

yahtzee.c

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{

          ?

}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

But how to implement this?

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

yahtzee.c

```c
int have_at_least_n_in_a_row(int n, const int dice[5])
{

          ?

}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

But how to implement this?

Use your head and think for a while...
TDD is **not** about not thinking!

yahtzee_test.c

```c
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

yahtzee.c

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{

        ?

}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

But how to implement this?

Use your head and think for a while...
TDD is **not** about not thinking!

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

```
Assertion failed: (score_small_straight((int[5]){1,2,3,4,1}) == 30)
```

yahtzee.c

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

```
Assertion failed: (score_small_straight((int[5]){1,2,3,4,1}) == 30)
```

yahtzee.c

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

Fail - Fix - Pass

```
Assertion failed: (score_small_straight((int[5]){1,2,3,4,1}) == 30)
```

yahtzee.c

```c
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee_test.c

```c
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

Fail - Fix - Pass

```
Assertion failed: (score_small_straight((int[5]){1,2,3,4,1}) == 30)
```

yahtzee.c

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
```

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```c
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
    assert(score_small_straight((int[5]){1,1,1,1,1}) == 0);
```

yahtzee.c

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

yahtzee_test.c

```
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
    assert(score_small_straight((int[5]){1,1,1,1,1}) == 0);
```

```
Yahtzee tests OK
```

yahtzee.c

```c
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
    assert(score_small_straight((int[5]){1,1,1,1,1}) == 0);
    assert(score_small_straight((int[5]){6,5,4,3,2}) == 30);
```

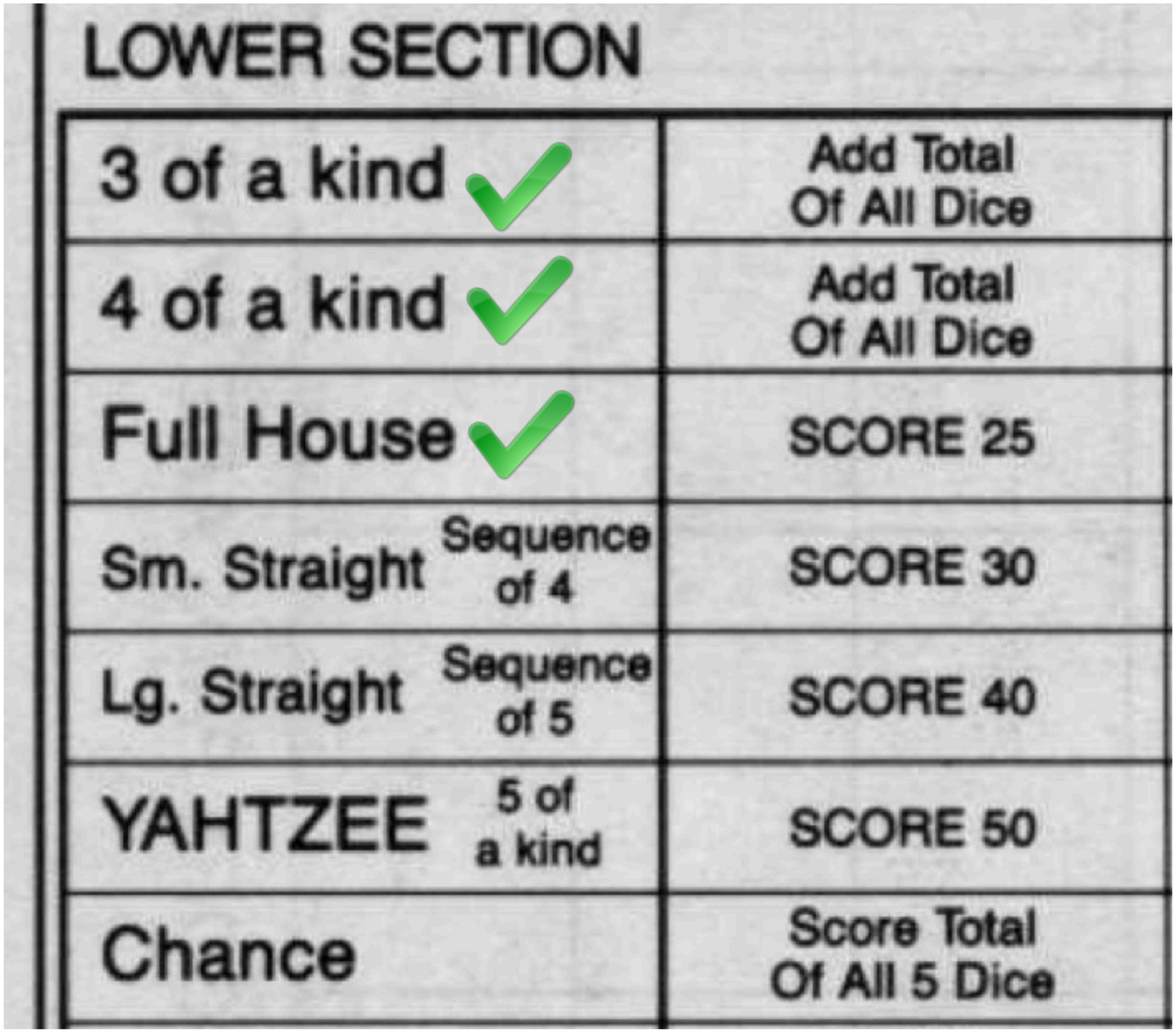
yahtzee.c

```c
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}
```

yahtzee_test.c

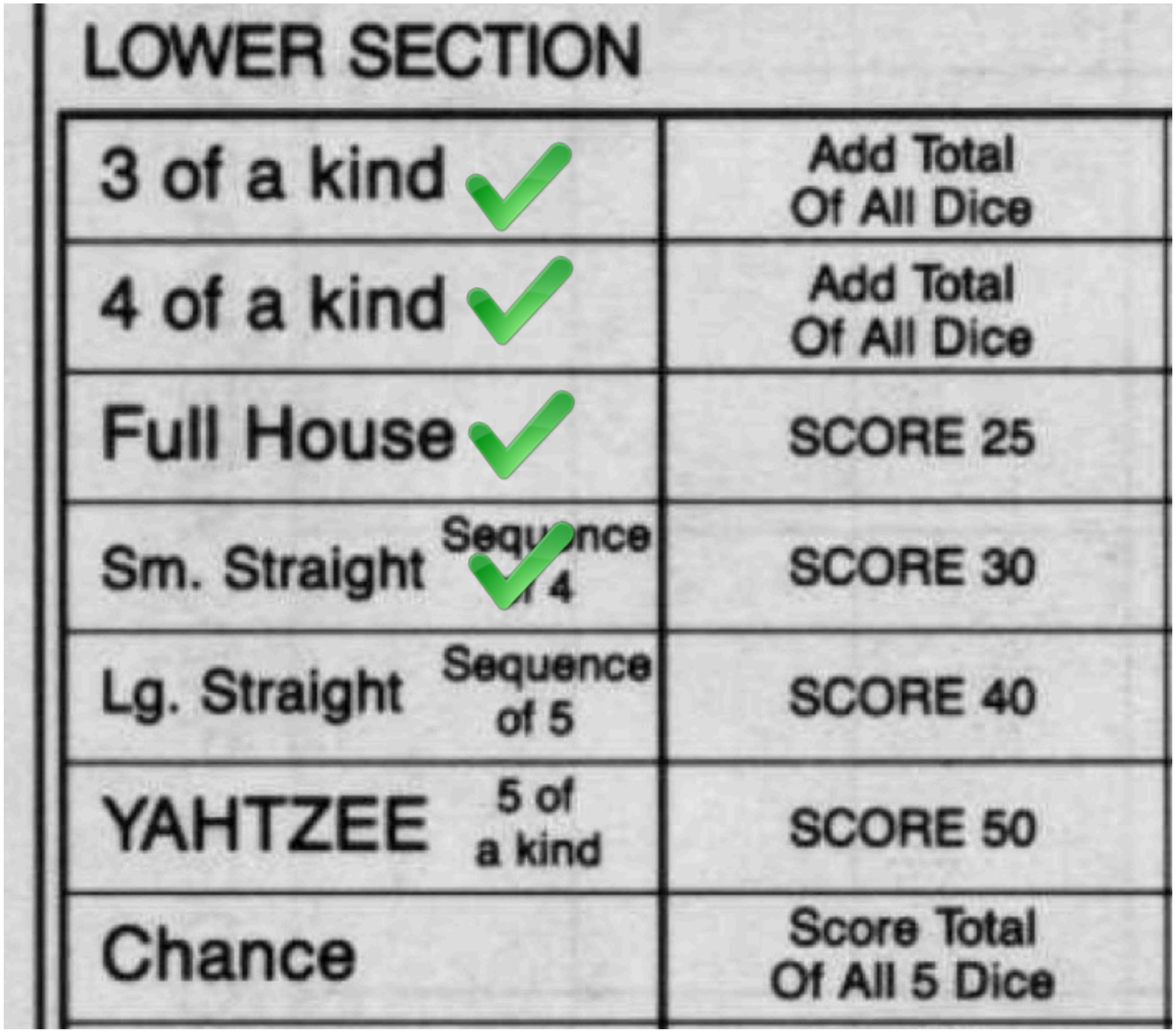
```c
    assert(score_small_straight((int[5]){1,2,3,4,1}) == 30);
    assert(score_small_straight((int[5]){1,1,1,1,1}) == 0);
    assert(score_small_straight((int[5]){6,5,4,3,2}) == 30);
```

```
Yahtzee tests OK
```



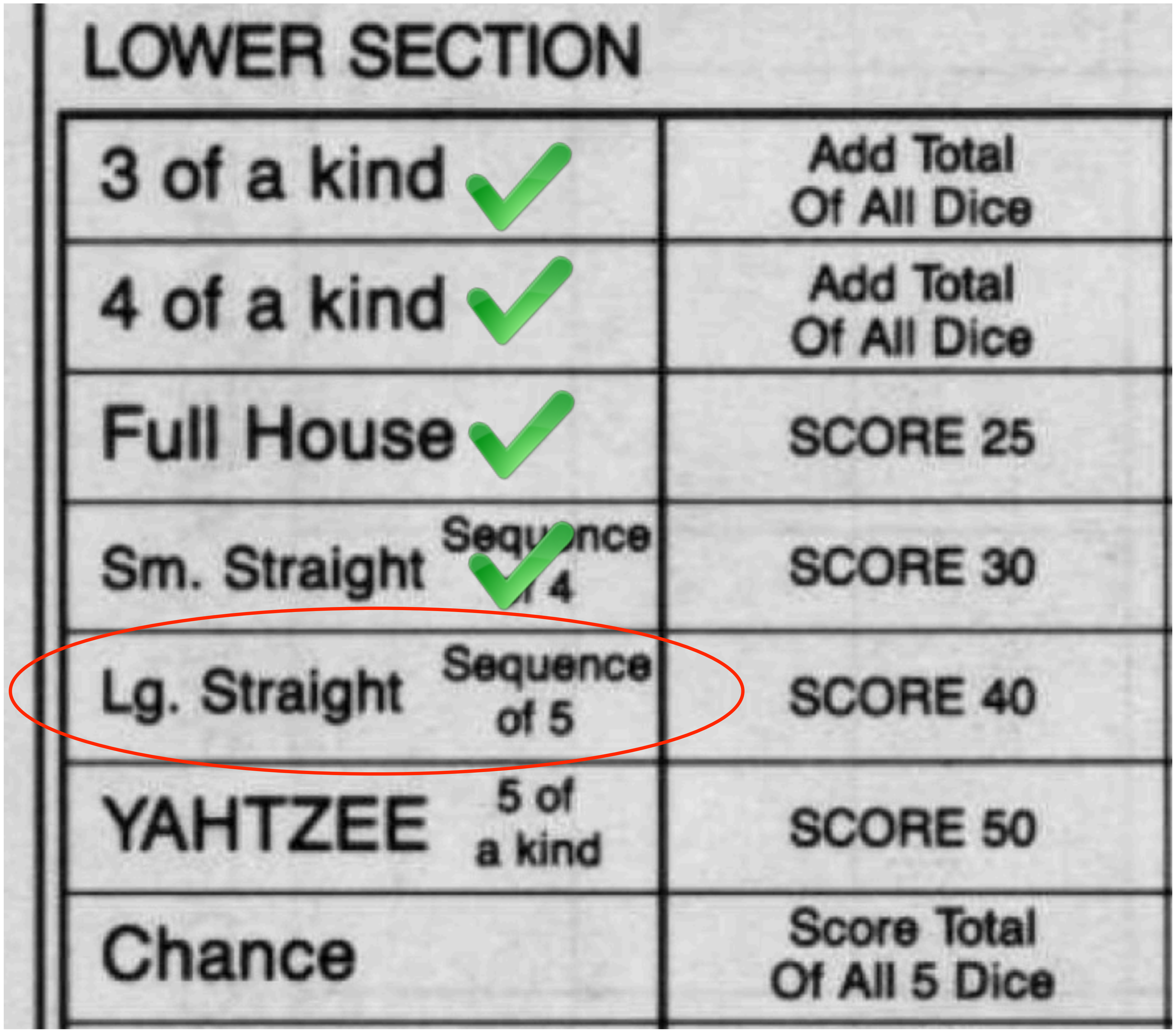
LOWER SECTION

3 of a kind ✓		Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓		Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓		SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight	Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight	Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE	5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance		Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



LOWER SECTION

3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4 ✓	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



LOWER SECTION

3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4 ✓	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄

yahtzee_test.c

```
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 30);
```

yahtzee.c

```
int score_large_straight(const int dice[5])
{
    return 0;
}
```

yahtzee.c

```c
int score_large_straight(const int dice[5])
{
    return 0;
}
```

```
Assertion failed: (score_large_straight((int[5]){1,2,3,4,5}) == 30)
```

yahtzee.c

```c
int score_large_straight(const int dice[5])
{
    return 0;
}
```

Fail - Fix - Pass

```
Assertion failed: (score_large_straight((int[5]){1,2,3,4,5}) == 30)
```

yahtzee.c

```c
int score_large_straight(const int dice[5])
{
    return 0;
}
```

yahtzee.c

```c
int score_large_straight(const int dice[5])
{
    return 0;
}
```

yahtzee.c

```c
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

Fail - **Fix** - Pass

What?

Inconceivable

```
Assertion failed: (score_large_straight((int[5]){1,2,3,4,5}) == 30)
```

yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

Fail - **Fix** - Pass

What?
Inconceivable

```
Assertion failed: (score_large_straight((int[5]){1,2,3,4,5}) == 30)
```

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}
```

yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

Fail - **Fix** - Pass

What?
Inconceivable

```
Assertion failed: (score_large_straight((int[5]){1,2,3,4,5}) == 30)
```

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}
```

yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

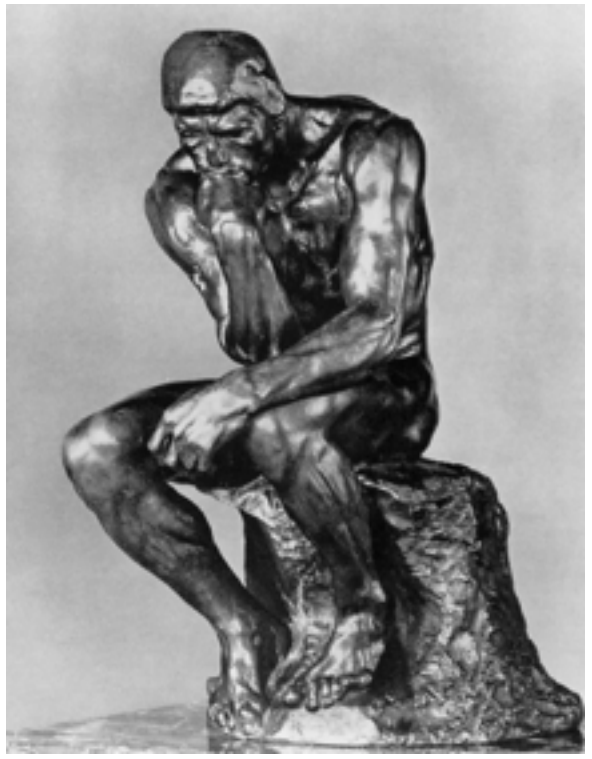
Fail - **Fix** - Pass

What?
Inconceivable

```
Assertion failed: (score_large_straight((int[5]){1,2,3,4,5}) == 30)
```

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}
```

hmm...



yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee_test.c

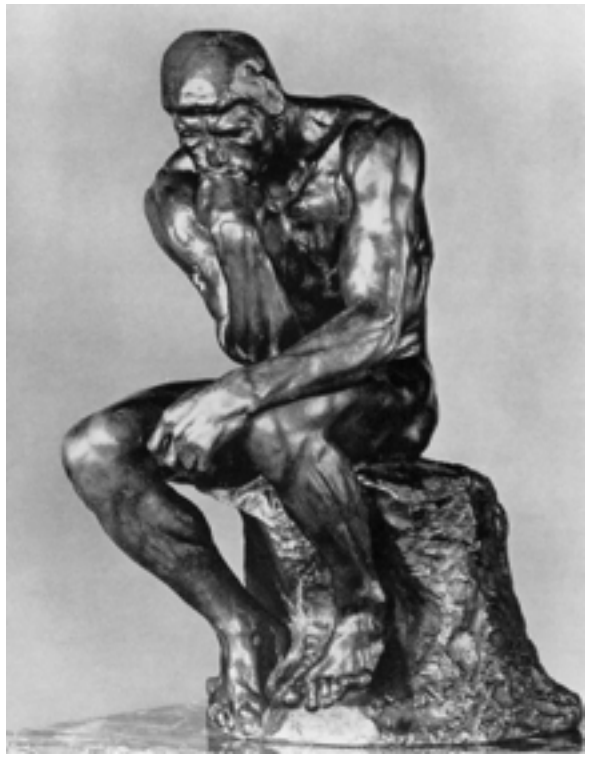
```
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 30);
```

What?
Inconceivable

```
Assertion failed: (score_large_straight((int[5]){1,2,3,4,5}) == 30)
```

```
int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}
```

hmm...



yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

Fail - **Fix** - Pass

There is a bug in the test.

yahtzee_test.c

```
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 30);
```

What?
Inconceivable

```
Assertion failed: (score_large_straight((int[5]){1,2,3,4,5}) == 30)
```

yahtzee.c

```c
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 30);
```

yahtzee.c

```c
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 30);
```

yahtzee.c

```c
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 40);
```

yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee_test.c

```
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 40);
```

yahtzee.c

```c
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee_test.c

```c
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 40);
```

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

Fail - **Fix** - Pass

yahtzee_test.c

```
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 40);
```

Fail - Fix - **Pass**

```
Yahtzee tests OK
```

yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

yahtzee_test.c

```
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 40);
    assert(score_large_straight((int[5]){6,6,6,6,6}) == 0);
    assert(score_large_straight((int[5]){6,5,4,2,3}) == 40);
```

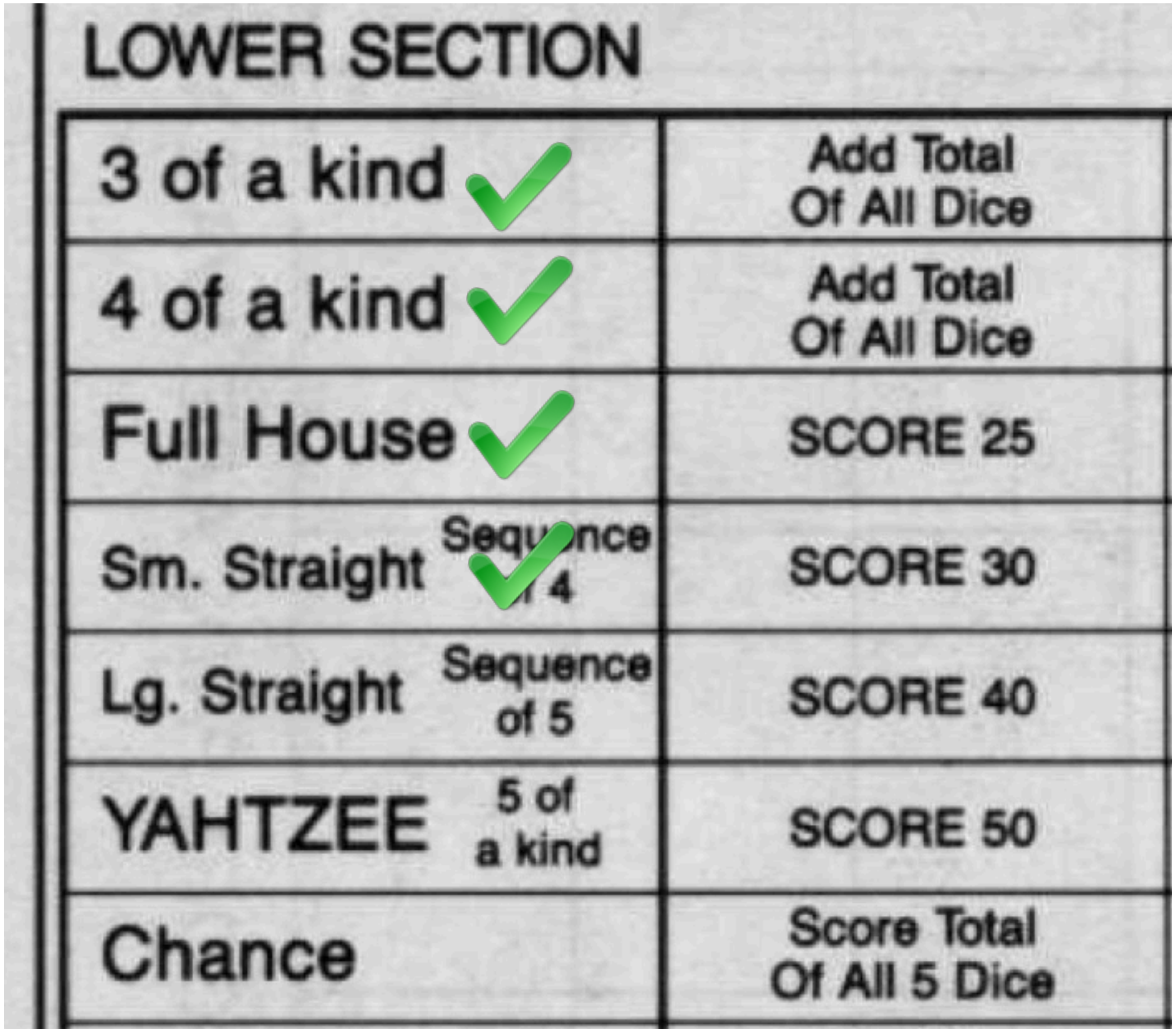
yahtzee.c

```
int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}
```

yahtzee_test.c

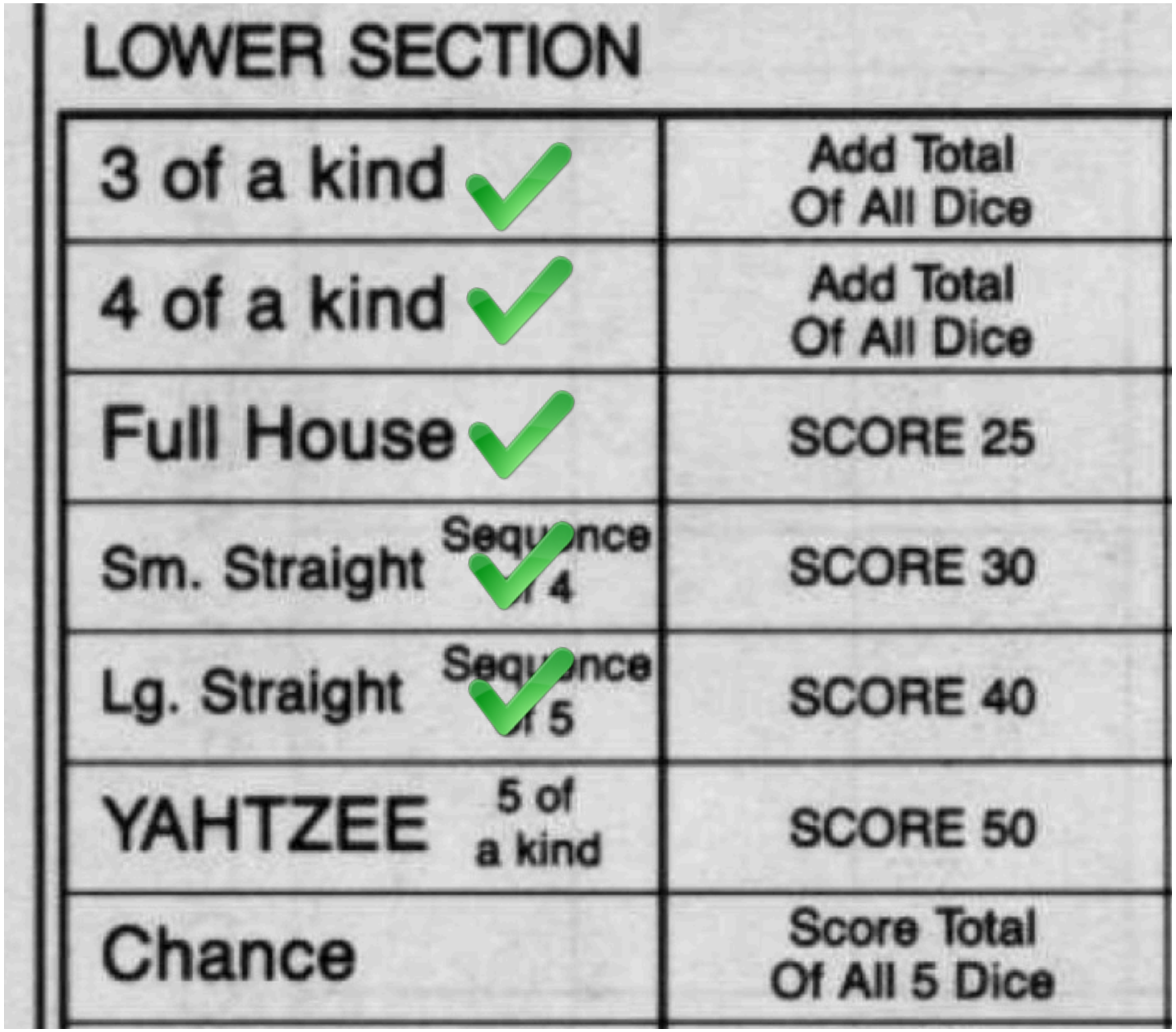
```
    assert(score_large_straight((int[5]){1,2,3,4,5}) == 40);
    assert(score_large_straight((int[5]){6,6,6,6,6}) == 0);
    assert(score_large_straight((int[5]){6,5,4,2,3}) == 40);
```

```
Yahtzee tests OK
```



LOWER SECTION

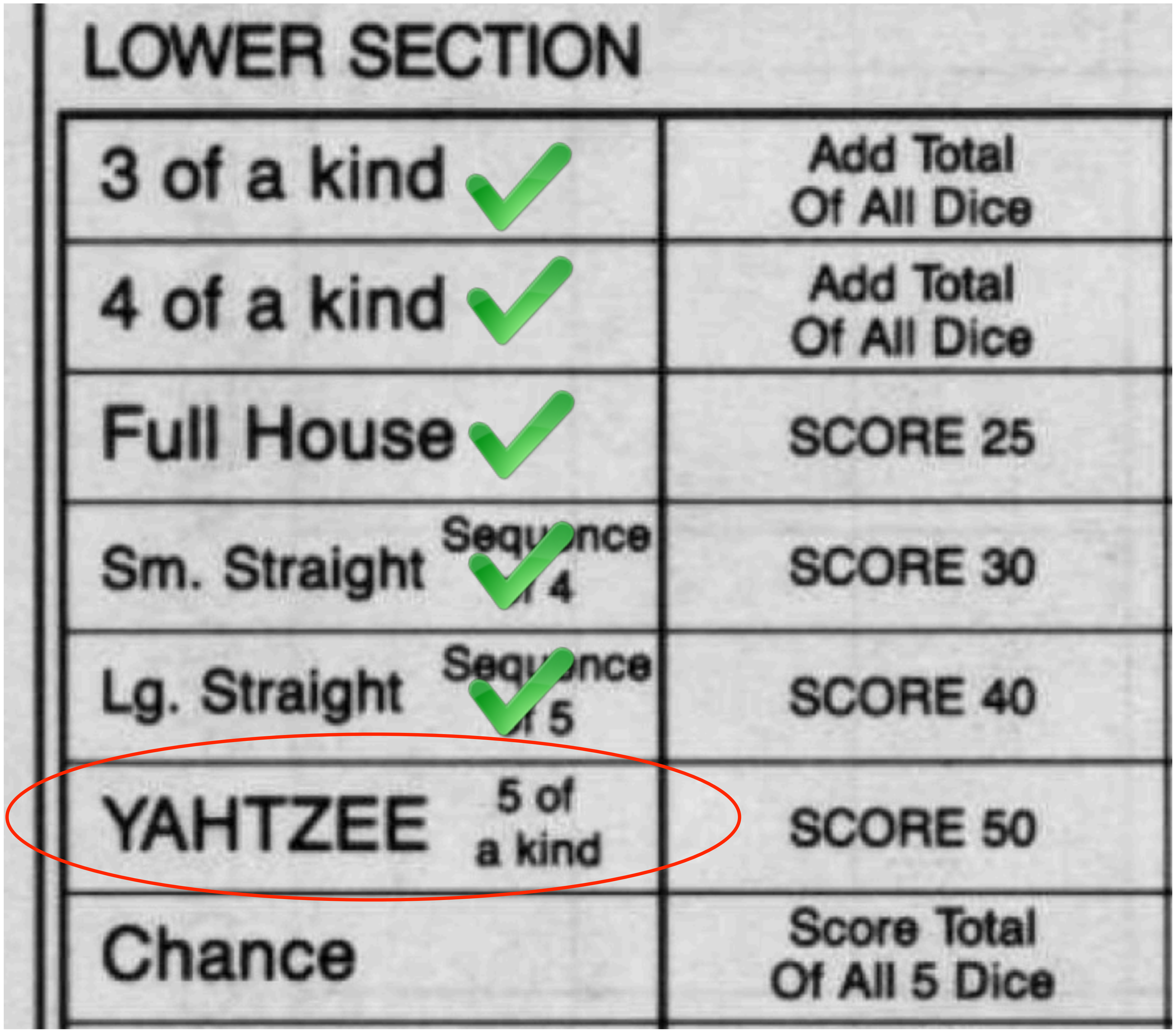
3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4 ✓	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



LOWER SECTION

3 of a kind	✓	Add Total Of All Dice
4 of a kind	✓	Add Total Of All Dice
Full House	✓	SCORE 25
Sm. Straight	Sequence of 4 ✓	SCORE 30
Lg. Straight	Sequence of 5 ✓	SCORE 40
YAHTZEE	5 of a kind	SCORE 50
Chance		Score Total Of All 5 Dice

⚁	⚂	⚃	⚃	⚃
⚃	⚄	⚄	⚄	⚄
⚁	⚁	⚄	⚄	⚄
⚀	⚂	⚃	⚄	⚅
⚀	⚁	⚂	⚃	⚄
⚂	⚂	⚂	⚂	⚂
⚀	⚀	⚂	⚂	⚄



LOWER SECTION

3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4 ✓	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5 ✓	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄

yahtzee_test.c

```
    assert(score_yahtzee((int[5]){1,1,1,1,1}) == 50);
    assert(score_yahtzee((int[5]){6,6,6,6,6}) == 50);
    assert(score_yahtzee((int[5]){1,1,4,1,1}) == 0);
```

yahtzee.c

```c




int score_yahtzee(const int dice[5])
{
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_yahtzee((int[5]){1,1,1,1,1}) == 50);
    assert(score_yahtzee((int[5]){6,6,6,6,6}) == 50);
    assert(score_yahtzee((int[5]){1,1,4,1,1}) == 0);
```

yahtzee.c

```c
int score_yahtzee(const int dice[5])
{
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_yahtzee((int[5]){1,1,1,1,1}) == 50);
    assert(score_yahtzee((int[5]){6,6,6,6,6}) == 50);
    assert(score_yahtzee((int[5]){1,1,4,1,1}) == 0);
```

```
Assertion failed: (score_yahtzee((int[5]){1,1,1,1,1}) == 50)
```

yahtzee.c

```c
int score_yahtzee(const int dice[5])
{
    return 0;
}
```

yahtzee_test.c

```c
    assert(score_yahtzee((int[5]){1,1,1,1,1}) == 50);
    assert(score_yahtzee((int[5]){6,6,6,6,6}) == 50);
    assert(score_yahtzee((int[5]){1,1,4,1,1}) == 0);
```

Fail - Fix - Pass

```
Assertion failed: (score_yahtzee((int[5]){1,1,1,1,1}) == 50)
```

yahtzee.c

```
int score_yahtzee(const int dice[5])
{
    return 0;
}
```

yahtzee_test.c

```
    assert(score_yahtzee((int[5]){1,1,1,1,1}) == 50);
    assert(score_yahtzee((int[5]){6,6,6,6,6}) == 50);
    assert(score_yahtzee((int[5]){1,1,4,1,1}) == 0);
```

Fail - Fix - Pass

```
Assertion failed: (score_yahtzee((int[5]){1,1,1,1,1}) == 50)
```

yahtzee.c

```c




int score_yahtzee(const int dice[5])
{
    return have_exactly_n_of_a_kind(5,dice) ? 50 : 0;
}
```

yahtzee_test.c

```c
    assert(score_yahtzee((int[5]){1,1,1,1,1}) == 50);
    assert(score_yahtzee((int[5]){6,6,6,6,6}) == 50);
    assert(score_yahtzee((int[5]){1,1,4,1,1}) == 0);
```

yahtzee.c

```
int score_yahtzee(const int dice[5])
{
    return have_exactly_n_of_a_kind(5,dice) ? 50 : 0;
}
```

Fail - **Fix** - Pass

yahtzee_test.c

```
    assert(score_yahtzee((int[5]){1,1,1,1,1}) == 50);
    assert(score_yahtzee((int[5]){6,6,6,6,6}) == 50);
    assert(score_yahtzee((int[5]){1,1,4,1,1}) == 0);
```

yahtzee.c

```c
int score_yahtzee(const int dice[5])
{
    return have_exactly_n_of_a_kind(5,dice) ? 50 : 0;
}
```

Fail - **Fix** - Pass

yahtzee_test.c

```c
    assert(score_yahtzee((int[5]){1,1,1,1,1}) == 50);
    assert(score_yahtzee((int[5]){6,6,6,6,6}) == 50);
    assert(score_yahtzee((int[5]){1,1,4,1,1}) == 0);
```

```
Yahtzee tests OK
```

yahtzee.c

```
int score_yahtzee(const int dice[5])
{
    return have_exactly_n_of_a_kind(5,dice) ? 50 : 0;
}
```

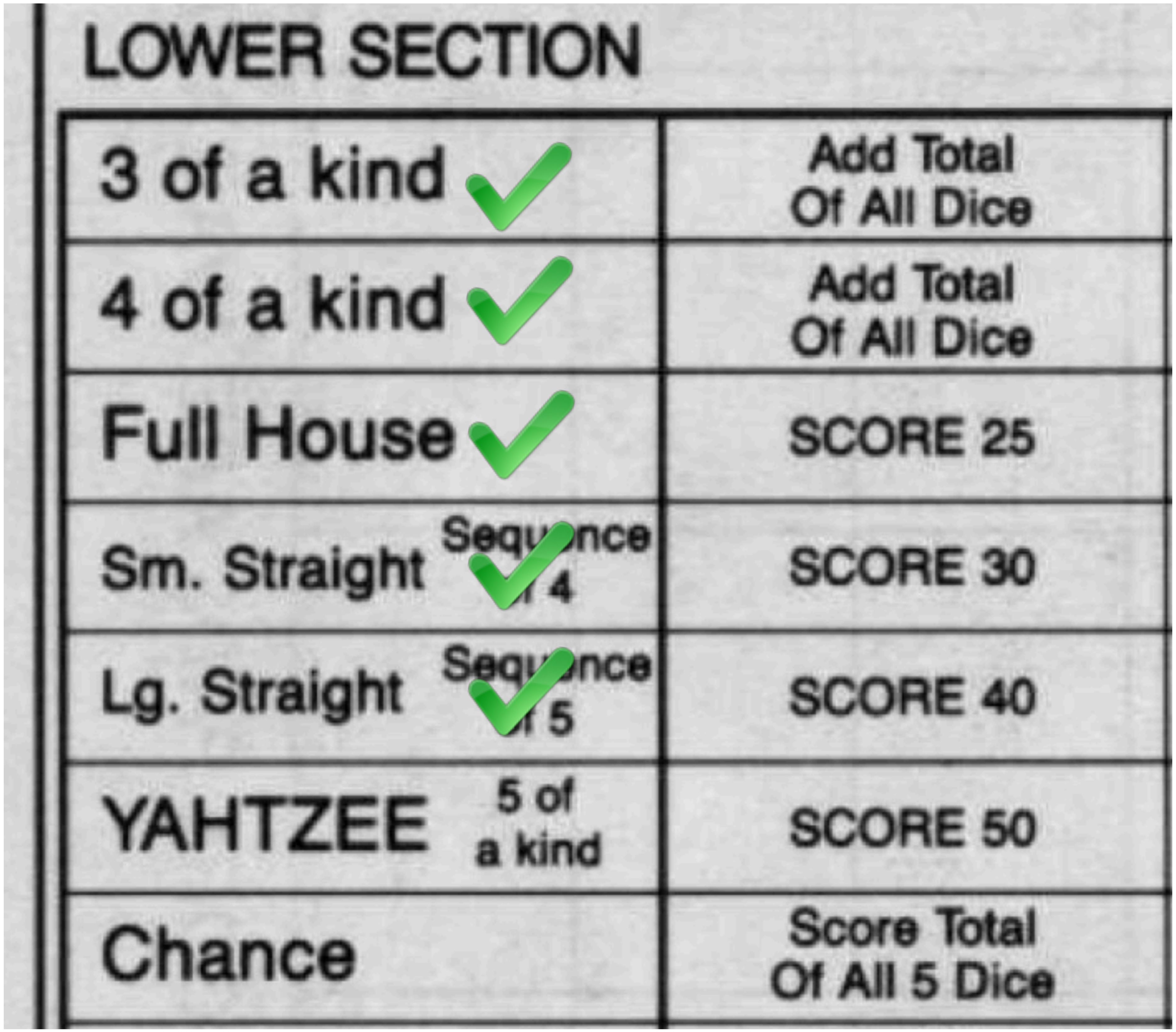
Fail - **Fix** - Pass

yahtzee_test.c

```
    assert(score_yahtzee((int[5]){1,1,1,1,1}) == 50);
    assert(score_yahtzee((int[5]){6,6,6,6,6}) == 50);
    assert(score_yahtzee((int[5]){1,1,4,1,1}) == 0);
```

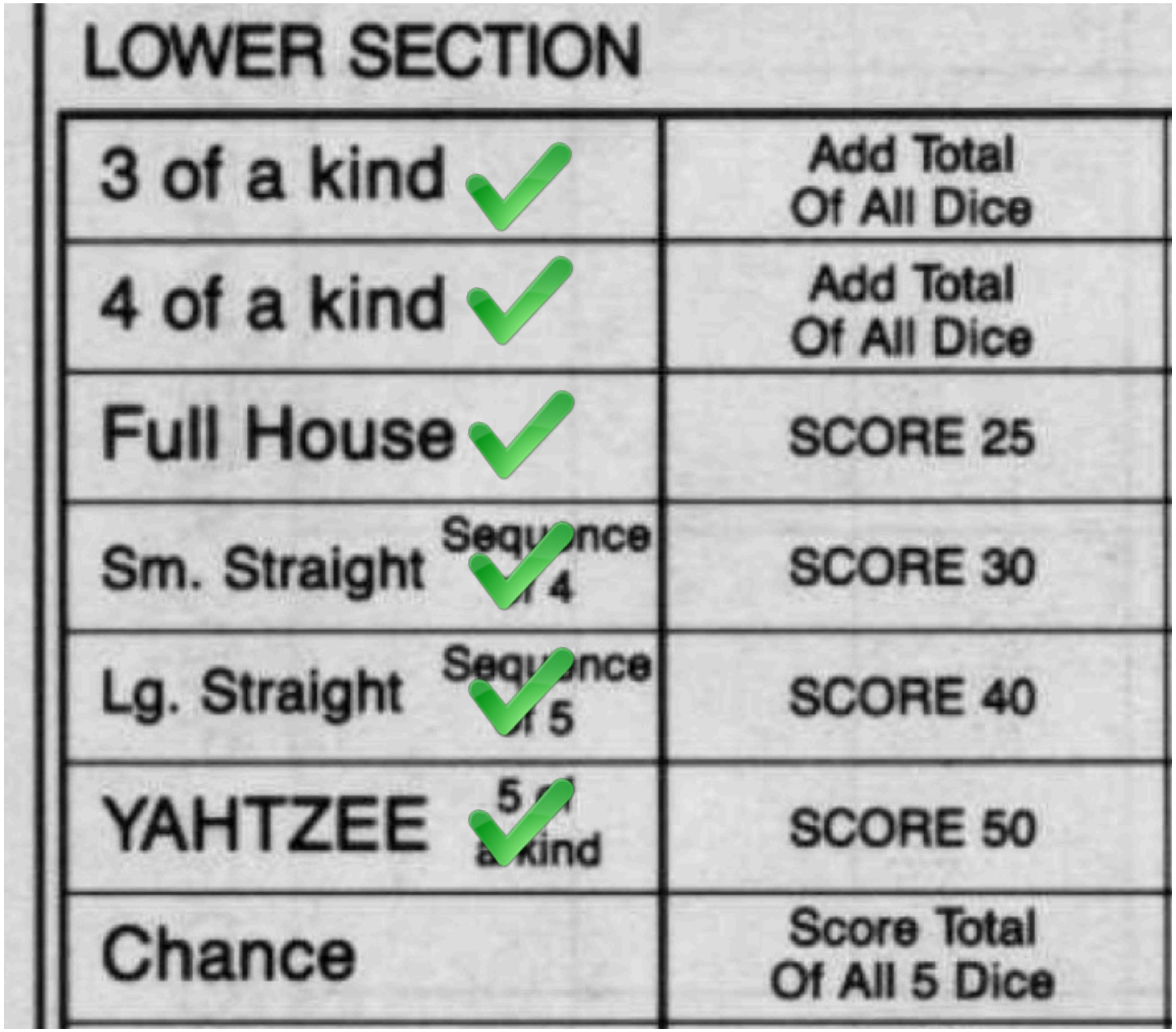
Fail - Fix - **Pass**

```
Yahtzee tests OK
```



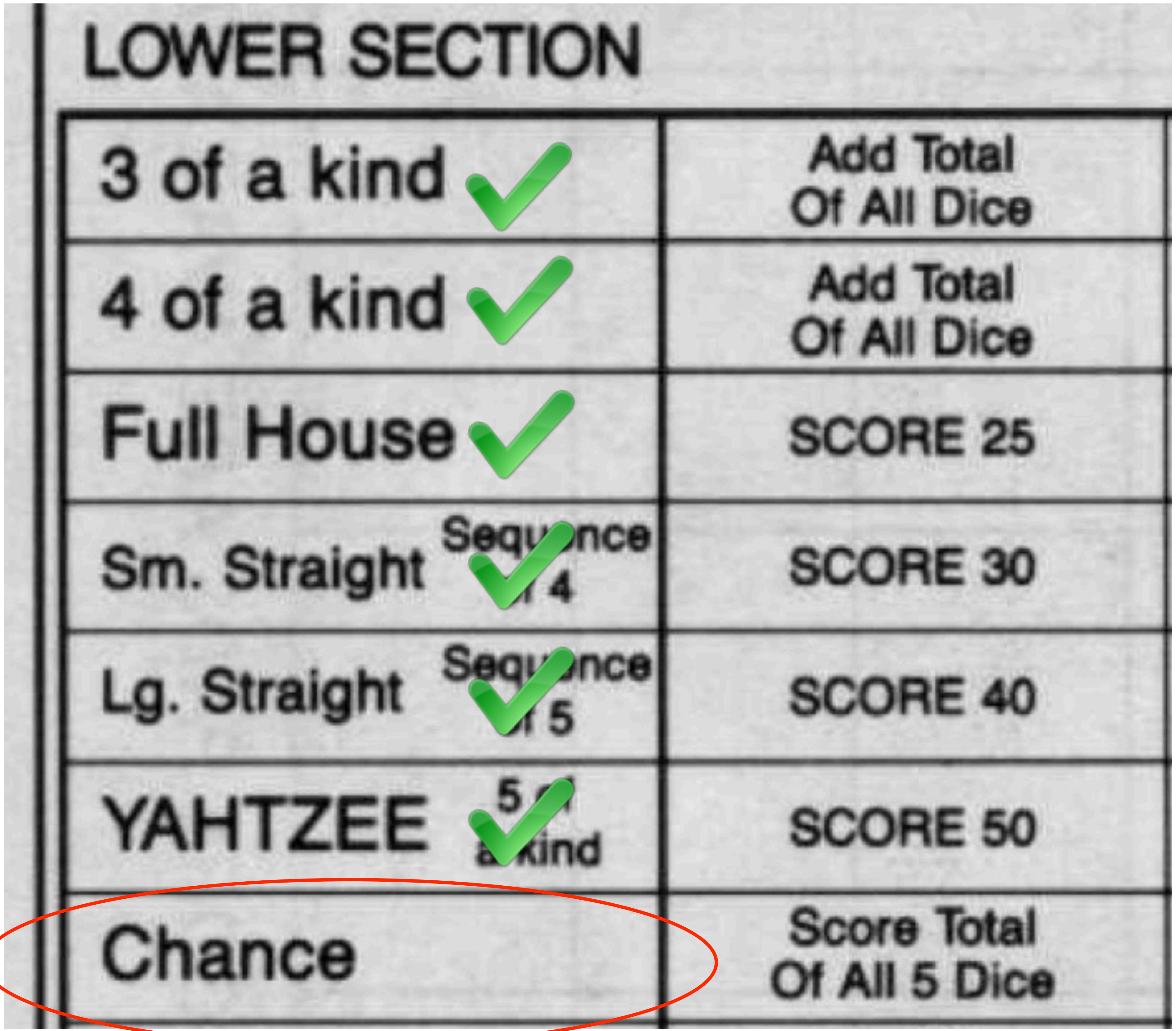
LOWER SECTION

3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight Sequence of 4 ✓	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight Sequence of 5 ✓	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



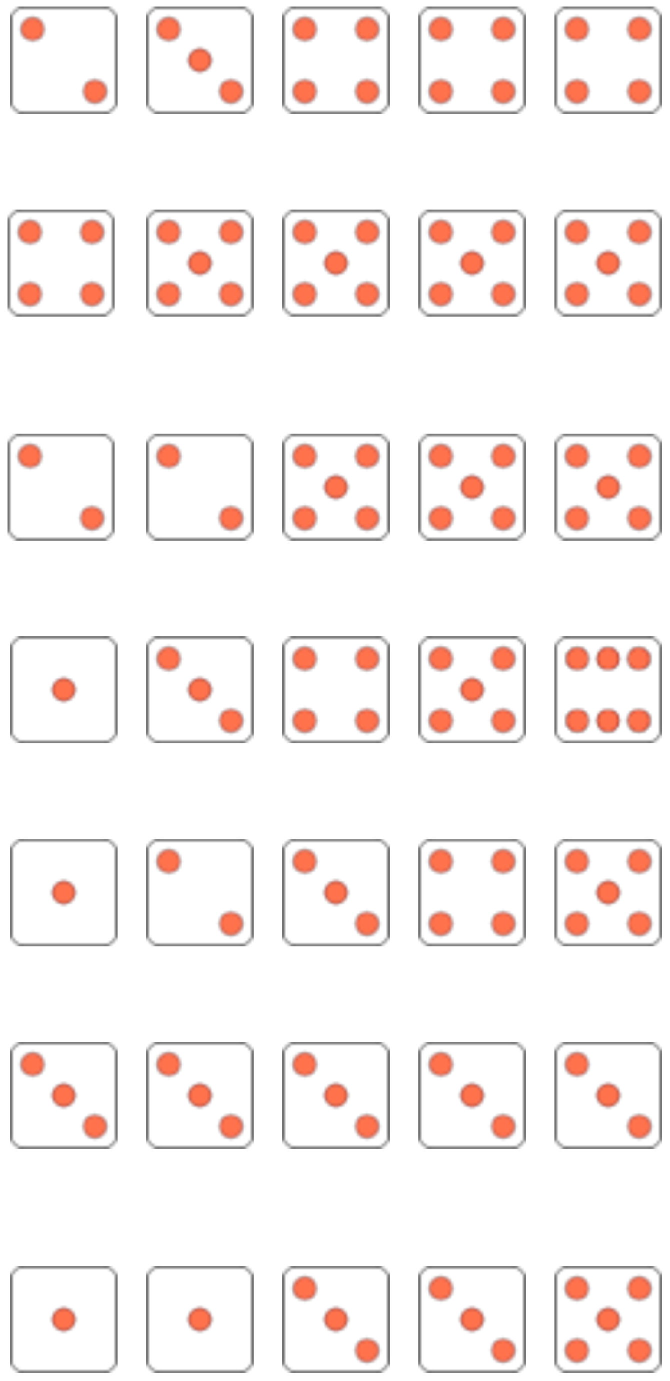
LOWER SECTION

3 of a kind ✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind ✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House ✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight ✓ Sequence of 4	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight ✓ Sequence of 5	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE ✓ 5 of a kind	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



LOWER SECTION

3 of a kind ✓	Add Total Of All Dice
4 of a kind ✓	Add Total Of All Dice
Full House ✓	SCORE 25
Sm. Straight Sequence of 4 ✓	SCORE 30
Lg. Straight Sequence of 5 ✓	SCORE 40
YAHTZEE 5 of a kind ✓	SCORE 50
Chance	Score Total Of All 5 Dice



yahtzee.c

```c
int score_chance(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.c

```
int score_chance(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
int score_small_straight(const int dice[5]);
int score_large_straight(const int dice[5]);
int score_yahtzee(const int dice[5]);
```

yahtzee.c

```
int score_chance(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
int score_small_straight(const int dice[5]);
int score_large_straight(const int dice[5]);
int score_yahtzee(const int dice[5]);

```

yahtzee.c

```c
int score_chance(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
int score_small_straight(const int dice[5]);
int score_large_straight(const int dice[5]);
int score_yahtzee(const int dice[5]);
int score_chance(const int dice[5]);
```

yahtzee.c

```c
int score_chance(const int dice[5])
{
    return sum_of_dice(dice);
}
```

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
int score_small_straight(const int dice[5]);
int score_large_straight(const int dice[5]);
int score_yahtzee(const int dice[5]);
int score_chance(const int dice[5]);
```

yahtzee_test.c

```c
    assert(score_chance((int[5]){1,1,4,1,1}) == 8);
    assert(score_chance((int[5]){2,3,4,5,6}) == 20);
    assert(score_chance((int[5]){6,6,6,6,6}) == 30);
```

yahtzee.c

```
int score_chance(const int dice[5])
{
    return sum_of_dice(dice);
}
```

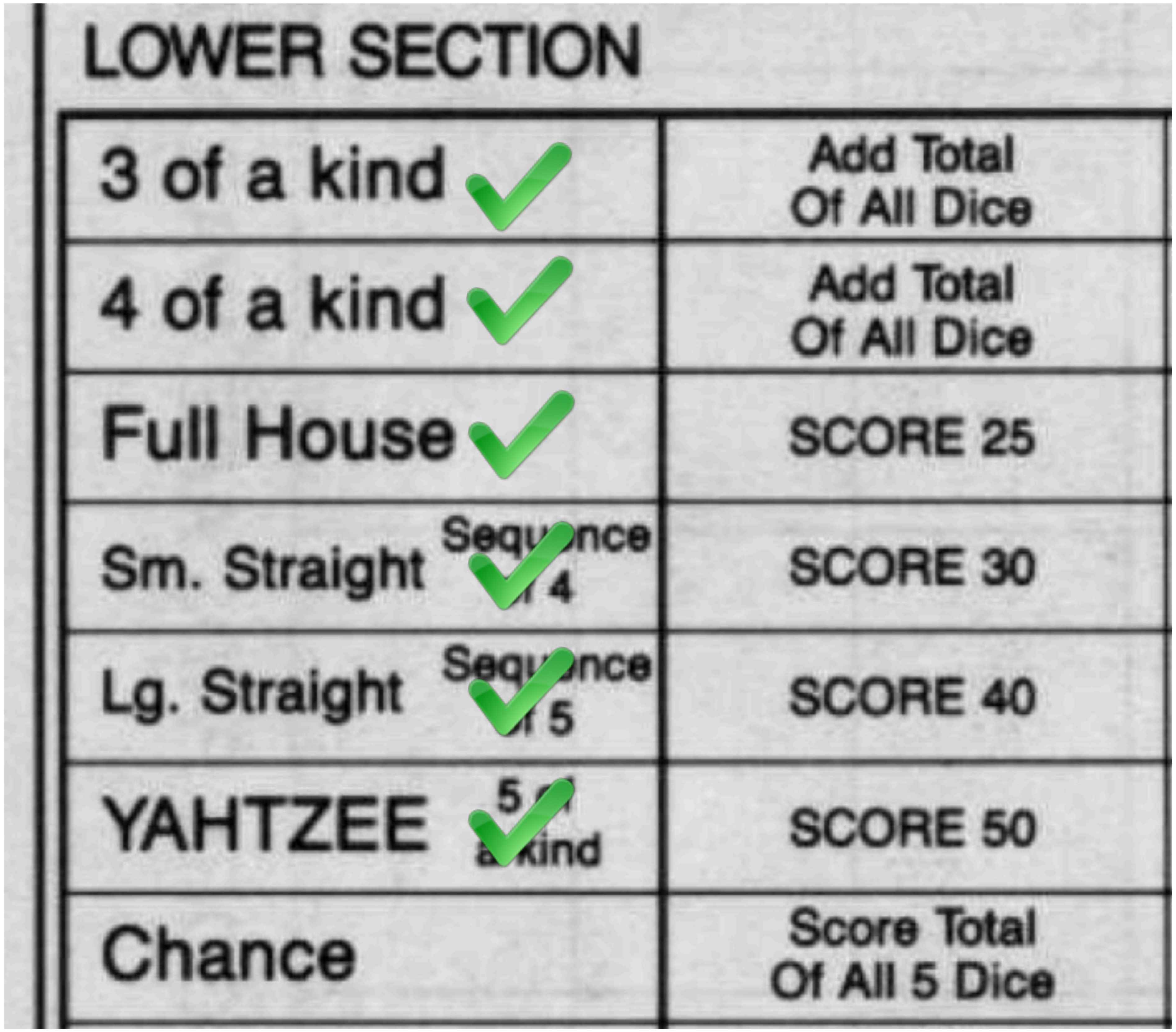
yahtzee.h

```
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
int score_small_straight(const int dice[5]);
int score_large_straight(const int dice[5]);
int score_yahtzee(const int dice[5]);
int score_chance(const int dice[5]);
```

yahtzee_test.c

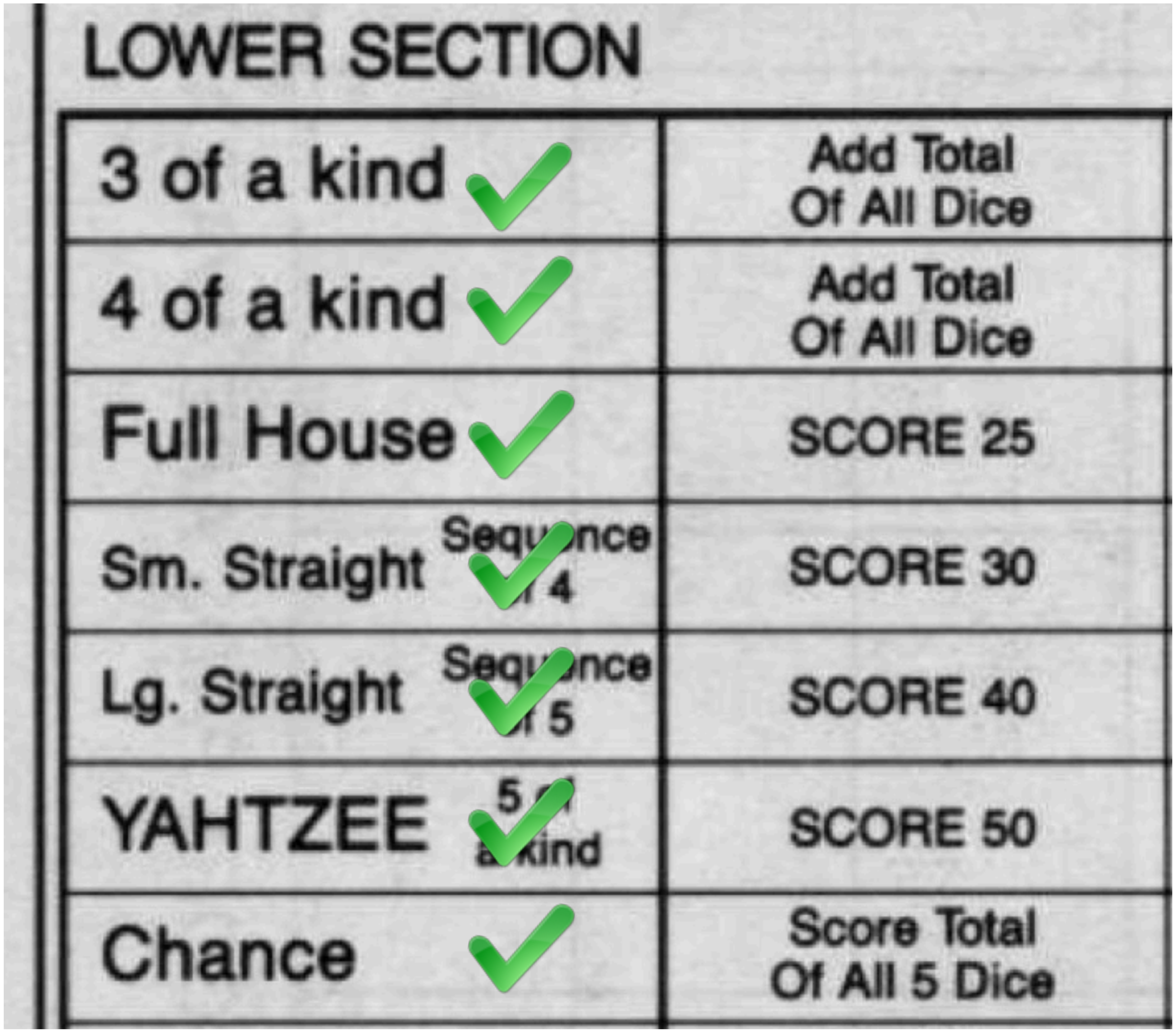
```
    assert(score_chance((int[5]){1,1,4,1,1}) == 8);
    assert(score_chance((int[5]){2,3,4,5,6}) == 20);
    assert(score_chance((int[5]){6,6,6,6,6}) == 30);
```

```
Yahtzee tests OK
```



LOWER SECTION

3 of a kind	✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind	✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House	✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight	Sequence of 4 ✓	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight	Sequence of 5 ✓	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE	5 of a kind ✓	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance		Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄



LOWER SECTION

3 of a kind	✓	Add Total Of All Dice	⚁ ⚂ ⚃ ⚃ ⚃
4 of a kind	✓	Add Total Of All Dice	⚃ ⚄ ⚄ ⚄ ⚄
Full House	✓	SCORE 25	⚁ ⚁ ⚄ ⚄ ⚄
Sm. Straight	Sequence of 4 ✓	SCORE 30	⚀ ⚂ ⚃ ⚄ ⚅
Lg. Straight	Sequence of 5 ✓	SCORE 40	⚀ ⚁ ⚂ ⚃ ⚄
YAHTZEE	5 of a kind ✓	SCORE 50	⚂ ⚂ ⚂ ⚂ ⚂
Chance	✓	Score Total Of All 5 Dice	⚀ ⚀ ⚂ ⚂ ⚄

yahtzee.h

```c
int score_three_of_a_kind(const int dice[5]);
int score_four_of_a_kind(const int dice[5]);
int score_full_house(const int dice[5]);
int score_small_straight(const int dice[5]);
int score_large_straight(const int dice[5]);
int score_yahtzee(const int dice[5]);
int score_chance(const int dice[5]);
```

Makefile

```make
CC=gcc
CFLAGS=-std=c99 -O -Wall -Wextra -pedantic
LD=gcc

all: yahtzee.a

yahtzee.a: yahtzee.o
	ar -rcs $@ $^

check: yahtzee_tests
	./yahtzee_tests

yahtzee.o: yahtzee.c yahtzee.h

yahtzee_tests.o: yahtzee_tests.c yahtzee.h

yahtzee_tests: yahtzee_tests.o yahtzee.a
	$(LD) -o $@ $^

clean:
	rm -f *.o *.a yahtzee_tests
```

yahtzee_tests.h

```c
#include "yahtzee.h"
#include <stdio.h>
#include <assert.h>

int main(void)
{
    assert(score_three_of_a_kind((int[5]){1,1,1,2,2}) == 3+2+2);
    assert(score_three_of_a_kind((int[5]){1,1,1,6,4}) == 3+10);
    assert(score_three_of_a_kind((int[5]){1,1,6,1,4}) == 3+10);
    assert(score_three_of_a_kind((int[5]){1,6,1,2,4}) == 0);
    assert(score_three_of_a_kind((int[5]){6,6,6,6,6}) == 30);

    assert(score_four_of_a_kind((int[5]){1,1,1,1,5}) == 9);
    assert(score_four_of_a_kind((int[5]){1,1,3,1,5}) == 0);
    assert(score_four_of_a_kind((int[5]){1,1,1,1,1}) == 5);

    assert(score_full_house((int[5]){3,3,3,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,5,5,5}) == 25);
    assert(score_full_house((int[5]){3,3,1,5,5}) == 0);

    assert(score_small_straight((int[5]){1,2,3,4,6}) == 30);
    assert(score_small_straight((int[5]){2,3,4,5,6}) == 30);
    assert(score_small_straight((int[5]){2,3,1,5,6}) == 0);
    assert(score_small_straight((int[5]){6,5,4,3,3}) == 30);

    assert(score_large_straight((int[5]){1,2,3,4,5}) == 40);
    assert(score_large_straight((int[5]){6,6,6,6,6}) == 0);
    assert(score_large_straight((int[5]){6,5,4,2,3}) == 40);

    assert(score_yahtzee((int[5]){1,1,1,1,1}) == 50);
    assert(score_yahtzee((int[5]){6,6,6,6,6}) == 50);
    assert(score_yahtzee((int[5]){1,1,4,1,1}) == 0);

    assert(score_chance((int[5]){1,1,4,1,1}) == 8);
    assert(score_chance((int[5]){2,3,4,5,6}) == 20);
    assert(score_chance((int[5]){6,6,6,6,6}) == 30);

    puts("Yahtzee tests OK");
}
```

yahtzee.c

```c
#include "yahtzee.h"
#include <stdbool.h>
#include <stddef.h>

static int sum_of_dice(const int dice[5])
{
    int sum = 0;
    for (size_t die = 0; die < 5; die++)
        sum += dice[die];
    return sum;
}

static int count_face(int face, const int dice[5])
{
    int count = 0;
    for (size_t die = 0; die < 5; die++)
        if (dice[die] == face)
            count++;
    return count;
}

static bool have_at_least_n_of_a_kind(int n, const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) >= n)
            return true;
    return false;
}

static bool have_exactly_n_of_a_kind(int n, const int dice[5])
{
    for (int face = 1; face <= 6; face++)
        if (count_face(face,dice) == n)
            return true;
    return false;
}

int have_at_least_n_in_a_row(int n, const int dice[5])
{
    int max_seq_len = 0;
    int seq_len = 0;
    for (int face = 1; face <= 6; face++) {
        if (count_face(face,dice) == 0)
            seq_len = 0;
        else
            seq_len++;
        if (seq_len > max_seq_len)
            max_seq_len = seq_len;
    }
    return max_seq_len >= n;
}
```

```c
int score_three_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(3, dice))
        return sum_of_dice(dice);
    return 0;
}

int score_four_of_a_kind(const int dice[5])
{
    if (have_at_least_n_of_a_kind(4, dice))
        return sum_of_dice(dice);
    return 0;
}

int score_full_house(const int dice[5])
{
    if (have_exactly_n_of_a_kind(3, dice) &&
        have_exactly_n_of_a_kind(2, dice))
        return 25;
    return 0;
}

int score_small_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(4,dice))
        return 30;
    return 0;
}

int score_large_straight(const int dice[5])
{
    if (have_at_least_n_in_a_row(5,dice))
        return 40;
    return 0;
}

int score_yahtzee(const int dice[5])
{
    if (have_exactly_n_of_a_kind(5, dice))
        return 50;
    return 0;
}

int score_chance(const int dice[5])
{
    return sum_of_dice(dice);
}
```

!