TDD in C

... or The Bowling Game Kata with C and assert()

Olve Maudal , oma@pvv.org

(a 10 minute lightening-talk @ Smidig 2007 Oslo, 26-27 November)

Bowling Game Kata

Bowling Game Kata in C

The following is a demonstration of how you can do test-driven development in plain C, without any bloated test framework. We are going to write some code for scoring a game of bowling.

Since the seminal article "Engineer Notebook: An Extreme Programming Episode" published in 2001 by Robert C. Martin and Robert S. Koss:

- http://www.objectmentor.com/resources/articles/xpepisode.htm

calculating the score for a bowling game has gained status as an advanced "Hello World" for programming languages. For any programming language out there you will find a bowling score implementation inspired by the "XP Episode". There is also a lot of derivative work from this article, some of them demonstrating how design evolves through Test-Driven Development.

What you will see now is taken more or less directly out of the excellent “Bowling Game Kata” presentation by Robert C. Martin.

- http://butunclebob.com
- http://butunclebob.com/ArticleS.UncleBob.TheBowlingGameKata
- http://butunclebob.com/files/downloads/Bowling%20Game%20Kata.ppt

Basically the only thing I have done is to translate from Java/JUnit into C/assert()

Since Uncle Bob is a nice guy...

... we include this page, because he asked us to do so:

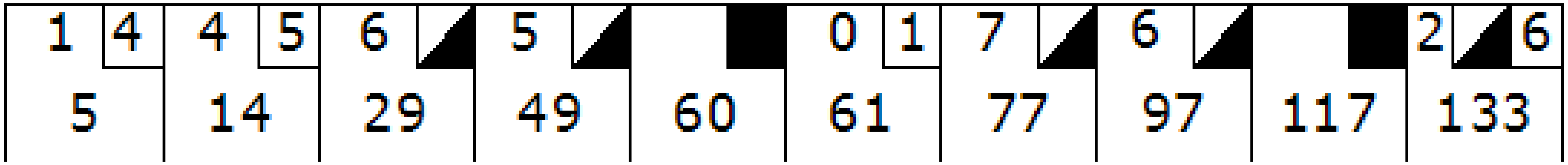
Bowling Game Kata

Object Mentor, Inc.
www.objectmentor.com
blog.objectmentor.com

fitnesse.org
XProgramming.com
JU.org
www.junit.org

The following slides are not verbatim copies, but they are close enough to deserve a proper copyright notice...

Scoring Bowling



1 4	4 5	6 ◢	5 ◢	■	0 1	7 ◢	6 ◢	■	2 ◢ 6
5	14	29	49	60	61	77	97	117	133

The game consists of 10 frames as shown above. In each frame the player has two opportunities to knock down 10 pins. The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares.

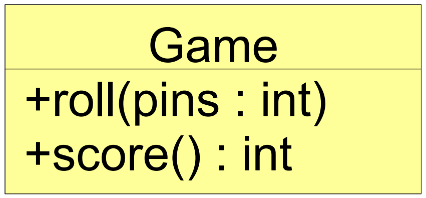
A spare is when the player knocks down all 10 pins in two tries. The bonus for that frame is the number of pins knocked down by the next roll. So in frame 3 above, the score is 10 (the total number knocked down) plus a bonus of 5 (the number of pins knocked down on the next roll.)

A strike is when the player knocks down all 10 pins on his first try. The bonus for that frame is the value of the next two balls rolled.

In the tenth frame a player who rolls a spare or strike is allowed to roll the extra balls to complete the frame. However no more than three balls can be rolled in tenth frame.

[source: Uncle Bob]

The Requirements.

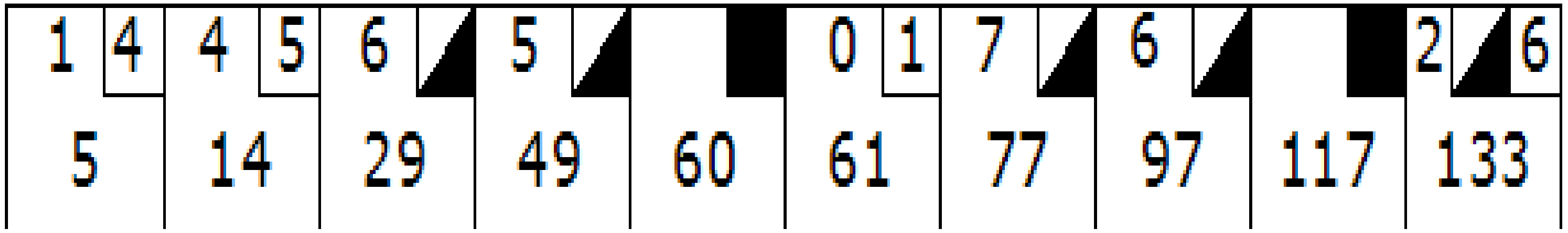


Write a class named "Game" that has two methods:

- roll(pins : int) is called each time the player rolls a ball. The argument is the number of pins knocked down.
- score() : int is called only at the very end of the game. It returns the total score for that game.

[source: Uncle Bob]

Scoring Bowling & The Requirements

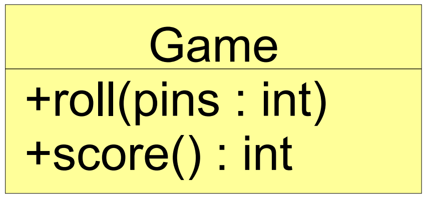


The **game** consists of 10 **frames** as shown above. In each **frame** the **player** has two opportunities to knock down 10 **pins**. The **score** for the **frame** is the total number of **pins** knocked down, plus **bonuses** for **strikes** and **spares**.

A **spare** is when the **player** knocks down all 10 **pins** in **two tries**. The **bonus** for that **frame** is the number of **pins** knocked down by the next **roll**. So in **frame** 3 above, the **score** is 10 (the total number knocked down) plus a **bonus** of 5 (the number of pins knocked down on the next roll.)

A **strike** is when the **player** knocks down all 10 **pins** on his first try. The **bonus** for that **frame** is the value of the next two **balls rolled**.

In the **tenth frame** a **player** who **rolls** a **spare** or **strike** is allowed to **roll** the **extra balls** to **complete the frame**. However no more than three **balls** can be rolled in **tenth frame**.

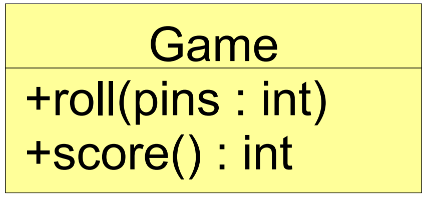


Write a class named "Game" that has two methods:

- roll(pins : int) is called each time the player rolls a ball. The argument is the number of pins knocked down.
- score() : int is called only at the very end of the game. It returns the total score for that game.

[source: Uncle Bob]

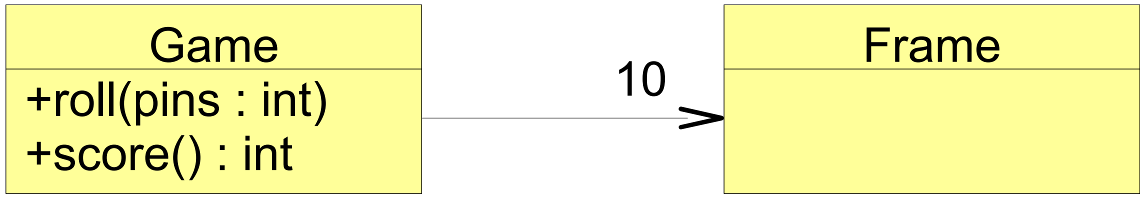
A quick design session



Clearly we need the Game class.

[source: Uncle Bob]

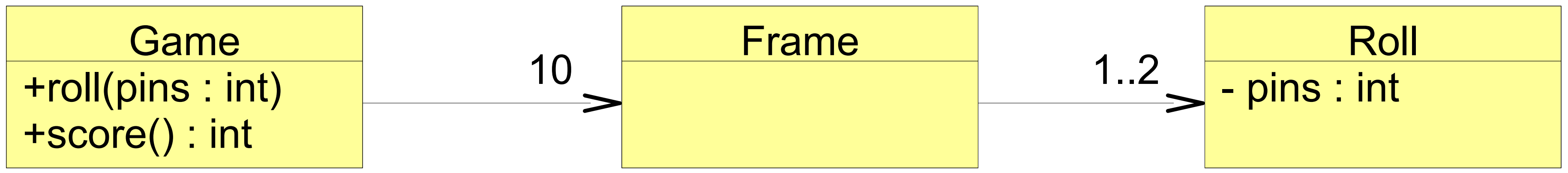
A quick design session



A game has 10 frames.

[source: Uncle Bob]

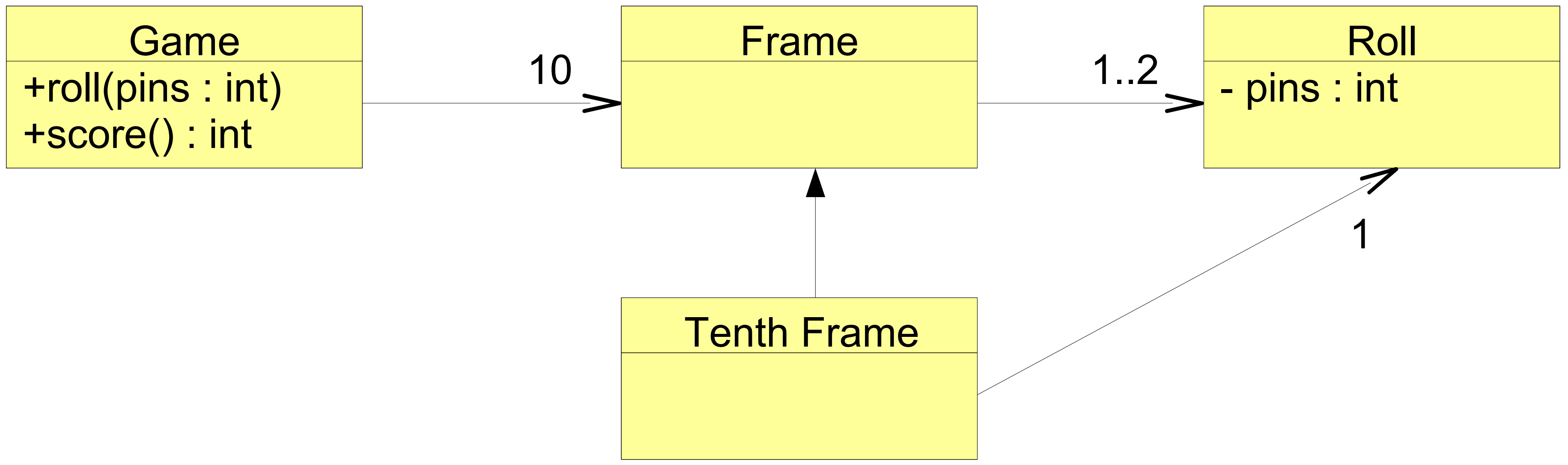
A quick design session



A frame has 1 or two rolls.

[source: Uncle Bob]

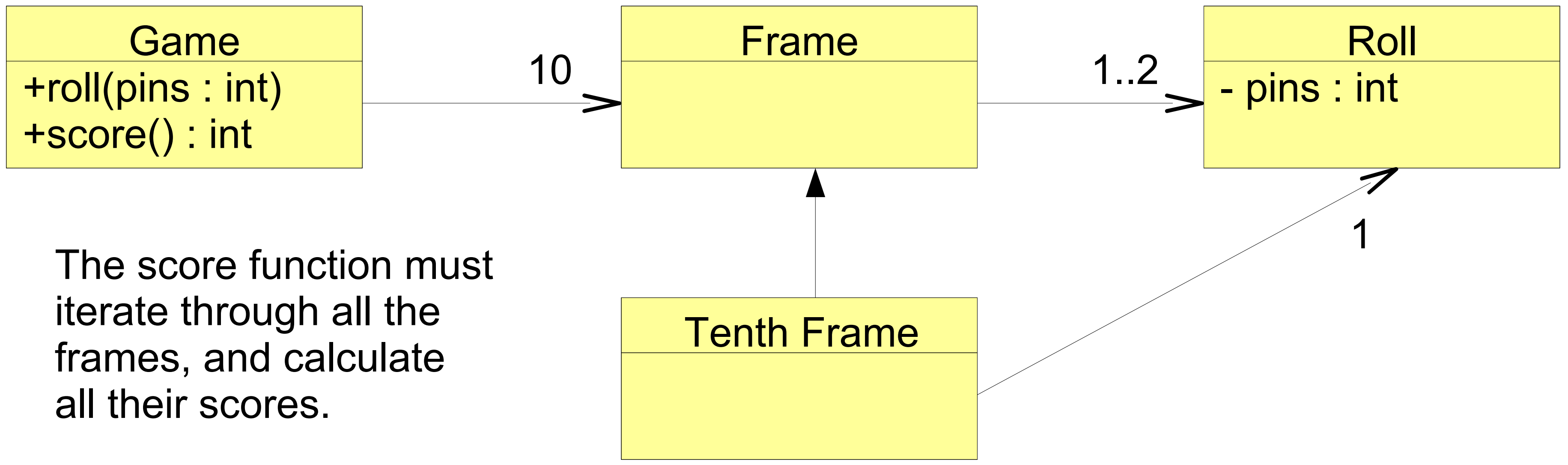
A quick design session



The tenth frame has two or three rolls.
It is different from all the other frames.

[source: Uncle Bob]

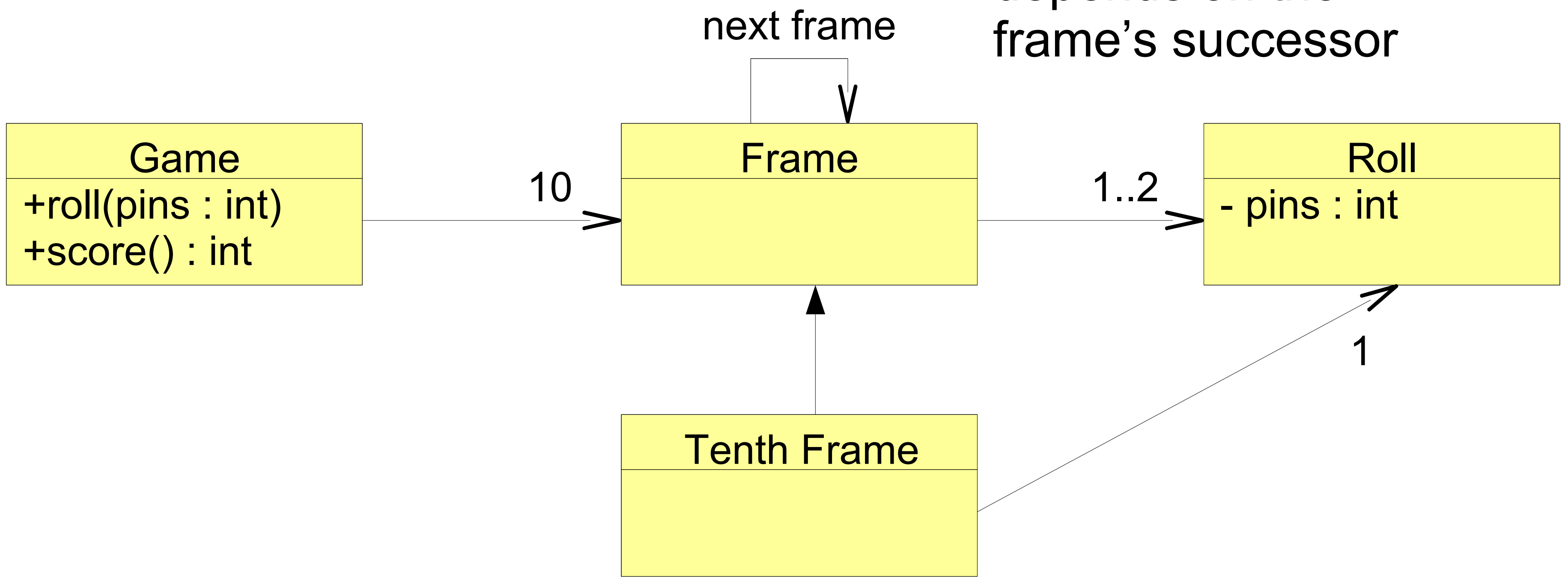
A quick design session



The score function must iterate through all the frames, and calculate all their scores.

[source: Uncle Bob]

A quick design session

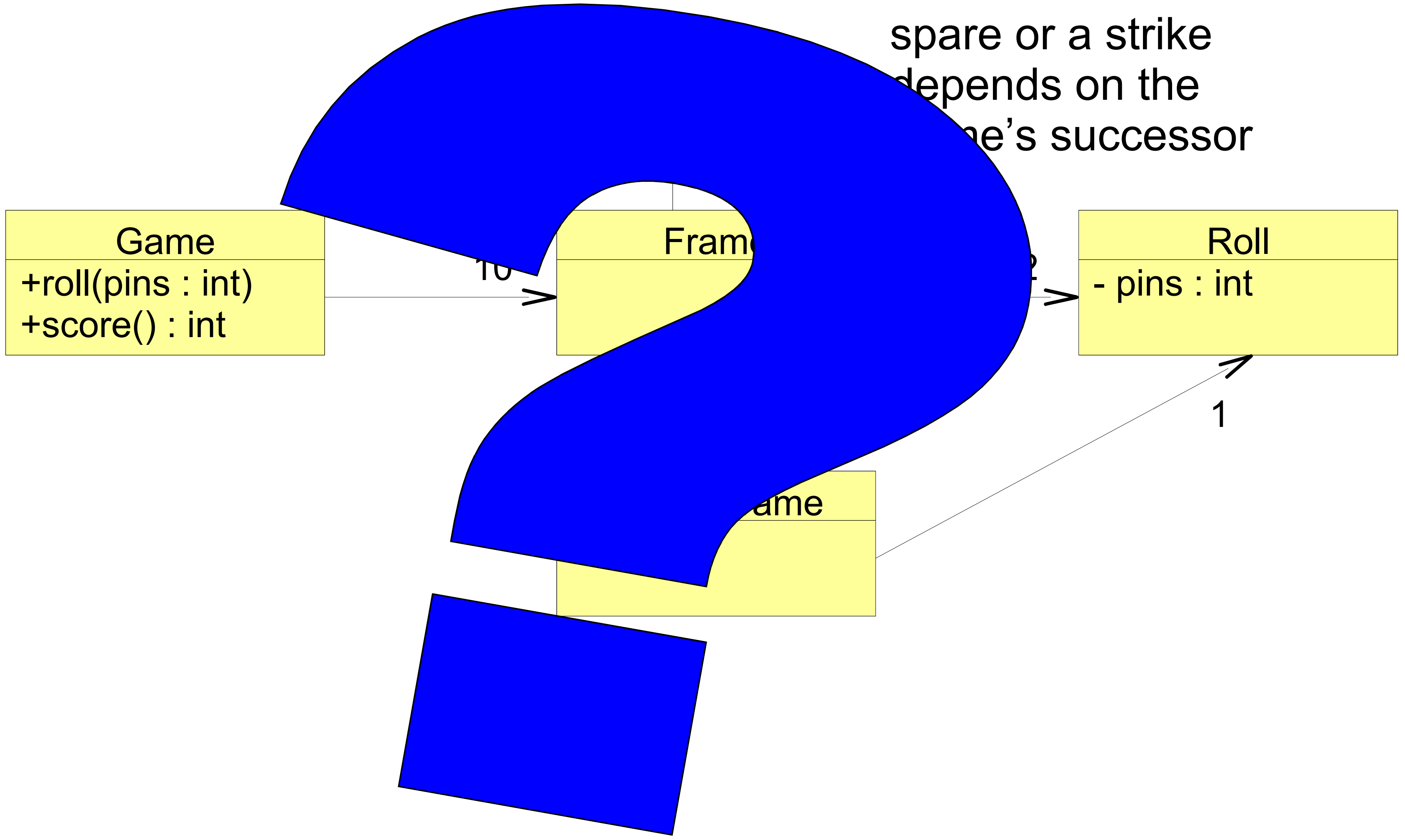


The score for a spare or a strike depends on the frame's successor

[source: Uncle Bob]

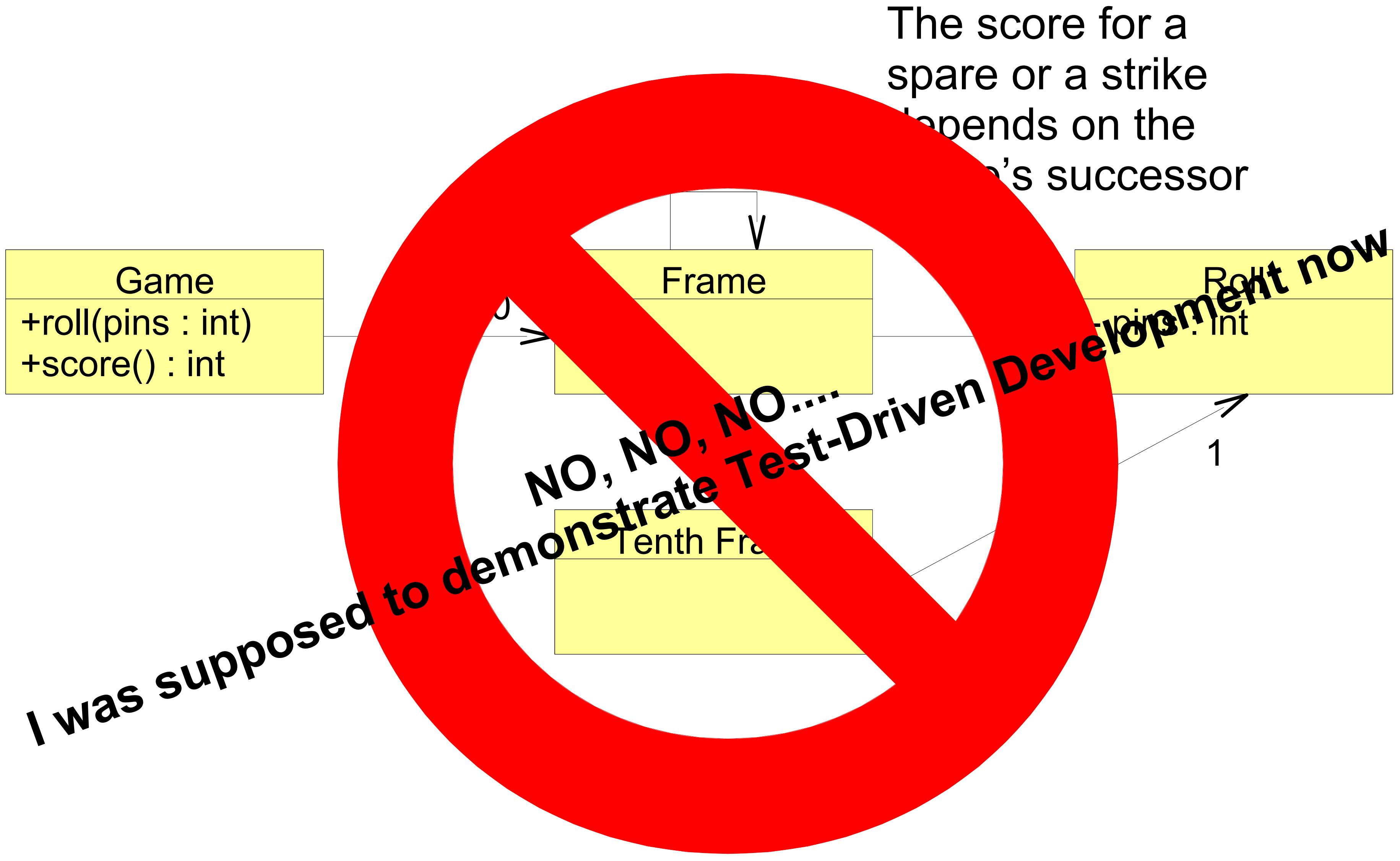
A quick design session

The score for a spare or a strike depends on the ~~frame's~~ successor



[source: Uncle Bob]

A quick design session


The score for a spare or a strike depends on the frame's successor
Game
+roll(pins : int)
+score() : int
Frame
Roll
- pins : int
1
Tenth Frame
NO, NO, NO....
I was supposed to demonstrate Test-Driven Development now

[source: Uncle Bob]

With TDD you are still “allowed” to do a lot of up-front design if you find it useful to understand the problem domain, but you should try to let your tests drive the design process. Doing TDD correctly, you will find that sometimes you end up with a surprisingly simple and solid design. The key idea is that you can evolve the design by telling the system **what** it should do, rather that **how** it should do it.

Getting Started with Test-Driven Development

- create a new directory named bowling
- create a 'bowling_game_test.c' file with a failing test
- execute and see the test fail

```c
// bowling_game_test.c

#include <assert.h>
#include <stdbool.h>

int main() {
    assert( false && "My first unit test" );
}
```

```
mkdir bowling
cd bowling
ed bowling_game_test.c
gcc -std=c99 -Wall bowling_game_test.c && ./a.out
bowling_game_test.c:5: failed assertion `false && "My first unit test"'
```

The First Test

```
test_gutter_game()
```

The First Test

```
// bowling_game_test.c



#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

The First Test

```
// bowling_game_test.c



#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c && ./a.out
/usr/bin/ld: Undefined symbols:
_bowling_game_init
_bowling_game_roll
_bowling_game_score
```

The First Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

The First Test

```
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c && ./a.out
/usr/bin/ld: Undefined symbols:
_bowling_game_init
_bowling_game_roll
_bowling_game_score
```

The First Test

```
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```
// bowling_game.c

#include "bowling_game.h"

void bowling_game_init() {
}

void bowling_game_roll(int pins) {
}

int bowling_game_score() {
    return -1;
}
```

```
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

The First Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

void bowling_game_init() {
}

void bowling_game_roll(int pins) {
}

int bowling_game_score() {
    return -1;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
bowling_game_test.c:12: failed assertion `bowling_game_score() == 0 && "test_gutter_game()"'
```

The First Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

void bowling_game_init() {
}

void bowling_game_roll(int pins) {
}

int bowling_game_score() {
    return -1;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutte    ame();
}
```

Now we have a failing unit test.

```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
bowling_game_test.c:12: failed assertion `bowling_game_score() == 0 && "test_gutter_game()"'
```

The First Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

void bowling_game_init() {
}

void bowling_game_roll(int pins) {
}

int bowling_game_score() {
    return -1;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

The First Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

The First Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
```

The First Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

The Second Test

```
test_all_ones()
```

The Second Test

```
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
}
int bowling_game_score() {
    return score;
}
```

```
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

int main() {
    test_gutter_game();
}
```

The Second Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

The Second Test

```
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
}
int bowling_game_score() {
    return score;
}
```

```
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
bowling_game_test.c:21: failed assertion `bowling_game_score() == 20 && "test_gutter_game()"'
```

The Second Test

```
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

The Second Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
```

The Second Test

```
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

The Second Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_g

static int score;

void bowling_game_i
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

Duplicate Code

Perhaps we need to do some refactoring of test code?

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

The Second Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

The Second Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

static void test_gutter_game() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    for (int i=0; i<20; i++)
        bowling_game_roll(1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

The Second Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

static void test_gutter_game() {
    bowling_game_init();
    roll_many(20,0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    roll_many(20,1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

The Second Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_i
    score = 0;
}
void bowling_game_r
    score += pins;
}
int bowling_game_sc
    return score;
}
```

Duplicate Code Eliminated

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

static void test_gutter_game() {
    bowling_game_init();
    roll_many(20,0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    roll_many(20,1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

The Second Test

```
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

static void test_gutter_game() {
    bowling_game_init();
    roll_many(20,0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    roll_many(20,1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
}
```

The Third Test

```
test_one_spare()
```

The Third Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Third Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
bowling_game_test.c:34: failed assertion `bowling_game_score() == 16 && "test_one_spare()"'
```

The Third Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

tempted to use flag to remember previous roll.

The Third Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
                         _one_spare()" );

                                  );
    test_all_ones();
    test_one_spare();
}
```

roll() calculates score, but name does not imply that..

score() does not calculate score, but name implies that it does.

The Third Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game

static int score;

void bowling_game_
    score = 0;
}
void bowling_g
    score += pins
}
int bowling_game
    return score;
}
```

Design is wrong.
Responsibilities
are misplaced.

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Third Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
bowling_game_test.c:34: failed assertion `bowling_game_score() == 16 && "test_one_spare()"'
```

The Third Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```
// bowling_game.h
...
```

```
// bowling_game.c

#include "bowling_game.h"

static int score;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```
// bowling_game.h
...
```

```
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    score = 0;
}
void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```c
// bowling_game.h
...
```

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    score += pins;
}
int bowling_game_score() {
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```
// bowling_game.h
...
```

```
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    return score;
}
```

```
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```
// bowling_game.h
...
```

```
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    for (int i=0; i<max_rolls; i++)
        score += rolls[i];
    return score;
}
```

```
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```
// bowling_game.h
...
```

```
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    for (int i=0; i<max_rolls; i++)
        score += rolls[i];
    return score;
}
```

```
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```c
// bowling_game.h
...
```

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    for (int i=0; i<max_rolls; i++)
        score += rolls[i];
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Third Test

```c
// bowling_game.h
...
```

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    for (int i=0; i<max_rolls; i++)
        score += rolls[i];
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
bowling_game_test.c:34: failed assertion `bowling_game_score() == 16 && "test_one_spare()"'
```

The Third Test

```
// bowling_game.h
...
```

```
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    for (int i=0; i<max_rolls; i++)
        if (rolls[i] + rolls[i+1] == 10) {
            // this is a spare
            score += ...
    return score;
}
```

```
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Third Test

```
// bowling_game.h
...
```

```
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls;
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins
    rolls[current_roll++] = pi
}

int bowling_game_score() {
    int score = 0;
    for (int i=0; i<max_rolls; i++)
        if (rolls[i] + rolls[i+1] == 10) {
            // this is a spare
            score += ...
    return score;
}
```

```
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
                                spare

                              e() == 16
                              e()" );
```

This isn't going to work because index might not refer to the first ball of a frame.

Design is still wrong.

Need to walk through the array two balls (one frame) at a time.

The Third Test

```
// bowling_game.h
...
```

```
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    for (int i=0; i<max_rolls; i++)
        score += rolls[i];
    return score;
}
```

```
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Third Test

```c
// bowling_game.h
...
```

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    for (int i=0; i<max_rolls; i++)
        score += rolls[i];
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```c
// bowling_game.h
...
```

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    for (int i=0; i<max_rolls; i++)
        score += rolls[i];
    return score;
}
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    for (int i=0; i<max_rolls; i++)
        score += rolls[i];
    return score;
}
```

```
// bowling_game.h
...
```

```
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        score += rolls[i] + rolls[i+1];
        i += 2;
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        score += rolls[i] + rolls[i+1];
        i += 2;
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        score += rolls[i] + rolls[i+1];
        i += 2;
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    // test_one_spare();
}
```

The Third Test

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        score += rolls[i] + rolls[i+1];
        i += 2;
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Third Test

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        score += rolls[i] + rolls[i+1];
        i += 2;
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
bowling_game_test.c:34: failed assertion `bowling_game_score() == 16 && "test_one_spare()"'
```

The Third Test

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {... }

void bowling_game_roll(int pins) { ... }

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        score += rolls[i] + rolls[i+1];
        i += 2;
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Third Test

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {... }

void bowling_game_roll(int pins) { ... }

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( rolls[i] + rolls[i+1] == 10 ) {
            // spare
            score += 10 + rolls[i+2];
            i += 2;
        } else {
            // normal
            score += rolls[i] + rolls[i+1];
            i += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Third Test

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {... }

void bowling_game_roll(int pins) { ... }

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( rolls[i] + rolls[i+1] == 10 ) {
            // spare
            score += 10 + rolls[i+2];
            i += 2;
        } else {
            // normal
            score += rolls[i] + rolls[i+1];
            i += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Third Test

```c
// bowling_game.c

#include "bowling_game.h"

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {... }

void bowling_game_roll(int pins) { ... }

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( rolls[i] + rolls[i+1] == 10 ) {
            // spare
            score += 10 + rolls[i+2];
            i += 2;
        } else {
            // normal
            score += rolls[i] + rolls[i+1];
            i += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Third Test

```
// bowling_game.c

...

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( rolls[i] + rolls[i+1] == 10 ) { //spare
            score += 10 + rolls[i+2];
            i += 2;
        } else { // normal throw
            score += rolls[i] + rolls[i+1];
            i += 2;
        }
    }
    return score;
}
```

bad name for variable

ugly comment in conditional

The Third Test

```
// bowling_game.c

...

int bowling_game_score() {
    int score = 0;
    int i = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( rolls[i] + rolls[i+1] == 10 ) { //spare
            score += 10 + rolls[i+2];
            i += 2;
        } else { // normal throw
            score += rolls[i] + rolls[i+1];
            i += 2;
        }
    }
    return score;
}
```

The Third Test

```
// bowling_game.c

...

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( rolls[frame_index] + rolls[frame_index+1] == 10 ) {  // spare
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else { // normal throw
            score += rolls[frame_index] + rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

The Third Test

```
// bowling_game.c

...

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( rolls[frame_index] + rolls[frame_index+1] == 10 ) {  // spare
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else { // normal throw
            score += rolls[frame_index] + rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

The Third Test

```
// bowling_game.c

...

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( rolls[frame_index] + rolls[frame_index+1] == 10 ) {  // spare
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else { // normal throw
            score += rolls[frame_index] + rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

ugly comment
in conditional

The Third Test

```
// bowling_game.c

...

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( rolls[frame_index] + rolls[frame_index+1] == 10 ) {  // spare
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else { // normal throw
            score += rolls[frame_index] + rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

The Third Test

```
// bowling_game.c

...

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_spare(frame_index) ) {  // spare
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else { // normal throw
            score += rolls[frame_index] + rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

The Third Test

```
// bowling_game.c

...

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_spare(frame_index) ) {  // spare
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else { // normal throw
            score += rolls[frame_index] + rolls[frame_ind   1];
            frame_index += 2;
        }
    }
    return score;
}
```

ugly comment
in conditional

The Third Test

```
// bowling_game.c

...

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_spare(frame_index) ) {  // spare
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else { // normal throw
            score += rolls[frame_index] + rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

useless comment

The Third Test

```
// bowling_game.c

...

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_spare(frame_index) ) {
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else { // normal throw
            score += rolls[frame_index] + rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

The Third Test

```
// bowling_game.c

...

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_spare(frame_index) ) {
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else { // normal throw
            score += rolls[frame_index] + rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

The Third Test

```c
// bowling_game.c

#include "bowling_game.h"
#include <stdbool.h>

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_spare(frame_index) ) {
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else {
            score += rolls[frame_index] +
                     rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

...

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
}
```

The Fourth Test

`test_one_strike()`

The Fourth Test

```
// bowling_game.c

#include "bowling_game.h"
#include <stdbool.h>

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_spare(frame_index) ) {
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else {
            score += rolls[frame_index] +
                     rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

```
// bowling_game.h
...
```

```
// bowling_game_test.c

...

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

...

static void test_one_strike() {
    bowling_game_init();
    bowling_game_roll(10);
    bowling_game_roll(3);
    bowling_game_roll(4);
    roll_many(16, 0);
    assert( bowling_game_score() == 24
            && "test_ane_strike()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
}
```

The Fourth Test

```c
// bowling_game.c

#include "bowling_game.h"
#include <stdbool.h>

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_spare(frame_index) ) {
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else {
            score += rolls[frame_index] +
                     rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

...

static void test_one_strike() {
    bowling_game_init();
    bowling_game_roll(10);
    bowling_game_roll(3);
    bowling_game_roll(4);
    roll_many(16, 0);
    assert( bowling_game_score() == 24
            && "test_one_strike()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
}
```

```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
bowling_game_test.c:44: failed assertion `bowling_game_score() == 24 && "test_one_strike()"'
```

The Fourth Test

```c
// bowling_game.c

...

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_spare(frame_index) ) {
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else {
            score += rolls[frame_index] +
                     rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

...

static void test_one_strike() {
    bowling_game_init();
    bowling_game_roll(10);
    bowling_game_roll(3);
    bowling_game_roll(4);
    roll_many(16, 0);
    assert( bowling_game_score() == 24
            && "test_one_strike()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
}
```

The Fourth Test

```
// bowling_game.c

...

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

static bool is_strike(int frame_index) {
    return rolls[frame_index] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_strike(frame_index) ) {
            score += 10 + rolls[frame_index+1] +
                     rolls[frame_index+2];
            frame_index += 1;
        } else if ( is_spare(frame_index) ) {
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else {
            score += rolls[frame_index] +
                     rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

```
// bowling_game.h
...
```

```
// bowling_game_test.c

...

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

...

static void test_one_strike() {
    bowling_game_init();
    bowling_game_roll(10);
    bowling_game_roll(3);
    bowling_game_roll(4);
    roll_many(16, 0);
    assert( bowling_game_score() == 24
            && "test_one_strike()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
}
```

The Fourth Test

```c
// bowling_game.c

...

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

static bool is_strike(int frame_index) {
    return rolls[frame_index] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_strike(frame_index) ) {
            score += 10 + rolls[frame_index+1] +
                     rolls[frame_index+2];
            frame_index += 1;
        } else if ( is_spare(frame_index) ) {
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else {
            score += rolls[frame_index] +
                     rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

...

static void test_one_strike() {
    bowling_game_init();
    bowling_game_roll(10);
    bowling_game_roll(3);
    bowling_game_roll(4);
    roll_many(16, 0);
    assert( bowling_game_score() == 24
            && "test_one_strike()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
}
```

The Fourth Test

```
// bowling_game.c

...

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

static bool is_strike(int frame_index) {
    return rolls[frame_index] == 10;
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_strike(frame_index) ) {
            score += 10 + rolls[frame_index+1] +
                     rolls[frame_index+2];
            frame_index += 1;
        } else if ( is_spare(frame_index) ) {
            score += 10 + rolls[frame_index+2];
            frame_index += 2;
        } else {
            score += rolls[frame_index] +
                     rolls[frame_index+1];
            frame_index += 2;
        }
    }
    return score;
}
```

```
// bowling_game.h
...
```

```
test.c

_many(int n, int pins) {
; i<n; i++)
game_roll(pins);

_one_strike() {
_init();
_roll(10);
_roll(3);
_roll(4);
, 0);
ing_game_score() == 24
test_one_strike()" );

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
}
```

The Fourth Test

```c
// bowling_game.c
...
static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

static bool is_strike(int frame_index) {
    return rolls[frame_index] == 10;
}

static int strike_score(int frame_index) {
    return 10 + rolls[frame_index+1] + rolls[frame_index+2];
}

static int spare_score(int frame_index) {
    return 10 + rolls[frame_index+2];
}

static int normal_score(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1];
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_strike(frame_index) ) {
            score += strike_score(frame_index);
            frame_index += 1;
        } else if ( is_spare(frame_index) ) {
            score += spare_score(frame_index);
            frame_index += 2;
        } else {
            score += normal_score(frame_index);
            frame_index += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

...

static void test_one_strike() {
    bowling_game_init();
    bowling_game_roll(10);
    bowling_game_roll(3);
    bowling_game_roll(4);
    roll_many(16, 0);
    assert( bowling_game_score() == 24
            && "test_one_strike()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
}
```

The Fourth Test

```c
// bowling_game.c
...
static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

static bool is_strike(int frame_index) {
    return rolls[frame_index] == 10;
}

static int strike_score(int frame_index) {
    return 10 + rolls[frame_index+1] + rolls[frame_index+2];
}

static int spare_score(int frame_index) {
    return 10 + rolls[frame_index+2];
}

static int normal_score(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1];
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_strike(frame_index) ) {
            score += strike_score(frame_index);
            frame_index += 1;
        } else if ( is_spare(frame_index) ) {
            score += spare_score(frame_index);
            frame_index += 2;
        } else {
            score += normal_score(frame_index);
            frame_index += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

...

static void test_one_strike() {
    bowling_game_init();
    bowling_game_roll(10);
    bowling_game_roll(3);
    bowling_game_roll(4);
    roll_many(16, 0);
    assert( bowling_game_score() == 24
            && "test_one_strike()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
}
```

The Fifth Test

```
test_perfect_game()
```

The Fifth Test

```c
// bowling_game.c
...
static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

static bool is_strike(int frame_index) {
    return rolls[frame_index] == 10;
}

static int strike_score(int frame_index) {
    return 10 + rolls[frame_index+1] + rolls[frame_index+2];
}

static int spare_score(int frame_index) {
    return 10 + rolls[frame_index+2];
}

static int normal_score(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1];
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_strike(frame_index) ) {
            score += strike_score(frame_index);
            frame_index += 1;
        } else if ( is_spare(frame_index) ) {
            score += spare_score(frame_index);
            frame_index += 2;
        } else {
            score += normal_score(frame_index);
            frame_index += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

...

static void test_perfect_game() {
    bowling_game_init();
    roll_many(12, 10);
    assert( bowling_game_score() == 300
            && "test_perfect_game()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
    test_perfect_game();
}
```

The Fifth Test

```c
// bowling_game.c
...
static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

static bool is_strike(int frame_index) {
    return rolls[frame_index] == 10;
}

static int strike_score(int frame_index) {
    return 10 + rolls[frame_index+1] + rolls[frame_index+2];
}

static int spare_score(int frame_index) {
    return 10 + rolls[frame_index+2];
}

static int normal_score(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1];
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_strike(frame_index) ) {
            score += strike_score(frame_index);
            frame_index += 1;
        } else if ( is_spare(frame_index) ) {
            score += spare_score(frame_index);
            frame_index += 2;
        } else {
            score += normal_score(frame_index);
            frame_index += 2;
        }
    }
    return score;
}
```

```c
// bowling_game.h
...
```

```c
// bowling_game_test.c

...

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

...

static void test_perfect_game() {
    bowling_game_init();
    roll_many(12, 10);
    assert( bowling_game_score() == 300
            && "test_perfect_game()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
    test_perfect_game();
}
```

Finally

Finally

```c
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

```c
// bowling_game.c

#include "bowling_game.h"
#include <stdbool.h>

enum { max_rolls = 21 };
static int rolls[max_rolls];
static int current_roll;

void bowling_game_init() {
    for (int i=0; i<max_rolls; i++)
        rolls[i] = 0;
    current_roll = 0;
}

void bowling_game_roll(int pins) {
    rolls[current_roll++] = pins;
}

static bool is_spare(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1] == 10;
}

static bool is_strike(int frame_index) {
    return rolls[frame_index] == 10;
}

static int strike_score(int frame_index) {
    return 10 + rolls[frame_index+1] + rolls[frame_index+2];
}

static int spare_score(int frame_index) {
    return 10 + rolls[frame_index+2];
}

static int normal_score(int frame_index) {
    return rolls[frame_index] + rolls[frame_index+1];
}

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_strike(frame_index) ) {
            score += strike_score(frame_index);
            frame_index += 1;
        } else if ( is_spare(frame_index) ) {
            score += spare_score(frame_index);
            frame_index += 2;
        } else {
            score += normal_score(frame_index);
            frame_index += 2;
        }
    }
    return score;
}
```

```c
// bowling_game_test.c

#include "bowling_game.h"

#include <assert.h>
#include <stdbool.h>

static void roll_many(int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(pins);
}

static void test_gutter_game() {
    bowling_game_init();
    roll_many(20,0);
    assert( bowling_game_score() == 0
            && "test_gutter_game()" );
}

static void test_all_ones() {
    bowling_game_init();
    roll_many(20,1);
    assert( bowling_game_score() == 20
            && "test_all_ones()" );
}

static void test_one_spare() {
    bowling_game_init();
    bowling_game_roll(5);
    bowling_game_roll(5); // spare
    bowling_game_roll(3);
    roll_many(17, 0);
    assert( bowling_game_score() == 16
            && "test_one_spare()" );
}

static void test_one_strike() {
    bowling_game_init();
    bowling_game_roll(10);
    bowling_game_roll(3);
    bowling_game_roll(4);
    roll_many(16, 0);
    assert( bowling_game_score() == 24
            && "test_one_strike()" );
}

static void test_perfect_game() {
    bowling_game_init();
    roll_many(12, 10);
    assert( bowling_game_score() == 300
            && "test_perfect_game()" );
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
    test_perfect_game();
}
```

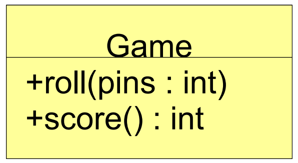
```
gcc -std=c99 -Wall bowling_game_test.c bowling_game.c && ./a.out
```

Finally

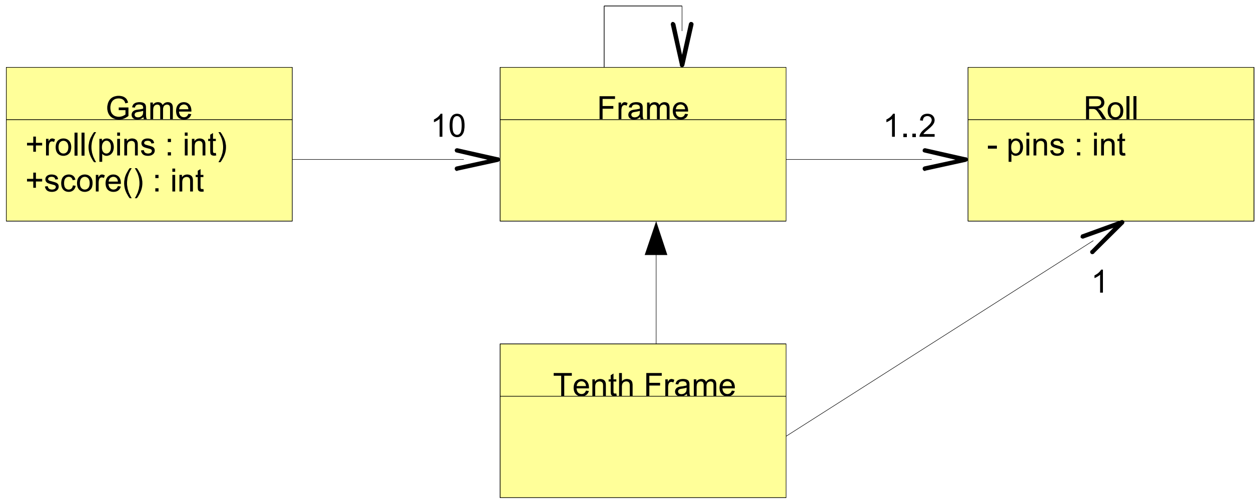
```
// bowling_game.c

...

int bowling_game_score() {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_strike(frame_index) ) {
            score += strike_score(frame_index);
            frame_index += 1;
        } else if ( is_spare(frame_index) ) {
            score += spare_score(frame_index);
            frame_index += 2;
        } else {
            score += normal_score(frame_index);
            frame_index += 2;
        }
    }
    return score;
}
```

Comparing TDD and OOAD

Design by TDD

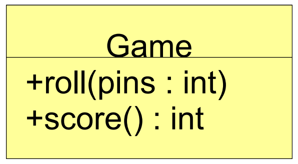


Design by OOAD

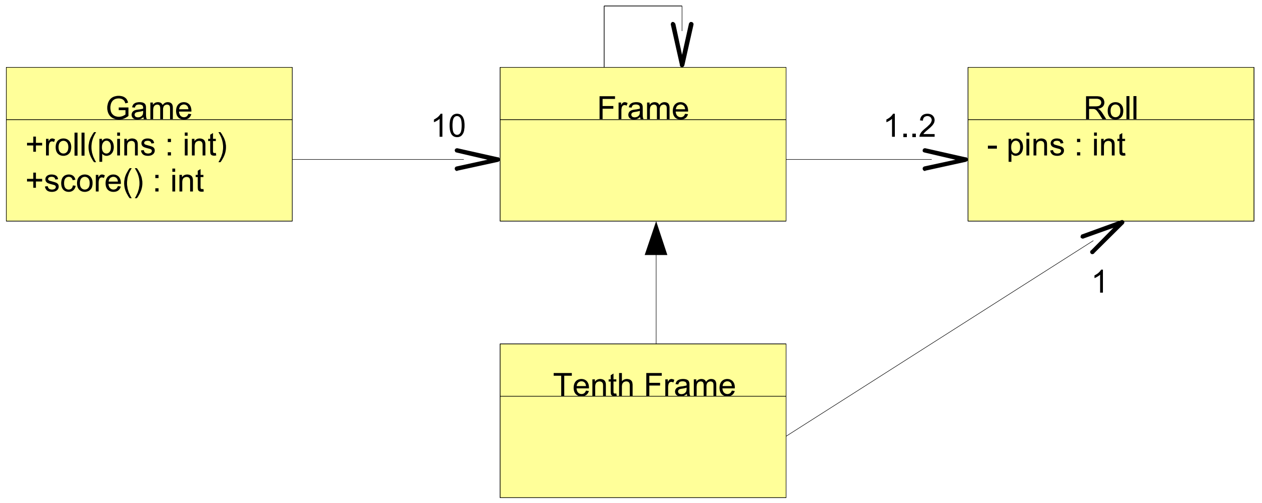


Comparing TDD and OOAD

Design by TDD



Design by OOAD



(ok, this design is not very good... but it illustrates the point well)

but some of you may ask:
Why use a ...

a naive singleton-like implementation?

```
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

Instead of...

a naive singleton-like implementation?

```
// bowling_game.h

void bowling_game_init();
void bowling_game_roll(int pins);
int bowling_game_score();
```

Instead of...

working with a more decent bowling_game object?

```
// bowling_game.h

struct bowling_game;
struct bowling_game * bowling_game_create();
void bowling_game_destroy(struct bowling_game * game);
void bowling_game_roll(struct bowling_game * game, int pins);
int bowling_game_score(struct bowling_game * game);
```

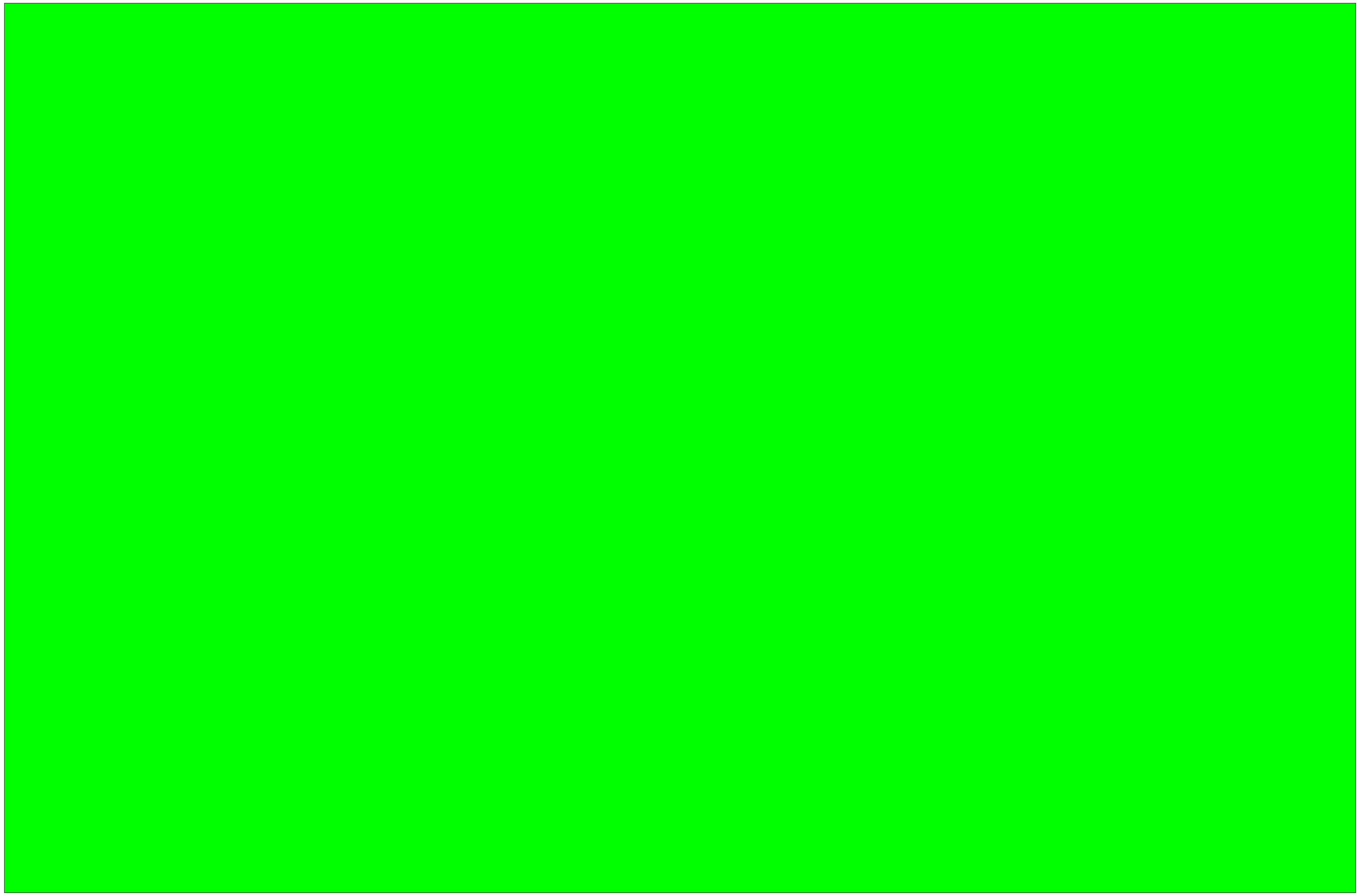
and you are probably right. So,

let's do a complete redesign

```c
// bowling_game.c

#include "bowling_game.h"
#include <stdbool.h>
#include <stdlib.h>

enum { max_rolls = 21 };

struct bowling_game {
    int rolls[max_rolls];
    int current_roll;
};

struct bowling_game * bowling_game_create() {
    struct bowling_game * game = malloc(sizeof(struct bowling_game));
    for (int i=0; i<max_rolls; i++)
        game->rolls[i] = 0;
    game->current_roll = 0;
    return game;
}

void bowling_game_destroy(struct bowling_game * game) {
    free(game);
}

void bowling_game_roll(struct bowling_game * game, int pins) {
    game->rolls[game->current_roll++] = pins;
}

static bool is_spare(struct bowling_game * game, int frame_index) {
    return game->rolls[frame_index] + game->rolls[frame_index+1] == 10;
}

static bool is_strike(struct bowling_game * game, int frame_index) {
    return game->rolls[frame_index] == 10;
}

static int strike_score(struct bowling_game * game, int frame_index) {
    return 10 + game->rolls[frame_index+1] + game->rolls[frame_index+2];
}

static int spare_score(struct bowling_game * game, int frame_index) {
    return 10 + game->rolls[frame_index+2];
}

static int normal_score(struct bowling_game * game, int frame_index) {
    return game->rolls[frame_index] + game->rolls[frame_index+1];
}

int bowling_game_score(struct bowling_game * game) {
    int score = 0;
    int frame_index = 0;
    for (int frame=0; frame<10; ++frame) {
        if ( is_strike(game, frame_index) ) {
            score += strike_score(game, frame_index);
            frame_index += 1;
        } else if ( is_spare(game, frame_index) ) {
            score += spare_score(game, frame_index);
            frame_index += 2;
        } else {
            score += normal_score(game, frame_index);
            frame_index += 2;
        }
    }
    return score;
}
```

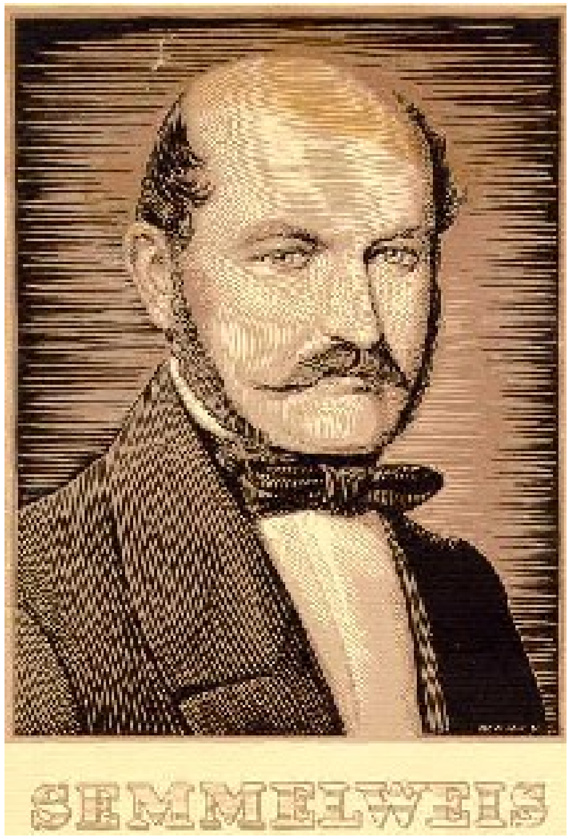
```c
// bowling_game.h

struct bowling_game;
struct bowling_game * bowling_game_create();
void bowling_game_destroy(struct bowling_game * game);
void bowling_game_roll(struct bowling_game * game, int pins);
int bowling_game_score(struct bowling_game * game);
```

```c
// bowling_game_test.c

#include "bowling_game.h"
#include <assert.h>
#include <stdbool.h>

static void roll_many(struct bowling_game * game, int n, int pins) {
    for (int i=0; i<n; i++)
        bowling_game_roll(game, pins);
}

static void test_gutter_game() {
    struct bowling_game * game = bowling_game_create();
    roll_many(game,20,0);
    assert( bowling_game_score(game) == 0 && "test_gutter_game()" );
    bowling_game_destroy(game);
}

static void test_all_ones() {
    struct bowling_game * game = bowling_game_create();
    roll_many(game,20,1);
    assert( bowling_game_score(game) == 20 && "test_all_ones()" );
    bowling_game_destroy(game);
}

static void test_one_spare() {
    struct bowling_game * game = bowling_game_create();
    bowling_game_roll(game,5);
    bowling_game_roll(game,5); // spare
    bowling_game_roll(game,3);
    roll_many(game, 17, 0);
    assert( bowling_game_score(game) == 16 && "test_one_spare()" );
    bowling_game_destroy(game);
}

static void test_one_strike() {
    struct bowling_game * game = bowling_game_create();
    bowling_game_roll(game,10); // strike
    bowling_game_roll(game,3);
    bowling_game_roll(game,4);
    roll_many(game, 16, 0);
    assert( bowling_game_score(game) == 24 && "test_one_strike()" );
    bowling_game_destroy(game);
}

static void test_perfect_game() {
    struct bowling_game * game = bowling_game_create();
    roll_many(game, 12, 10);
    assert( bowling_game_score(game) == 300 && "test_perfect_game()" );
    bowling_game_destroy(game);
}

int main() {
    test_gutter_game();
    test_all_ones();
    test_one_spare();
    test_one_strike();
    test_perfect_game();
}
```



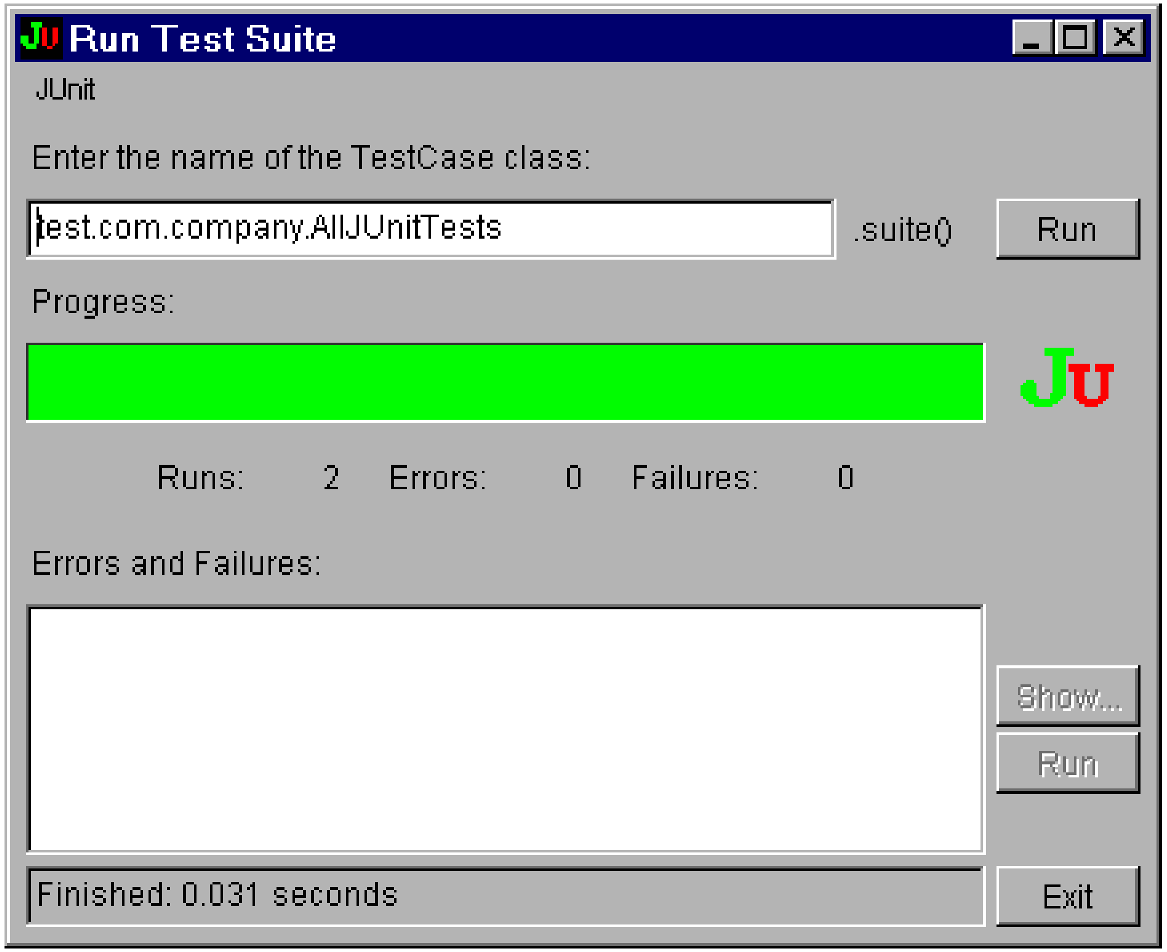
Uncle Bob compares the “discovery” of **test first** in software engineering, as somewhat equivalent to ...

the discovery of the importance of hygiene...



and in the same way as doctors now must wash their hands

we must write tests before we write code...



Enter the name of the TestCase class:

test.com.company.AllJUnitTests

Progress:

Runs: 2 Errors: 0 Failures: 0

Errors and Failures:

Finished: 0.031 seconds

A common excuse for programmers of traditional programming languages, such as C, is to say that “test-first” might work for you but we can't do it because we do not have the tools to do it.

Bullshit!

C programmers might not have all the beautyful tool support ...

but with a bit extra work, they can still practice proper “hygiene”.

TDD in C

... or The Bowling Game Kata with C and assert()