Introduction to C++

1. Using random numbers

To generate a random number rand() was used, this generates an int between 0 and RAND_MAX, which was then divided by RAND_MAX to obtain a decimal. When the program runs, it takes one argument to read this using argv, as its read as a char it had to be converted it to an int, if the user does not pass an int the program will fail as there is no validation.
To calculate the average, I added all the numbers together and then divided by the total. To calculate this total I just created an int variable and added one to it every time a number is generated.
The variance is calculated using a function that takes the mean, a list of numbers and the length of the list. By looping through all the numbers, each number has the mean subtracted and is then squared by two. Finally the variance is divided by the length of the list.

To prove that the variance is 1/12 when N tends to infinite, I ran the program using a small. A number first

```
For length:9
The mean is:0.389193
The variance is0.0768513
```

```
For length:1000
The mean is:0.497962
The variance is0.0787488
```

Then I increased the number and as shown below, the higher the number gets the closer it is to 1/12 which equals 0.8333

```
For length:999999
The mean is:0.500028
The variance is0.0832147
```

This was the highest number possible as any higher number would give a segmentation fault

2. Using random numbers

The function name dice uses the random function to generate a number then this number is evaluated. To do this I have six different options, if number is lower than 1/6 the number will be 1 if the number is higher than 1/6 but lower than 2/6 it will be equal to 2. The way this works is it will evaluate each condition and if it doesn't satisfy the condition go to the next one, therefore even though the number 3/6 is smaller than 5/6 and 6/6 it will never reach these conditions as the number will be returned before it can.
Two dice functions just runs the function twice in a row and returns the sum of the outputs.
Too generate the score I created a vector with 12 0, the first numbers won't be necessary as the dice will never return less than 1, but I found it easier to build it this way. The user will pass an argument with the number of trials wanted. For each trial the two_dice() function will be called and then each score will be stored in the respective space in the vector by adding 1. Therefore, the vector stores the frequency. A score of 3 will be stored in the space [3] of the vector.

3. The game of craps

I used a game of craps function returning a vector of ints with a variable “game” which keeps track of the current running game, e.g. if the game variable is set to true. There is another variable “throw”, which keeps track of the number of throws. The dices are thrown, and 1 is added to the number of throws. If the score is a losing number(2,3,12) win is set to 0, if the number is (7,11) then the win variable is set to 0.
If it is any other number, then this number is set to the variable point. There will then be a while loop which will only end once game is set to false. At the start of the loop the dice is thrown and 1 is added to the number of throws, if the score is equal to 7 then the game is lost and game is set to false, if the score is equal to the point then the game finishes but the player wins.
When running this program, it will take an argument which will determine the number of games executed. The result of these games is added to a file. Each number is separated by a space, although the layout of the file is not very straightforward, I preferred having a very raw format and then manipulating it with python. It is basically a list of number where the odd numbers are the result of the games(1=win and 0=loss) and the even number is the number of throws associated with the game.
Whilst I tried to avoid duplicating the code by using header files I got an error and therefore ended up just copying the same code in program2 and 3 as it seemed like an easier fix.

```
Undefined symbols for architecture arm64:
  "two_dice()", referenced from:
      game_craps() in Problem3-4aec97.o
ld: symbol(s) not found for architecture arm64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

a) How many games are won on the 1st, 2nd, 3rd...20th roll, and after the 20th roll.

Won games

Throws	Throws Games
1	2224
2	809
3	582
4	370
5	286
6	199
7	135
8	106
9	69
10	59
11	36
12	30
13	23
14	17
15	11
16	12
17	5
18	8
19	1
20	2
21	3

23	1
35	1

After the 20th roll four games are won:

3 at 21 rolls
1 at 23 rolls
1 at 35 rolls

if we include the 20th roll then 7 games are won.

b) How many games are lost on the 1st, 2nd, 3rd...20th roll, and after the 20th roll.

Lost games

Throws	Games Lost
1	1104
2	1087
3	766
4	580
5	417
6	294
7	209
8	148
9	108
10	77
11	52
12	49
13	42
14	23
15	19
16	8
17	5
18	6
19	7
20	6
21	1
22	1
24	1
25	1

After the 20th roll four games are lost:

1 at 21 rolls
1 at 22 rolls
1 at 24 rolls
1 at 25 rolls

if we include the 20th roll then 10 games are lost

c) What are the chances of winning at craps ? (include an error estimate)

To get this we divide number of wins, by total number of games

```
Total Wins: 4989 Total Loss: 5011
4989 / 10000
Probability of wins =  0.4989
```
49.9%

$$\mathrm{SE} = \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}.$$

To get the error estimate I used:

$$\sqrt{\frac{\mathbf{0.4948(1-0.4948)}}{1000}}) = 0.01581$$

Probability of winning = 49.9% ± 1.581

(d) What is the average length of a game of craps ? (include an error estimate)

To do this I summed all the lengths of each game and divided by total game.
The average length is 3.3868 and the standard deviation is 3.0225319
Estimate error, requires the standard deviation which is equal to:

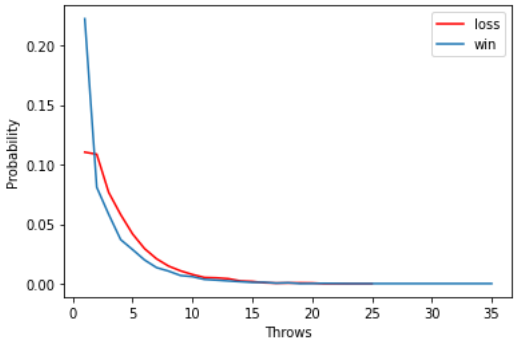
$$SE_{\bar{x}} = \frac{s}{\sqrt{n}}$$

$$\frac{3.0225}{\sqrt{1000}} = 0.09557$$

Average length = 3.3868 ± 0.09557

(e) Do the chances of winning improve with the length of the game ?

The following graph shows the cumulative probability of winning and losing a game each round, considering the total games. Number of wins is divided by total games. Initially the probability for a win is higher but then rapidly decreases. Therefore, the chances of winning are higher at the beginning and then are pretty much equal to the chance of losing.
Most games are won or lost in the first rounds, (average length is around 3 so this makes sense) The issue is this graph will show lower probabilities towards the end as there are less games being played at the end.
Probabilities of game being won or loss in a round



The following graph displays the probability in each round of winning or losing, this was used to evaluate each round individually and see the outcome clearer. Each probability is calculated by dividing the total number of wins for that number of throws by the sum of the total number of wins and losses for that throw. This graph shows us that the probability of winning in different rounds. As the game gets longer it first decreases, it then increases again and finally will decrease. The graph stops at 20 as barely any games were won or lost after 20 throws. In conclusion, the round that obtained a higher chance of winning was still the first round to be played