

ACCA4017 - Anton Hall 1809131

by Anton Hall

FILE	12316_ANTON_HALL_ACCA4017_- _ANTON_HALL_1809131_157877_1027300193.DOCX (749.11K)		
TIME SUBMITTED	23-FEB-2019 11:19PM (UTC+0000)	WORD COUNT	4271
SUBMISSION ID	101086288	CHARACTER COUNT	25029

University of Wales, Trinity Saint David

Software Development – ACCA4017

Programs in C and Documentation

Anton Hall

Contents

Introduction	1
Part A (Understanding)	1
1. Variables.....	1
2. For Loops.....	2
3. Arrays	3
Part B (Implementation)	4
1. Exercise 1	4
1.1 Problem.....	4
1.2 Pseudocode.....	4
1.3 Annotated Code	4
1.4 Testing.....	5
1.4.1 Normal	5
1.4.2 Extreme	5
1.4.3 Erroneous	6
1.4.4 Recommendations	6
2. Exercise 2	7
2.1 Problem.....	7
2.2 Pseudocode.....	7
2.3 Annotated Code	8
2.4 Testing.....	9
2.4.1 Normal	9
2.4.2 Extreme	9
2.4.3 Erroneous	11
2.4.4 Recommendations	12
3. Exercise 3	13
3.1 Problem.....	13
3.2 Pseudocode.....	13
3.3 Annotated Code	13
3.4 Testing.....	14
3.4.1 Normal	14
3.4.2 Extreme	15
3.4.3 Erroneous	16
3.4.4 Recommendations	17
4. Exercise 4	18
4.1 Problem.....	18

4.2 Pseudocode.....	18
4.3 Annotated Code	19
4.4 Testing.....	21
4.4.1 Normal	21
4.4.2 Erroneous.....	24
4.4.3 Recommendations	24
Conclusion.....	24

Introduction

The purpose of this assignment is to demonstrate a good level of understanding and provide examples of the use of the C programming language. The assignment will be divided into part A and part B. Part A will include the explanation of Variables, for loops and arrays, and how they are important in programming. Part B will include four example programs that escalate in complexity broken down into problem analysis, pseudocode, annotated C code and testing results.

Part A (Understanding)

1. Variables

A variable in a program acts as a predefined named location in memory which is used to store values that can be utilised throughout code. In the Dictionary of Computing a variable is described as: "A unit of storage that can be modified during program execution, usually by assignment or read operations. A variable is generally denoted by an identifier or by a name." [1] The previous quote states exactly how a variable in a program behaves in its most basic form, it is a pre-defined container where data can be stored to be used throughout a program. Variables are normally set up at the start of a program by declaring the variable type and its value, but it can also be altered throughout the code. Variables are important in coding as they can be used to store user inputs and the inputs can be different each time a program is executed, making programs more flexible. Once a variable is defined it can be called numerous times, making a program easier to write, rather than having to hard code the same value repeatedly.

When declaring a variable name in C there are some basic rules that should be followed. Firstly, the name of the variable is important, it should replicate the values that the variable is likely to store, for example if you are storing someone's age then declare the variable as 'age'. C is case sensitive so variable names should begin with either an underscore or a lower-case letter. They cannot start with a number and cannot have spaces. Variables should also not be declared as words that are reserved in C. Secondly, you must declare the variable as a specific type so that it can receive a specific type of value, the table below shows some of the basic variable types that are used in C.

Type	Defined by	Description
Character	char	Used to store a single letter character
Floating point	float	Used to store a number with a decimal part
Integer	int	Used to store a whole number

Table 1 - Basic C Variables

When writing code, it is also important to consider how your variable is going to be used. If you want a variable that is specific to a program function and can only be manipulated by that function, then you would declare a local variable inside the function. Alternatively, if you wanted a variable that can be called throughout any of the program functions you would declare a global variable. Global variables can be used throughout the program and can be manipulated by any of the program functions. Global variables are declared outside of any functions and are generally are the start of the code.

2. For Loops

For loops are used as a type of repetition that allows you to create a loop in code that needs to iterate a specific number of times. For loops can also be referred to as counting loops as they are used in code when you want a loop to only be executed a set number of times. This is achieved by using a count inside the loop that goes up by one each time the loop is executed, when a condition is met.

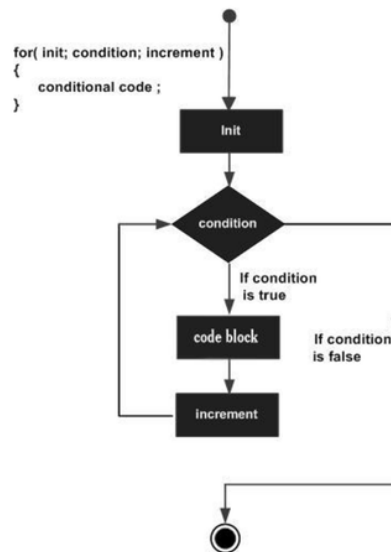


Figure 1 - For loop flow chart example

For loops can be described in three separate steps as shown by the diagram above and example code below:

1. Initialise – This is where the count variable gets initialised.
2. Condition – This is where the counter variable is tested against a given condition. If the variable returns true to the condition, then the loop will get executed. If the variable returns false to the condition, the loop is terminated.
3. Increment – When the loop is successfully executed the count variable is either incremented or decremented depending of the operation set up in the loop condition. [2]

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

```
int main()
```

```
{
```

```
    int i;
```

```
    for (i=1; i<=; i++)
```

```
    {
```

```
        printf("%d\n", i);
```

```
    }
```

```
}
```

When using a count loop in code we generally name the count variable 'i', therefore you should refrain from naming any other variable 'i'.

A for loop could be used in code if you wanted a user to input something a specific number of times. For example, if you wanted a user to input their five favourite films into a list then you would use a for loop to increment each time the user enters their favourite film whilst saving each film choice into a variable. You could then print the stored variable once the for loop has completed its iterations.

3. Arrays

An array is a type of variable that can hold several values of a single type. The size of the array (number of different values it can store) is declared when you create the array variable, this is fixed and cannot be changed later in the code. Each item in an array is referred to as an element and each element can be manipulated using the elements index number. All index numbering begins at 0, therefore, to access element 5 you would use index 4 as illustrated in the table below: [3]

Array length	5				
Index	0	1	2	3	4
Element	1	2	3	4	5

Table 2 - Array indexing example

If we think of arrays as a useful tool to store a collection of variables of the same type, we can understand how they can be useful in code. If we need to store the results of 30 different student exams in a program, we could easily create a variable for each of the 30 students and use an input function to manipulate each of the 30 variables, however this would be time consuming and be quite a lengthy program to perform a very simple task. Instead of this, we can use one array to store all 30 students results and use one counting loop to input the results at each of the 30 elements within the array, this would save the programmer a lot of time when writing the program.

Part B (Implementation)

1. Exercise 1

1.1 Problem

This exercise will demonstrate the use of different variables to store different types of information within a program. The first variable we need to store is a name, so we will need to use the char variable which allows us to store a letter, however as we need to store a full name, we will need to make this variable an array. A height is also required to be stored, therefore this will have to be declared as a float variable as heights are normally decimal numbers. The final variable is an age, as this is normally recorded as a whole number the int variable can be used. These three stored variables will then need to be printed to the screen for the user to read.

1. Create variables (char, float, int) and declare them.
2. Print the variables using the correct format specifier for each variable type (char = %s, float = %f, int = %d)

1.2 Pseudocode

Main()

Set name [6] = name

Set height = height

Set age = age

Print "name"

Print "height"

Print "age"

1.3 Annotated Code

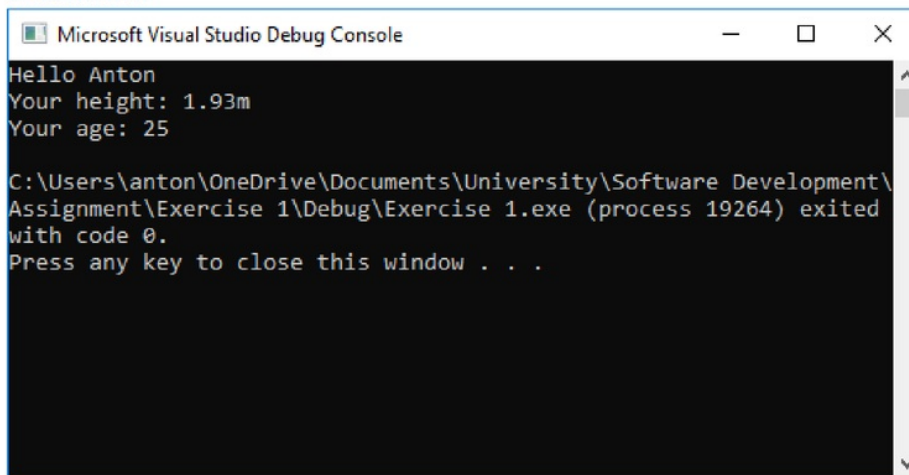
```
#include <stdio.h>

void main()
{
    char name[6] = { 'A', 'n', 't', 'o', 'n' }; // char is the variable type - single
    character. C doesn't have the "string" function so we must create an array.
    float height = 1.93; // float is the variable type - decimal number
    int age = 25; // int is the variable type - whole number

    printf("Hello %s\n", name); //prints the name variable
    printf("Your height: %.2fm\n", height); //prints the height variable
    printf("Your age: %d\n", age); //prints the age variable
}
```

1.4 Testing

1.4.1 Normal



```
Microsoft Visual Studio Debug Console

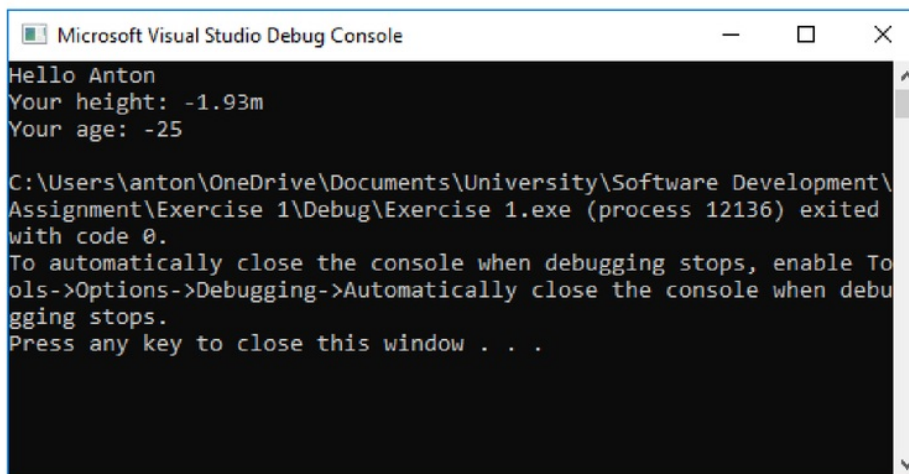
Hello Anton
Your height: 1.93m
Your age: 25

C:\Users\anton\OneDrive\Documents\University\Software Development\
Assignment\Exercise 1\Debug\Exercise 1.exe (process 19264) exited
with code 0.
Press any key to close this window . . .
```

Result: Program ran as expected and the variables that were input were printed correctly

1.4.2 Extreme

Using negative values:



```
Microsoft Visual Studio Debug Console

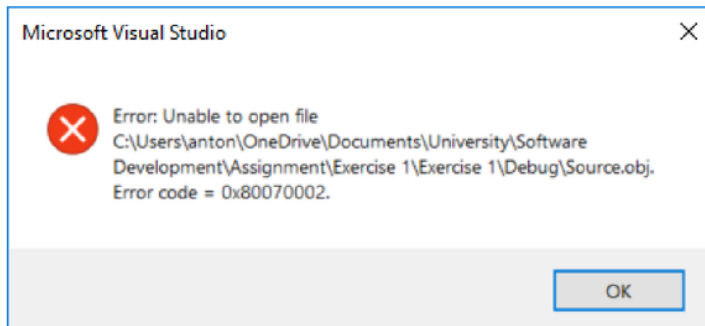
Hello Anton
Your height: -1.93m
Your age: -25

C:\Users\anton\OneDrive\Documents\University\Software Development\
Assignment\Exercise 1\Debug\Exercise 1.exe (process 12136) exited
with code 0.
To automatically close the console when debugging stops, enable To
ols->Options->Debugging->Automatically close the console when debu
gging stops.
Press any key to close this window . . .
```

Result: Program ran as expected and the variables that were input were printed correctly

1.4.3 Erroneous

Using the wrong character type for the variable used:



Result: Program fails to run

1.4.4 Recommendations

This is a very simple program so it is not particularly worthwhile spending a considerable amount of time perfecting it, however some simple improvements could be made to make this program more user friendly. The scanf function could be used to allow the user to input their own variables. Validation of the inputs would stop the program printing negative values and completely failing when the wrong character type is used.

2. Exercise 2

2.1 Problem

This program will need to allow the user to input values to calculate the areas of a circle and a triangle. Once these areas have been calculated the program will need to subtract the area of the triangle from the circle to give the user the total area of the circle, minus the triangle. As this program requires the user to input the different values, the scanf function will need to be used to store what the user enters in the relevant variable. These values will then need to be used to calculate the areas of each shape.

1. Create and store a variable for pi (float)
2. Create variables to store user inputs and area calculations (float)
3. Ask user to enter the radius of the circle (printf)
4. Store the users input (scanf)
5. Calculate the area and store the result
6. Show the user the calculated area of the circle
7. Ask the user to input the height of the triangle
8. Store the users input
9. Ask the user to input the base of the triangle
10. Store the users input
11. Calculate the area of the triangle
12. Show the user the calculated area of the triangle
13. Calculate the area of the circle minus the area of the triangle
14. Show the user the result of the program

2.2 Pseudocode

Set constant pi

main()

float circle, triangleh, triangleb, areac, areat, areas

print "Please enter the radius of your circle"
input "circle"

set areac = pi * (circle*circle)
print "The area of your circle is " + areac

print "Please enter the height of your triangle"
input "triangleh"
print "Please enter the base of your triangle"
input "triangleb"
set = triangleh * triangleb / 2

print "The area of your triangle is " + areat

set areas = areac – areat
print "The area of your circle minus the triangle is " + areas

2.3 Annotated Code

```
#include <stdio.h>

#define pi 3.14159265359 //declare a constant for pi to be used to calculate the area
of the circle

void main()
{
    float circle = 0, triangleh = 0, triangleb = 0, areac = 0, areat = 0, areas =
0; // create variables to hold the values the user inputs and the values calculated

    printf("Please enter the radius of your circle:");// ask the user to enter the
radius of the circle
    scanf("%f", &circle); // store the users input to the circle variable

    areac = pi * (circle*circle); // calculate the area of the circle using pi and
the radius the user entered and store the value

    printf("The area of your circle is %.2f\n\n", areac); //show the user the area
of the circle

    printf("Please enter the height of your triangle:");// ask the user to enter
the height of their triangle
    scanf("%f", &triangleh); // store the users input to the triangleh variable
    printf("Please enter the base of your triangle:");// aks the user to enter the
base of their trianlge
    scanf("%f", &triangleb); // store the users inpute to the trangleb variable

    areat = triangleh * triangleb / 2; // calculate the area of the triangle using
the values the user entered

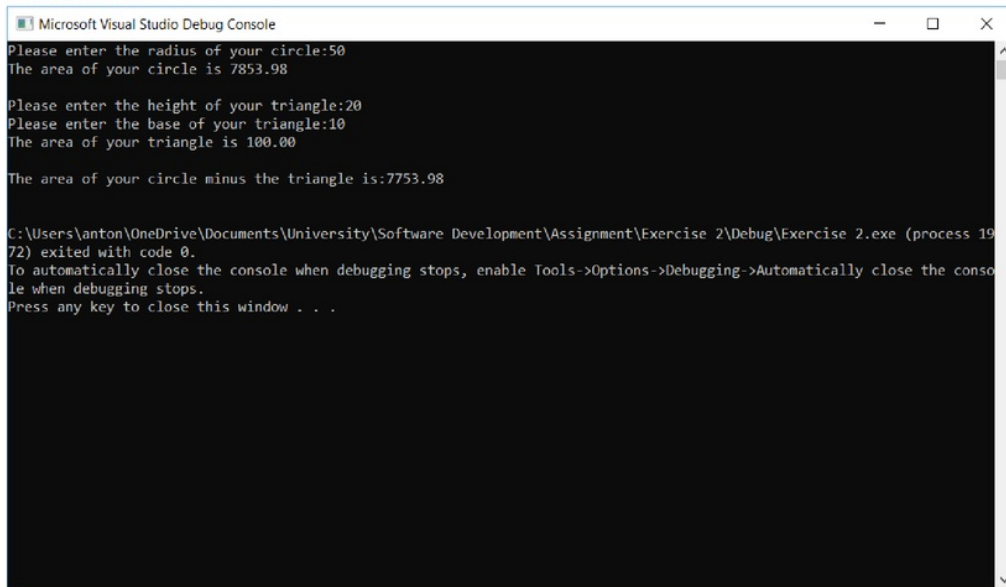
    printf("The area of your triangle is %.2f\n\n", areat); // show the user the
area of the triangle

    areas = areac - areat; // calculate the area of the circle minus the triangle

    printf("The area of your circle minus the triangle is: %.2f\n\n", areas); // show
the user the area of the circle minus the area of the triangle
}
```

2.4 Testing

2.4.1 Normal



```
Microsoft Visual Studio Debug Console
Please enter the radius of your circle:50
The area of your circle is 7853.98

Please enter the height of your triangle:20
Please enter the base of your triangle:10
The area of your triangle is 100.00

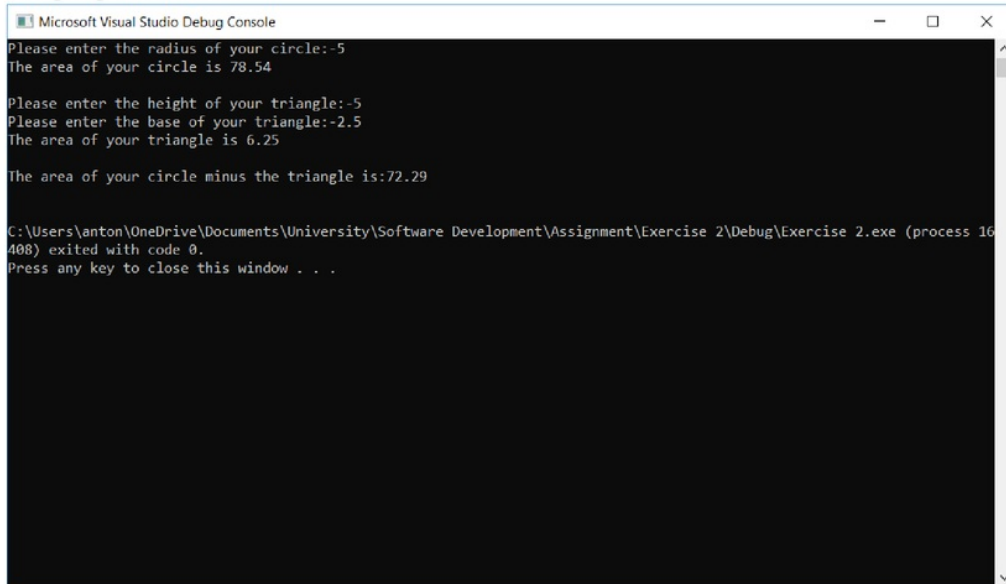
The area of your circle minus the triangle is:7753.98

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 2\Debug\Exercise 2.exe (process 1972) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Result: Program runs as expected. All user inputs are stored correctly, and the areas of each shape are calculated correctly

2.4.2 Extreme

Using negative values:



```
Microsoft Visual Studio Debug Console
Please enter the radius of your circle:-5
The area of your circle is 78.54

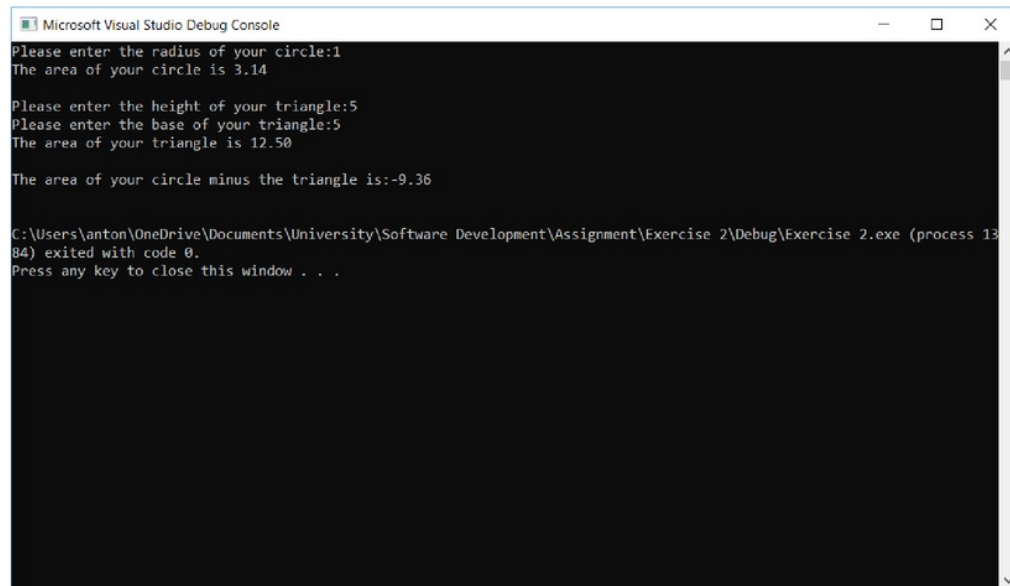
Please enter the height of your triangle:-5
Please enter the base of your triangle:-2.5
The area of your triangle is 6.25

The area of your circle minus the triangle is:72.29

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 2\Debug\Exercise 2.exe (process 16408) exited with code 0.
Press any key to close this window . . .
```

Result: Program runs as expected but gives a positive area

Making the triangle bigger than the circle:



```
Microsoft Visual Studio Debug Console
Please enter the radius of your circle:1
The area of your circle is 3.14

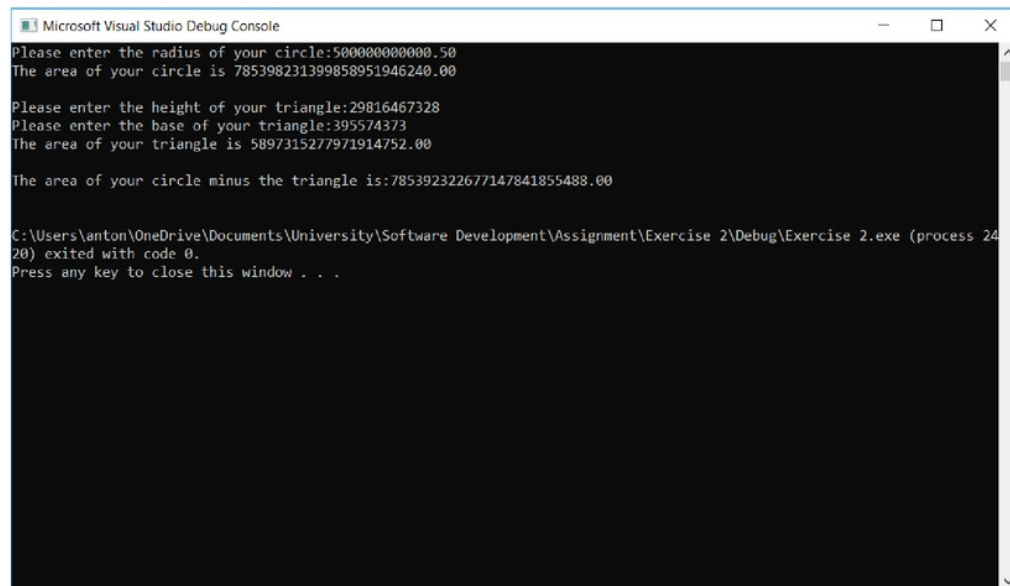
Please enter the height of your triangle:5
Please enter the base of your triangle:5
The area of your triangle is 12.50

The area of your circle minus the triangle is:-9.36

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 2\Debug\Exercise 2.exe (process 1384) exited with code 0.
Press any key to close this window . . .
```

Result: Program runs as expected, however the final calculation shows a negative value and the user shouldn't have a negative area of a shape. An if statement could be added to the code to tell the user that the area of the triangle is bigger than that of the circle, or a while loop to restart the program until a positive result is achieved.

Using large values:



```
Microsoft Visual Studio Debug Console
Please enter the radius of your circle:500000000000.50
The area of your circle is 785398231399858951946240.00

Please enter the height of your triangle:29816467328
Please enter the base of your triangle:395574373
The area of your triangle is 5897315277971914752.00

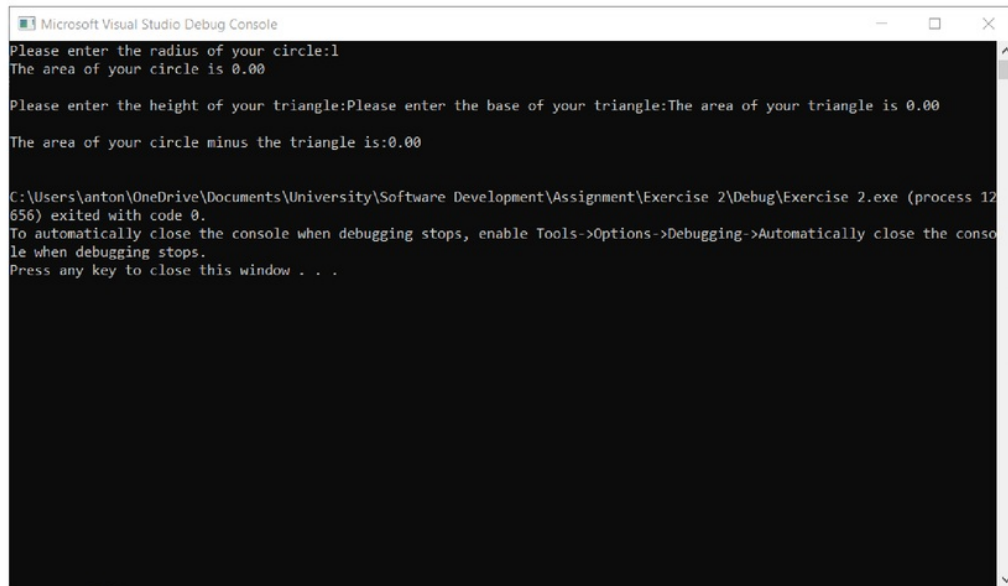
The area of your circle minus the triangle is:785392322677147841855488.00

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 2\Debug\Exercise 2.exe (process 2420) exited with code 0.
Press any key to close this window . . .
```

Result: Program runs as expected

2.4.3 Erroneous

Entering a character:



```
Microsoft Visual Studio Debug Console
Please enter the radius of your circle:1
The area of your circle is 0.00

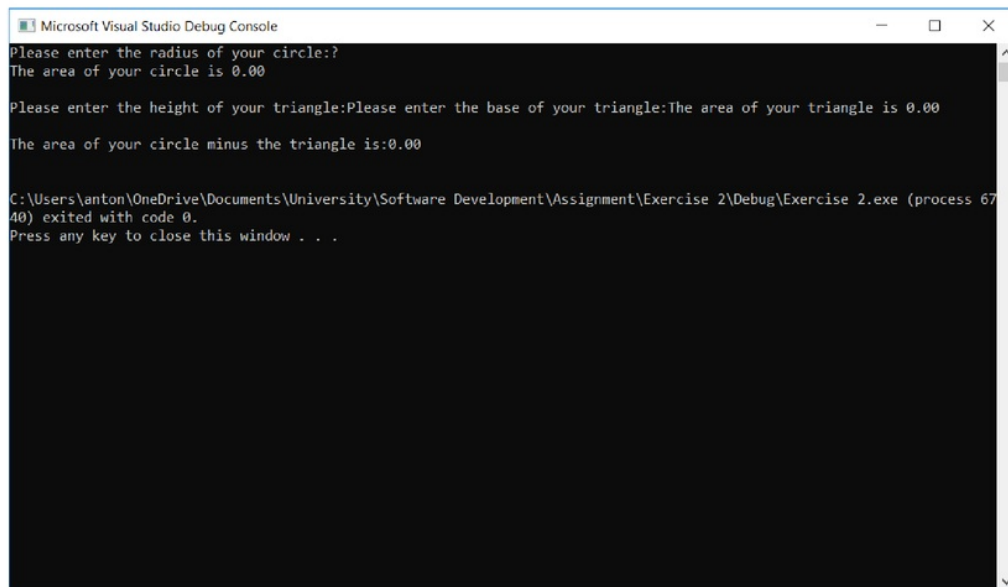
Please enter the height of your triangle:Please enter the base of your triangle:The area of your triangle is 0.00

The area of your circle minus the triangle is:0.00

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 2\Debug\Exercise 2.exe (process 12656) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Result: Program fails as expected

Entering a special character:



```
Microsoft Visual Studio Debug Console
Please enter the radius of your circle:??
The area of your circle is 0.00

Please enter the height of your triangle:Please enter the base of your triangle:The area of your triangle is 0.00

The area of your circle minus the triangle is:0.00

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 2\Debug\Exercise 2.exe (process 6740) exited with code 0.
Press any key to close this window . . .
```

Result: Program fails as expected

2.4.4 Recommendations

This program runs well but some improvements could be made to make it more intuitive. Firstly, you could add a test loop to check that the area of the triangle is not bigger than that of the circle. In addition, you could add some validation so that negative numbers cannot be input into the program as negative inputs in this instance would not make sense.

3. Exercise 3

3.1 Problem

This program needs to allow a user to enter the price of five products into an array. It makes sense to use an array for this problem as if the user is entering a price five times all five prices will be of the same variable type. Therefore, we only need one array instead of five separate variables, even though an array is slightly more complex to create. To allow the user to enter the five prices into the array we will need a counting loop to store each of the five prices into the array variable at the correct index number. Once the five prices have been entered, the program will need to calculate a total price of the items and display this to the user, then it will need to calculate the VAT on the total price and add this to the total price and display this to the user.

1. Set VAT as a constant – as this value will never change.
2. Create a count variable
3. Create an array with 5 index's, a subtotal, and total variables
4. Create a for loop to iterate 5 times
- 4.4 Ask the user to input the price for item 1
- 4.5 Add item one to the sum (repeat 5 times)
5. Print the total price of the five items
6. Calculate the total including VAT
7. Print the total price including VAT

3.2 Pseudocode

```
constant VAT 0.2
```

```
main()
```

```
    set i = 0
    set priceArray[5] {0,0,0,0,0}, sum = 0, sumvat = 0

    for (i = 0; i < 5; i++)
    {
        print "Please enter the price of item" + i + 1
        input priceArray[i]
        set sum = sum + priceArray[i]
    }
    print "The total price of your items is" + sum

    set sumvat = sum + sum * VAT

    print "The total price of your items including VAT is" + sumvat
```

3.3 Annotated Code

```
#include <stdio.h>

#define VAT 0.2 // create a constant for VAT
```

```

int main() {

    int i = 0; // set the count variable i to 0
    float priceArray[5] = { 0, 0, 0, 0, 0 }, sum = 0, sumvat = 0; // create an
    array, total sum and total VAT sum variable and initialise all values to 0

    for (i = 0; i < 5; i++) { //Start a for loop that will loop 5 times

        printf("Please enter the price of item %d:\n", i + 1); //Ask the user to
        enter the price of the item that corresponds the for loop count +1 (as for loops start
        at 0)
        scanf("%f", &priceArray[i]); // store the user input to the array that
        corresponds to the for loop count
        sum = sum + priceArray[i]; // store the current array value into the sum
        variable
    }

    printf("\nThe total price of your items is: %.2f\n", sum); // print the total
    price of the items entered by the user

    sumvat = sum + sum * VAT; // calculate the total price including VAT

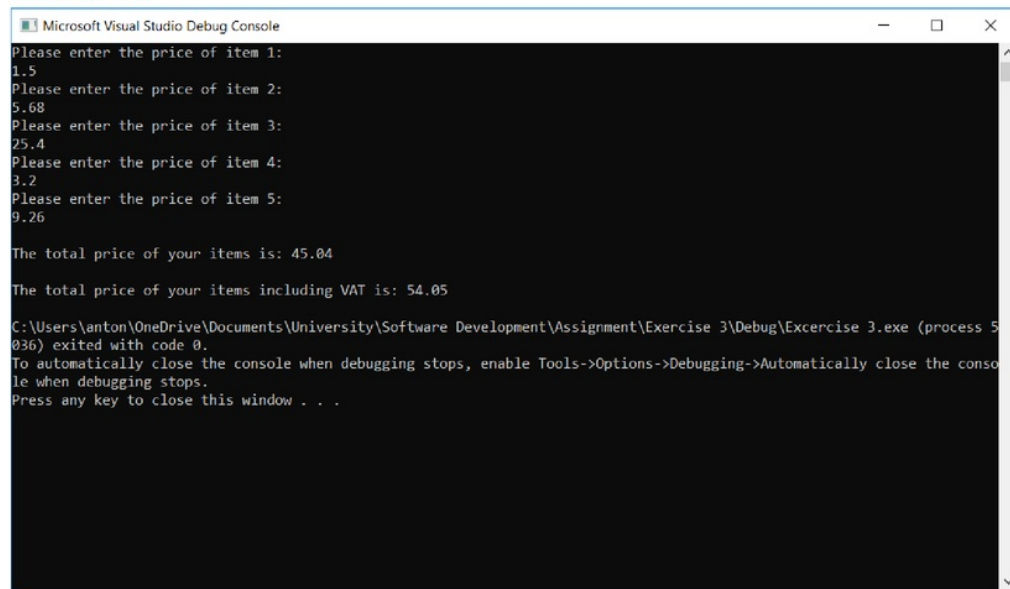
    printf("The total price of your items including VAT is: %.2f\n", sumvat); //
    print the total price of the items plus VAT entered by the user

}

```

3.4 Testing

3.4.1 Normal



```

Microsoft Visual Studio Debug Console
Please enter the price of item 1:
1.5
Please enter the price of item 2:
5.68
Please enter the price of item 3:
25.4
Please enter the price of item 4:
3.2
Please enter the price of item 5:
9.26

The total price of your items is: 45.04

The total price of your items including VAT is: 54.05

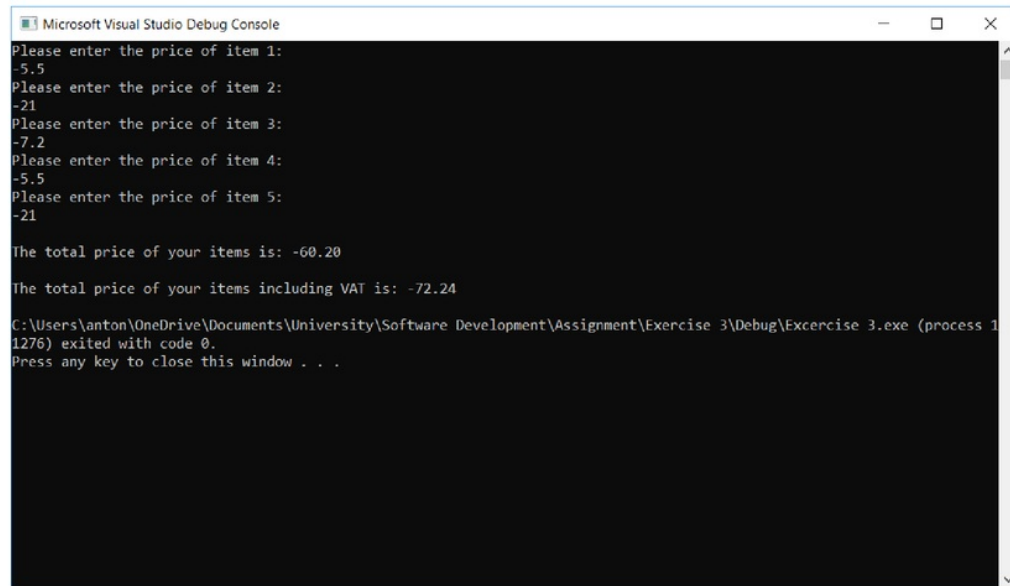
C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 3\Debug\Exercise 3.exe (process 5036) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

Result: Program runs as expected

3.4.2 Extreme

Using negative values:



```
Microsoft Visual Studio Debug Console
Please enter the price of item 1:
-5.5
Please enter the price of item 2:
-21
Please enter the price of item 3:
-7.2
Please enter the price of item 4:
-5.5
Please enter the price of item 5:
-21

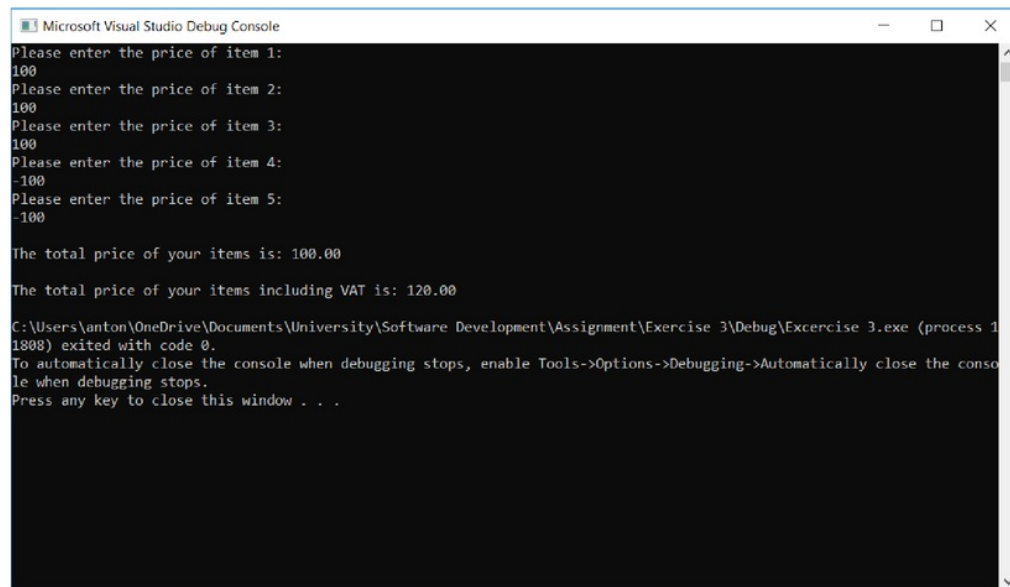
The total price of your items is: -60.20

The total price of your items including VAT is: -72.24

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 3\Debug\Excercise 3.exe (process 1276) exited with code 0.
Press any key to close this window . . .
```

Result: Program runs as expected and produces a correct result

Using both positive and negative numbers:



```
Microsoft Visual Studio Debug Console
Please enter the price of item 1:
100
Please enter the price of item 2:
100
Please enter the price of item 3:
100
Please enter the price of item 4:
-100
Please enter the price of item 5:
-100

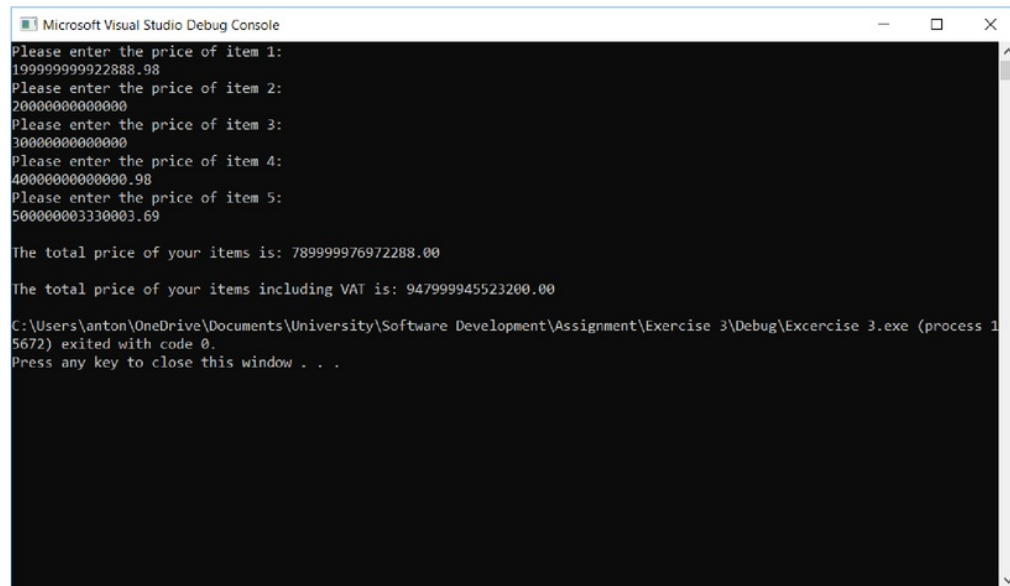
The total price of your items is: 100.00

The total price of your items including VAT is: 120.00

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 3\Debug\Excercise 3.exe (process 1808) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Result: Program fails and produces an incorrect result

Using Large numbers:



```
Microsoft Visual Studio Debug Console
Please enter the price of item 1:
19999999922888.98
Please enter the price of item 2:
20000000000000
Please enter the price of item 3:
30000000000000
Please enter the price of item 4:
40000000000000.98
Please enter the price of item 5:
500000003330003.69

The total price of your items is: 78999976972288.00

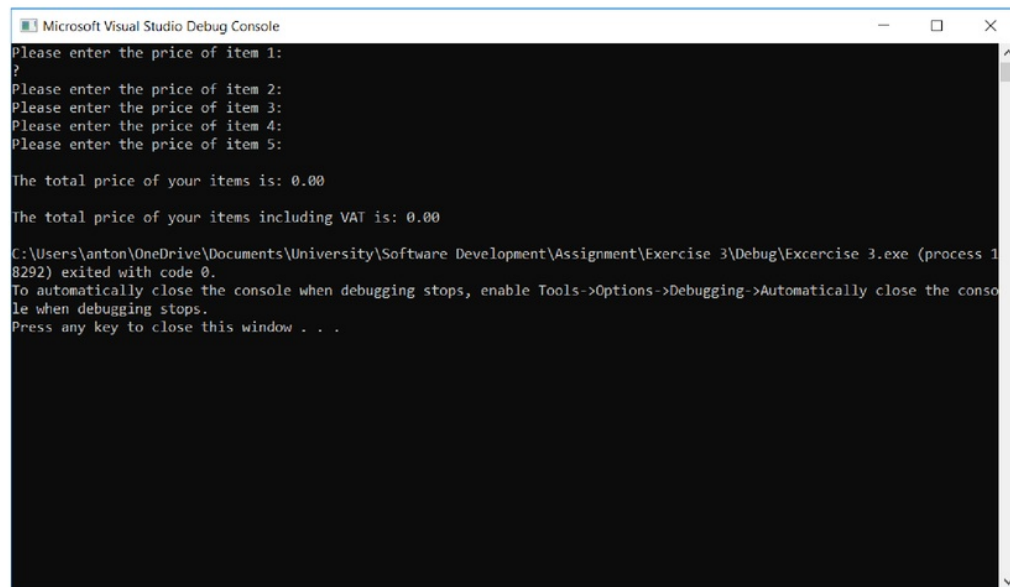
The total price of your items including VAT is: 94799945523200.00

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 3\Debug\Excercise 3.exe (process 15672) exited with code 0.
Press any key to close this window . . .
```

Result: Program runs as expected

3.4.3 Erroneous

Using a special character:



```
Microsoft Visual Studio Debug Console
Please enter the price of item 1:
?
Please enter the price of item 2:
Please enter the price of item 3:
Please enter the price of item 4:
Please enter the price of item 5:

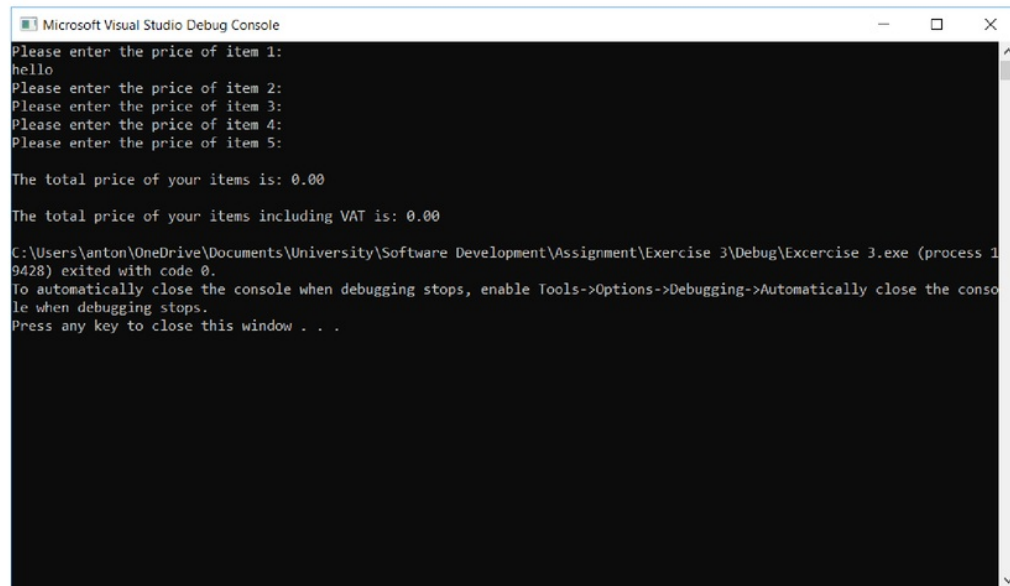
The total price of your items is: 0.00

The total price of your items including VAT is: 0.00

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 3\Debug\Excercise 3.exe (process 18292) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Result: Program ends

Using letters:



```
Microsoft Visual Studio Debug Console
Please enter the price of item 1:
hello
Please enter the price of item 2:
Please enter the price of item 3:
Please enter the price of item 4:
Please enter the price of item 5:

The total price of your items is: 0.00
The total price of your items including VAT is: 0.00

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise 3\Debug\Exercise 3.exe (process 19428) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Result: Program ends

3.4.4 Recommendations

This program achieves its job by summing prices and creating a price total with VAT, but only if the correct values are input into the program. To improve this program, I would add validation rules to the input so that instead of the program ending instantly when an incorrect value is entered, it would simply ask the user to try again.

4. Exercise 4

4.1 Problem

This program brief is creating a virtual fruit machine where three wheels of numbers theoretically spin and give a random result and those results correspond with points the user will win. The first problem we face when creating a program like this is creating random numbers. There are functions in c that we can utilise to create these random numbers. The next problem to overcome is displaying the correct points score to the user that corresponds with their random result of number, an if statement can be used to achieve this. The final point of the brief that the code must adhere to is that the user can have as many goes as they like; a while loop will allow the user to play as many times as they wish.

1. Seed random number with system clock, only do this once
2. Run slot machine function
3. Set variables – random numbers and high and low boundaries
4. Set spin variable
5. Welcome player to the game
6. Start while loop
7. Get random numbers from random function
8. Start if statement and print results
9. Ask user if they want to play again which will run the while loop again or end the program

4.2 Pseudocode

random prototype

slotmachine prototype

random(int, int)

```
{  
    set myrand = 0  
    set myrand = rand() % (high - low + 1) + low  
    return myrand  
}
```

slotmachine()

```
{  
    set random1 = 0, random2 = 0, random3 = 0, score = 0, low = 1, high = 5  
    set spin = 'Y'  
  
    print "Welcome message"  
  
    while (spin is 'Y')  
    {  
        random1 = random(low, high)  
        random2 = random(low, high)  
        random3 = random(low, high)  
  
        if ((random1 = 1) & (random2 = 1) & (random3 = 1)  
            or (random1 = 2) & (random2 = 2) & (random3 = 2)  
            or (random1 = 3) & (random2 = 3) & (random3 = 3)  
            or (random1 = 4) & (random2 = 4) & (random3 = 4))  
        {  
            // ...  
        }  
    }  
}
```

```

        {
            Set score = 20
            print "random1, random2, random3"
            print "You win, and your points are:" +score
        }
        else if ((random1 = 5) & (random2 = 5) & (random3 = 5))
        {
            Set score = 2000
            Print "random1, random2, random3"
            Print "***JACKPOT**You win, and your points are: **JACKPOT**"
            +score
        }
        else if ((random1 = 5) & (random2 = 5) or (random1 = 5) & (random3 = 5) or
(random2 = 5) & (random3 = 5))
        {
            Set score = 100
            Print "random1, random2, random3"
            Print "You win, and your points are:" + score
        }
        else
        {
            Set score = 0
            Print "random1, random2, random3"
            Print "You lose, and your points are:" + score)
        }
        Print "Do you want to spin again?(Y/N)"
        Input spin
    }
}

void main()
{
    Seed random
    Call slotmachine function
}

```

4.3 Annotated Code

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int random(int, int); // prototype
void slotmachine(void); // prototype

int random(int low, int high) { // function to create random numbers
    int myrand = 0; // create variable to carry random number and initialise to 0
    myrand = rand() % (high - low + 1) + low; //generate random number using high
and low variable values
    return myrand; //return random number
}

```

```

void slotmachine() { //function to run slot machine
    int random1 = 0, random2 = 0, random3 = 0, score = 0, low = 1, high = 5;
    //create variables to store random numbers and initialise values
    char spin = 'Y'; //create variable to store user input on spin choice

    printf("****Welcome to the Slot Machine****\n"); //Welcome users to the program

    while (spin == 'Y' || spin == 'y') { //start a while loop to run through the
program until the user decides to finish
        random1 = random(low, high); //set first random number to random number
gained from running the random function
        random2 = random(low, high); //set second random number to random number
gained from running the random function
        random3 = random(low, high); //set third random number to random number
gained from running the random function

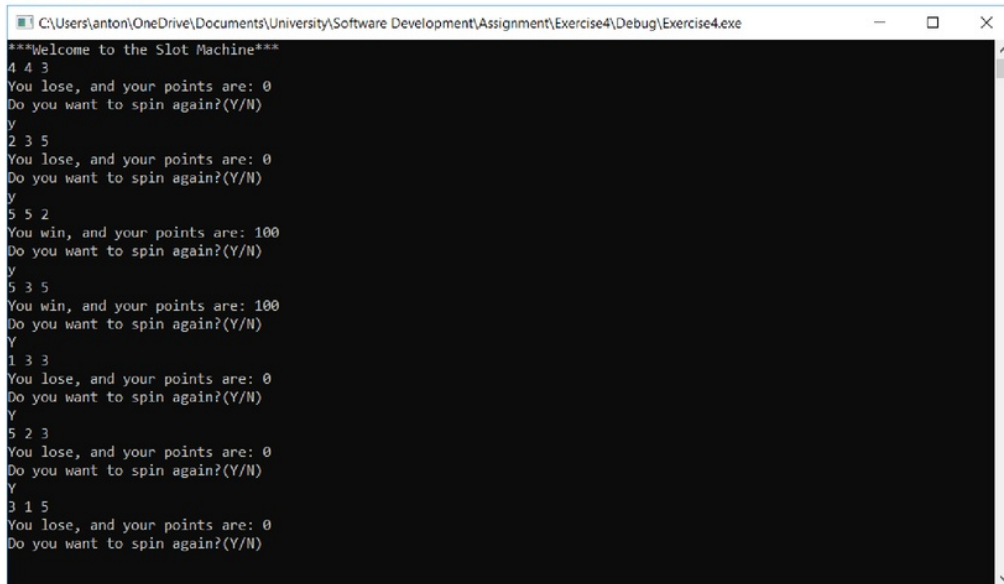
        if ((random1 == 1) && (random2 == 1) && (random3 == 1) // start if
statement to give results to the user that correspond to the random numbers generated
            || (random1 == 2) && (random2 == 2) && (random3 == 2)
            || (random1 == 3) && (random2 == 3) && (random3 == 3)
            || (random1 == 4) && (random2 == 4) && (random3 == 4)) { //first
argument - if all numbers are the same eg. '1' & '1' & '1' then score is 20
                score = 20; //set score variable to 20 if first argument is
achieved
                printf("%d %d %d\n", random1, random2, random3); //print random
numbers generated
                printf("You win, and your points are: %d\n", score); //print a
win message and score achieved
            }
            else if ((random1 == 5) && (random2 == 5) && (random3 == 5)) { //second
argument - if three 5s are generated
                score = 2000; //set score variable to 2000 if second argument is
achieved
                printf("%d %d %d\n", random1, random2, random3); //print random
numbers generated
                printf("****JACKPOT**You win, and your points are:
%d**JACKPOT**\n", score); //print win message and score achieved
            }
            else if ((random1 == 5) && (random2 == 5) || (random1 == 5) && (random3
== 5) || (random2 == 5) && (random3 == 5)) { //third argument - if any two numbers are
5
                score = 100; //set score variable to 100 if third argument is
achieved
                printf("%d %d %d\n", random1, random2, random3); //print random
numbers generated
                printf("You win, and your points are: %d\n", score); //print win
message and score achieved
            }
            else { //fourth argument - if none of the above arguments are achieved
                score = 0; //set score variable to 0
                printf("%d %d %d\n", random1, random2, random3); //print random
numbers generated
                printf("You lose, and your points are: %d\n", score); //print
lose message and score achieved
            }
            printf("Do you want to spin again?(Y/N)\n"); // ask user if they want to
play again
            scanf_s("%c", &spin); //store user response in the spin variable and if
the while loop argument is met program will run again
        }
    }
}

```

```
}  
  
void main()  
{  
    srand(time(NULL)); //seed random with system clock- do only ONCE  
    slotmachine(); //call slot machine function  
}
```

4.4 Testing

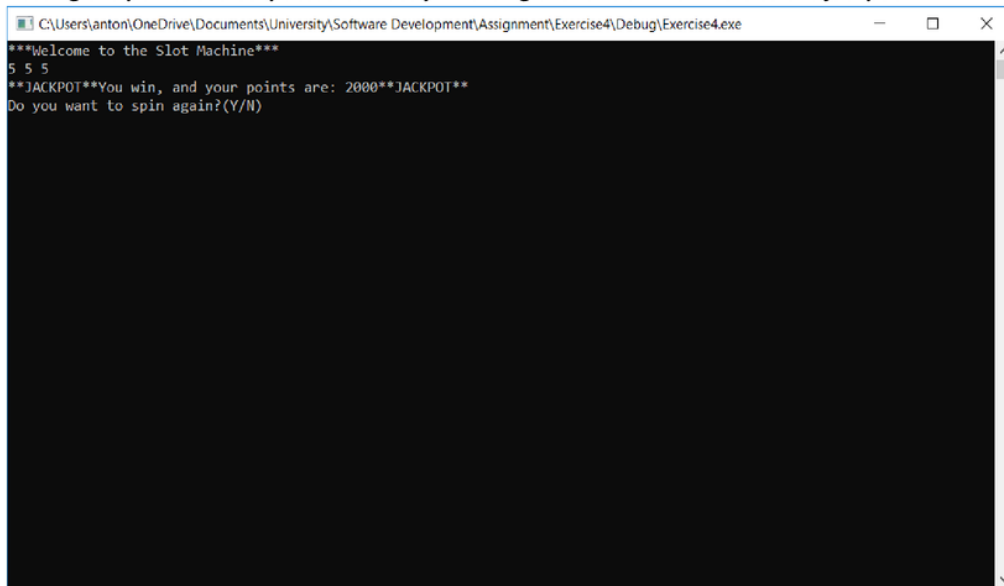
4.4.1 Normal



```
C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise4\Debug\Exercise4.exe  
***Welcome to the Slot Machine***  
4 4 3  
You lose, and your points are: 0  
Do you want to spin again?(Y/N)  
y  
2 3 5  
You lose, and your points are: 0  
Do you want to spin again?(Y/N)  
y  
5 5 2  
You win, and your points are: 100  
Do you want to spin again?(Y/N)  
y  
5 3 5  
You win, and your points are: 100  
Do you want to spin again?(Y/N)  
Y  
1 3 3  
You lose, and your points are: 0  
Do you want to spin again?(Y/N)  
Y  
5 2 3  
You lose, and your points are: 0  
Do you want to spin again?(Y/N)  
Y  
3 1 5  
You lose, and your points are: 0  
Do you want to spin again?(Y/N)
```

Results: Program runs as expected

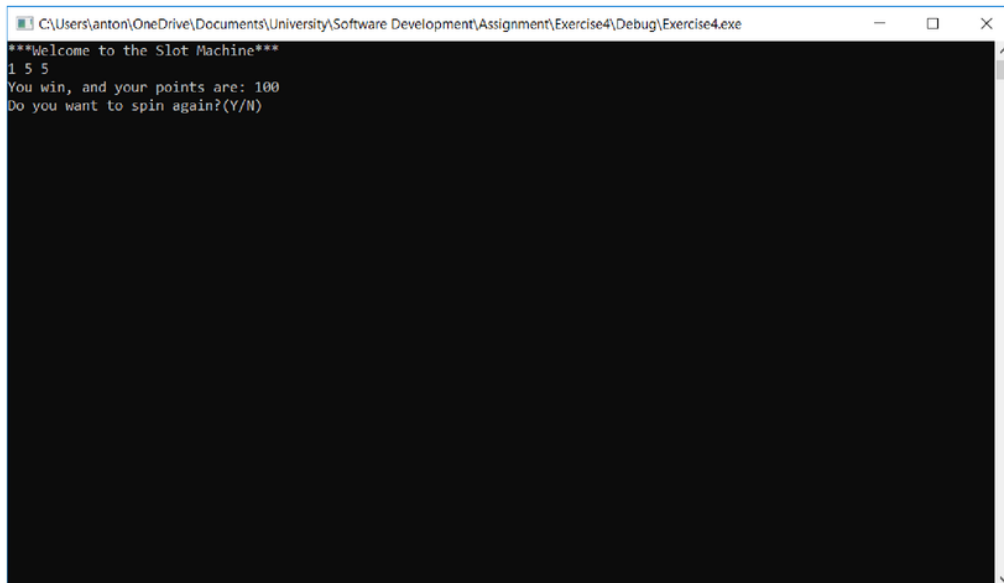
Testing the points are outputted correctly – Forcing the random numbers to the jackpot:



```
C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise4\Debug\Exercise4.exe
***Welcome to the Slot Machine***
5 5 5
**JACKPOT**You win, and your points are: 2000**JACKPOT**
Do you want to spin again?(Y/N)
```

Results: Program runs as expected

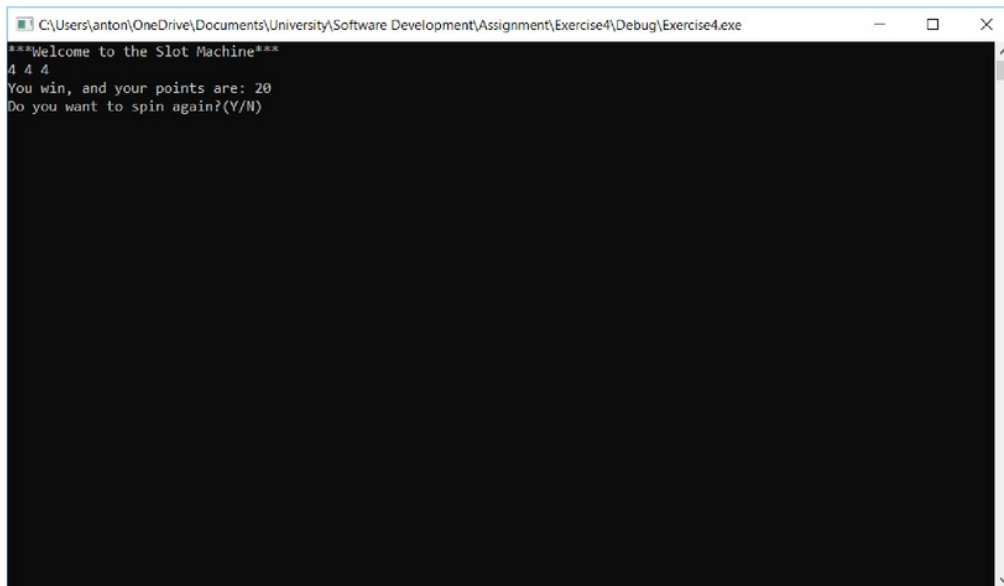
Forcing the random numbers to two 5s:



```
C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise4\Debug\Exercise4.exe
***Welcome to the Slot Machine***
1 5 5
You win, and your points are: 100
Do you want to spin again?(Y/N)
```

Result: Program runs as expected

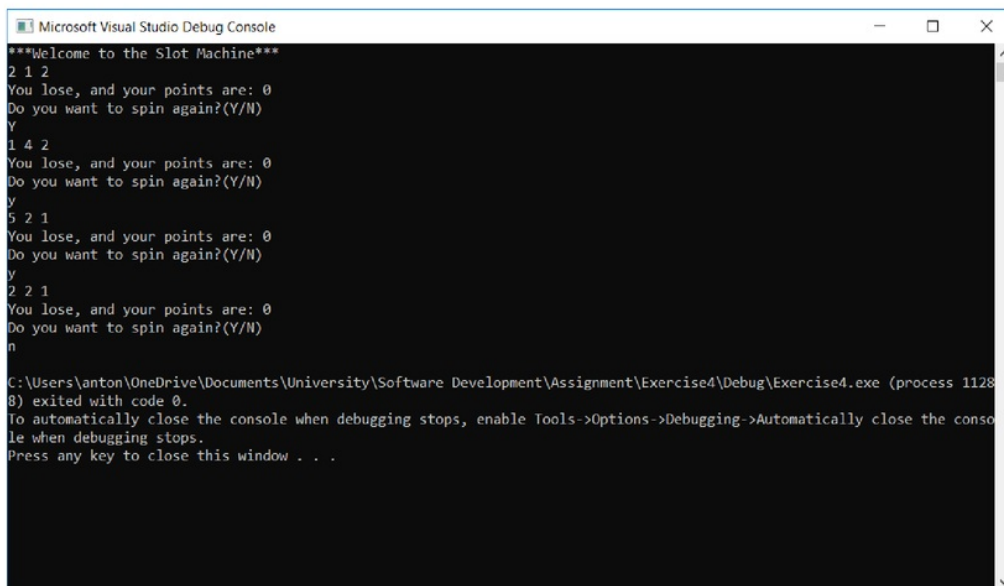
Forcing the random numbers to three of the same number:



```
C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise4\Debug\Exercise4.exe
***Welcome to the Slot Machine***
4 4 4
You win, and your points are: 20
Do you want to spin again?(Y/N)
```

Result: Program runs as expected

When the user chooses the end the game:



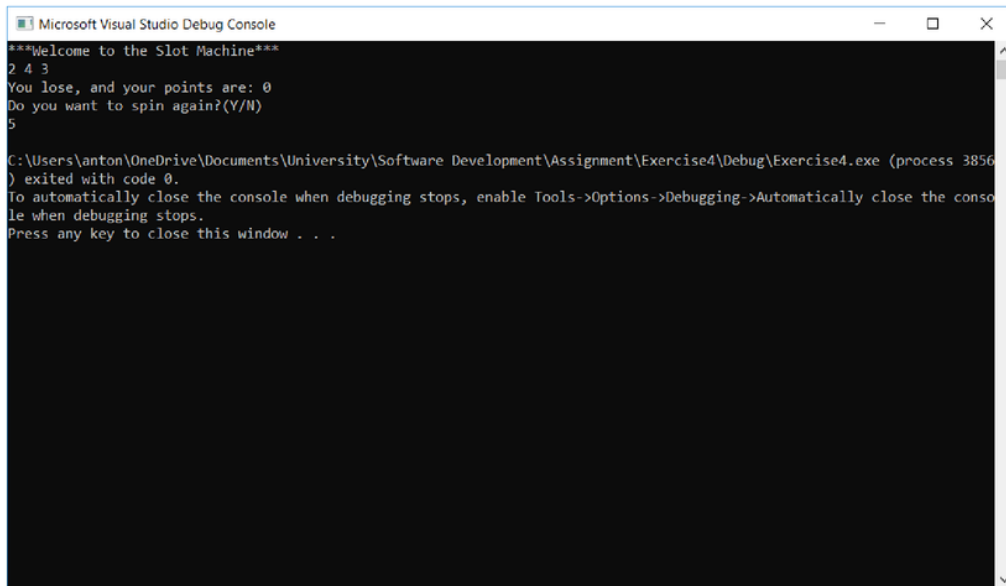
```
Microsoft Visual Studio Debug Console
***Welcome to the Slot Machine***
2 1 2
You lose, and your points are: 0
Do you want to spin again?(Y/N)
y
1 4 2
You lose, and your points are: 0
Do you want to spin again?(Y/N)
y
5 2 1
You lose, and your points are: 0
Do you want to spin again?(Y/N)
y
2 2 1
You lose, and your points are: 0
Do you want to spin again?(Y/N)
n

C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise4\Debug\Exercise4.exe (process 1128)
exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console
when debugging stops.
Press any key to close this window . . .
```

Result: Program runs as expected

4.4.2 Erroneous

When the user enters anything other than Y or N:



```
Microsoft Visual Studio Debug Console
***Welcome to the Slot Machine***
2 4 3
You lose, and your points are: 0
Do you want to spin again?(Y/N)
5
C:\Users\anton\OneDrive\Documents\University\Software Development\Assignment\Exercise4\Debug\Exercise4.exe (process 3856)
) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console
le when debugging stops.
Press any key to close this window . . .
```

Result: Program ends as expected

4.4.3 Recommendations

After testing this program meets the brief exactly. It allows the user to play as many times as they desire and gives them a random result each time. The improvements that could be made to the program is the user interface, it could be made more aesthetically pleasing and more obvious when the user wins the jackpot number of points.

Conclusion

This report has effectively demonstrated some of the functions available to a developer in the C programming language. The report is also reflective in that it illustrates how programs can be improved upon by adding minor changes to the source code, which often makes the program more reliable.

FINAL GRADE

/100

GENERAL COMMENTS

Instructor

Part A

Variables - Quite comprehensive. You could also mention that different variable types take up different amounts of space in memory 4.5 /5

Counting Loops - Good overall, the main points covered. Your example isn't quite right - there is no upper limit on the middle condition? $i \leq ?$ 4.5 /5

Arrays - You've covered the points, it would be nice to see a code example 4/5

Part B

Exercise 1

Code-Annotation - A bit limited, you should have the user input the values rather than hardcode them - this would also give you scope to think about issues of the user entering negative numbers etc. I'm not sure why as you have put this in your recommendation section? 2/5

Documentation A bit limited due to lack of input in the code You have used a negative number but I would not say that an age of -25 is correct! 2.5/5

Exercise 2

Code-Annotation A working solution produced, you need to think about negative numbers and the relative sizes of triangle and circle. 3.5/5

Documentation Good logical breakdown, you need to think about negative numbers and the relative sizes of triangle and circle. Your testing is correct and your recommendations are correct to, but you need to implement them (this is the whole point of testing!) 4 /5

Exercise 3

Code-Annotation - Good use of array/loop, you need to validation for negative numbers here. 4/5

Documentation - A bit too much code creeping into your logical breakdown - you don't need to talk about arrays and loops, this is an implementation decision. Just "ask user to enter 5 values is enough" In the testing, prices cannot be negative in real life, therefore this is not a correct result! 3.5/5

Exercise 4

Code-Annotation A good effort! Some inefficiencies in the very long if statements, some of these could have been combined (eg if you've already checked for a jackpot, you need only check whether the numbers are equal or not, we don't need to know what their values are, only that they are equal. No running total which is a shame, because you have implemented the while loop to repeat the program. 7.5 /10

Documentation Logical breakdown is a bit too codey, I would find it hard to write a program based on this. You need to write things like "Generate 3 random numbers between 1 and 5". You haven't mentioned the point scoring at all? Your pseudocode is also a bit close to C - you should be able to implement pseudocode in any language, so it doesn't make sense to include (for example) the prototype as many languages don't need them. Try to make it language-agnostic. Testing covers all eventualities I think, well done! 6.5/10

Total: 46.5/65 (72%)

PAGE 1
PAGE 2
PAGE 3
PAGE 4
PAGE 5
PAGE 6
PAGE 7
PAGE 8
PAGE 9
PAGE 10
PAGE 11
PAGE 12
PAGE 13
PAGE 14
PAGE 15
PAGE 16
PAGE 17
PAGE 18
PAGE 19
PAGE 20
PAGE 21
PAGE 22
PAGE 23

