

Brian Morris Programs in C and Documentation

by Brian Morris

FILE	11831_BRIAN_MORRIS_BRIAN_MORRIS_PROGRAMS_IN_C_AND_DOCUMENTATION_157877_104442862.DOCX (1.33M)		
TIME SUBMITTED	24-FEB-2019 06:37PM (UTC+0000)	WORD COUNT	3611
SUBMISSION ID	101107320	CHARACTER COUNT	21902

Programs in C and documentation

PARTS A AND B

BRIAN MORRIS (1809130)

Abstract

This report is in two parts the first part split into three sections. Firstly offers a brief definition of variables and discusses the use of variables in the C programming language. Secondly discusses the use of a counting For-Loop. Where its used and why one would look to implement such a loop as opposed to different looping methods. The final section briefly covers the use of Arrays within C programming.

The second part of this report documents four C programming solutions with annotated code listing, additional screen shoots of output produced during testing and screen shoots of errors outside of the running application.

Table of contents

Contents

Abstract	1
Part A Understanding.....	4
A1. What is a variable and why are they important in writing programs?	4
A2. Explain why the for-loop can also be referred to as a counting-loop and give an example of when you would use it.	4
A3. Explain what an array is, what their features are, and give an example of when you would use one	5
Part B.....	6
Exercise 1:	6
Exercise 1 Annotated Code	7
Exercises 1. Testing with normal values	8
Exercise 1. Testing with negative numbers.....	8
Exercise 1. Testing with boundary values	9
Exercise 1. Testing with impossible values	10
Exercise 2:	11
Exercise 2. Interpreted requirements.....	11
Exercise 2. Annotated Code	13
Exercises 2. Initial testing with normal values	14
Exercise 2. Testing the validation loops	15
Exercise 2. Testing with impossible values	16
Exercise 3:	18
Exercise 3. Annotated Code	19
Exercise 3. Testing with normal values	20
Exercise 3. Testing with extreme numbers.....	20
Exercise 3. Testing with negative values to enter the validation loop	21
Exercise 3. Testing with negative values to ensure the loop isn't tripped	22
Exercise 4:	23
Exercise 4. Requires the the following:-	24
Exercise 4. Annotated Code	25
Exercise 4 .Testing with valid input.....	27
Exercise 4. First points	27
Exercise 4. Another combination of 5s produce an additional score	27
Exercise 4. A different combination of 5s produce an additional score	28
Exercise 4. Three numbers in a row	28

Exercise 4. A different three numbers in a row	28
Exercise 4. Jackpot score.....	29
Exercise 4. Clean Exit.....	30
Exercise 4. Quitting the game on initial run with expected input	31
Exercise 4. Quitting the game with a non 'y' answer.....	31
Bibliography	32

Part A Understanding

A1. What is a variable and why are they important in writing programs?

A variable is a value that can be changed by a process, that is acted upon as information to feed into a process to affect an outcome. The variable is defined and labelled within a section of code to ensure a consistent area of physical memory can be referenced to store and retrieve information from a reserved memory location.

A variable is typically defined as a data type of the following :-

Integer a whole number declared using INT

Decimal a decimal number declared using FLOAT

Character an alpha numeric value declared as CHAR

String a collection of character declared as a CHAR[n] where n represents the number of characters required to be stored within the strings array plus 1.

There are other types of variables that can be used such as short int, long int and double to name a few however these variable types are not required or used in the solutions below.

A2. Explain why the for-loop can also be referred to as a counting-loop and give an example of when you would use it.

A for-loop is used when a specific value or condition needs to be met a specific amount of times and when met allow the process can continue as planned. This helps reduce the amount of code needed to be rewritten, allows for more flexibility in designing a solution and reduces the line count of the software ensuring the final code remains smaller. Code size is no too much of an issue with today's computers but when C was originally developed by Dennis Ritchie and his colleagues at Bell laboratories during the early 1970s [1] [2] it was a time when computers had severely limited memory and storage capacities compared to modern day systems.

In practice we could use a for loop to count a process a specific amount of times with the count being incremented by a value until conditions are met. The for-loop would cycle through the loop a set amount of times before continuing with further processing. As opposed to replicating the same part of the code a set amount of times until the exit condition exists.

Exercise 3 of this assignment contains an example of a known number count. The loop is set to count 5 times and the increment is used to address the memory space within an array to store prices and products

Conversely within the for loop a while loop has been utilized to check a condition of the data entered. If the condition is met the loop repeats indefinitely until a defined action until exit conditions are true.

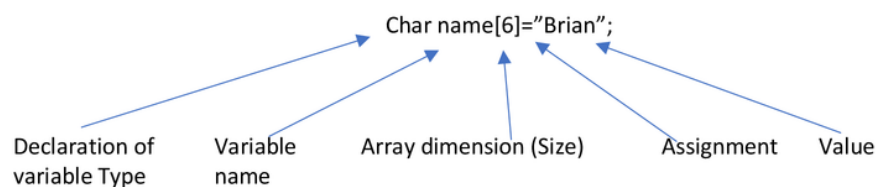
A3. Explain what an array is, what their features are, and give an example of when you would use one

An array is a useful way of storing multiple variable values of the same type within a specified area. Typically, we would also declare its type, identifying what data type the array structure was going to hold eg integer, character, decimal or string.

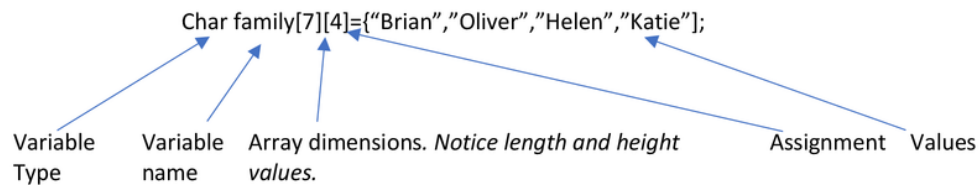
Strings within C are already a single dimensional array for example the string 'Brian' could be declared as follows:-

```
Char name[6]="Brian";
```

The above example would be broken down as follows



An array of strings would be a multi-dimensional array where the structure is two dimensional having both length and height for example:



Its possible for arrays to extend far beyond a 3rd dimension and the limits of a multi-dimensional array are limited only by the compiler of the software and the ability to manage and understand the data in arrays that big when diagnosing issues.

Part B

Exercise 1:

Show how variables can contain different types of values by declaring three variables that contain a name, a measure of height (e.g. 1.81cm) and a person's age (e.g. 32). Input these values and display them on the screen.

This exercise is requesting the following:-

1. Declare a string variable to be labelled 'Name'
2. Declare a floating point variable captured to two decimal places to be labelled 'Height'
3. Declare an integer variable to be labelled 'Age' .
4. Output the variables as required on the screen not defined in the original specification but could be displayed in a row or a column or as part of a sentence.

Exercise 1 Annotated Code

```
#include <stdio.h>

void main()
{
    /* Declare the variables we need for this code*/

    int Age; //Declare Age and an INteger
    char name[10]; //Declare String name with space for 9 characters
    float Height; //Declare floating point variable for Height

    printf("Please enter an age between 0-150:"); //Output to screen a
    request for user input
    scanf("%d", &Age); //capture an input and store the input as an
    integer in the declared variable Age
    while ((Age < 0) || (Age > 150)) //check int value (Age) outside a valid
    range while it its true repeat
    {
        printf("Please enter an age between 0-150:"); //Output to screen
        a request for user input
        scanf("%d", &Age); //capture an input and store the input as an
        integer in the declared variable Age
    }

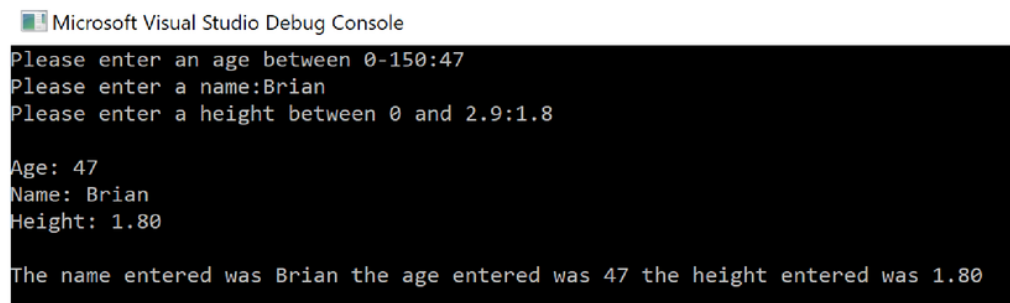
    printf("Please enter a name:"); //output to the screen a request for user input
    scanf(" %s", &name); // Capture a string variable 'name'

    printf("Please enter a height between 0 and 2.9:"); //output to the screen a
    prompt for user input
    scanf("%f", &Height); //Capture a floating point value
    while ((Height < 0) || (Height > 2.9000001)) //check floating point value and
    repeat whilst its true
    {
        printf("Please enter a height:"); //output to the screen a prompt for
        user input
        scanf("%f", &Height); //Capture a floating point value
    }

    // Output all the values as a separate line in a coloumn
    // note \n is an escape character to instruct the program to add a new line
    where ever it finds this
    printf("\nAge: %d\n", Age); //output the stored variables not \n is an escape
    character to instruct the rpogram to add a new line
    printf("Name: %s\n", name);
    printf("Height: %0.2f\n", Height); //Output the floting point value to two
    decimal spaces

    //Output the values again within a line of text
    printf("\nThe name entered was %s the age entered was %d the height entered was
    %0.2f\n\n\n", name, Age, Height);
}
```

Exercises 1. Testing with normal values



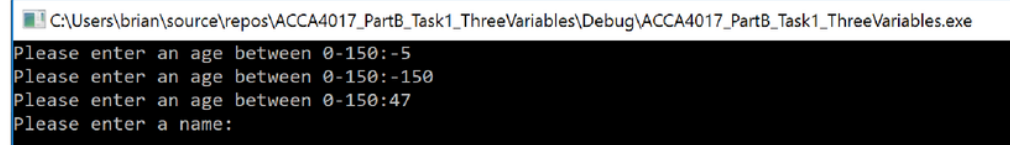
```
Microsoft Visual Studio Debug Console
Please enter an age between 0-150:47
Please enter a name:Brian
Please enter a height between 0 and 2.9:1.8

Age: 47
Name: Brian
Height: 1.80

The name entered was Brian the age entered was 47 the height entered was 1.80
```

All worked perfectly no issues here move along.

Exercise 1. Testing with negative numbers



```
C:\Users\brian\source\repos\ACCA4017_PartB_Task1_ThreeVariables\Debug\ACCA4017_PartB_Task1_ThreeVariables.exe
Please enter an age between 0-150:-5
Please enter an age between 0-150:-150
Please enter an age between 0-150:47
Please enter a name:
```

A while loop was included to check for a valid number input a number lower than 0 or greater than 150 will cause the system to loop until a 'valid' integer is input.

A similar loop was implemented for the height floating point variable as well.

Exercise 2:

Allow the user to enter the dimensions of shapes shown below (Figure 1), then calculate the area of the blue shaded part. Note: The triangular area in the centre is empty.

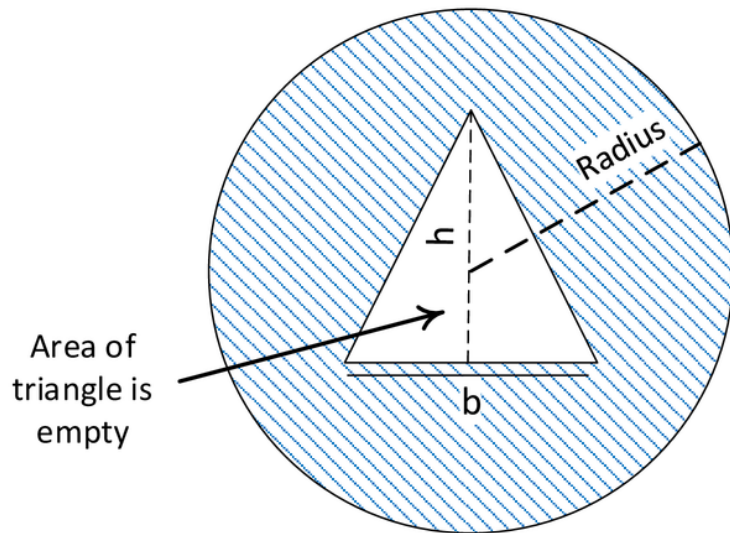


Figure 1 Calculate Shaded Area

Note:

Area of circle is given by $A = \pi r^2$

Area of triangle is given by $A = \frac{(b \times h)}{2}$

Exercise 2. Interpreted requirements

The exercises requires the following:-

1. It was originally assumed from the diagram that the Radius is the same value as the height of the triangle however after clarification its been confirmed it's a separate value. So the Radius needs to be entered as a separate floating point variable
2. While the base and height could be integer values they will be declared as floating point numbers.
3. The area will also be calculated as a decimal value I will format the decimal to 2 decimal places for readability.
4. Find the area of the triangle store that value in a float Area_Triangle
5. Find the area of the circle store that value in Area_Circle

6. Create a float variable to store $\text{Shaded_Area} = \text{Circle_Area} - \text{Area_Triangle}$

Pseudo code

Declare variables for height, base, radius

Enter radius

Enter base

Enter height

Calculate area of circle $\text{Pi} * (\text{Radius} * \text{Radius})$

Calculate area of Triangle $(\text{Base} * \text{Height}) / 2$

Calculate area of shaded area (circle-triangle)

Output results

After the initial plan additional items were incorporated to cover error checking loops on the values for Radius , Base length and height. An additional layer of validation was also added to ensure that the height of the tringle was not to large.

Whilst the value of the height variable could be larger than the diameter of the circle this height was validated against a calculation of the remaining shaded area the area of the triangle. If the result returned was a negative value the user would be prompted to re-enter a new value. This would repeat until the shaded are was not a negative value

Exercise 2. Annotated Code

```
#include <stdio.h>
#define PI 3.141592653; //declare the value of PI
void main()
{
    float Base,height,radius,Area_circle,Area_Triangle,Calculated_Area,new_height;
    //declare floating point variables for calculations
    radius = 0; //Set initial value for radius

    while (radius <= 0) //test value is less than or equal to zero, if false loop
until true
    {
        printf("\nPlease enter the radius of the circle:"); //Prompt user for
input for Radius
        scanf("%f", &radius); // Accept
user input
    }

    Base = 0; //set initial value for base
    while (Base <= 0) //test value is less than or equal to zero if false loop
until true
    {
        printf("\nPlease enter the length of the base of the triangle:");
//Prompt user for input for Triangle base
        scanf("%f", &Base);
        //Accept user input
    }

    while (radius <= Base) //loop through
until Base is an acceptable value ie equal or smaller than radius
    {
        printf("\nBase is too large please enter a smaller base value: ");
        scanf("%f", &Base);
    }
    Area_circle = (radius*radius)*PI; //Calculate the area of the
circle

    height = 0; //loop to
ensure valid entry of hight
    while (height<=0)
    {
        printf("\nEnter the height of the Triangle:"); //Prompt user for input
for Triangle Height
        scanf("%f", &height);
    }
    Area_Triangle = (Base*height) / 2; //Calculate
triangle area
    Calculated_Area = Area_circle - Area_Triangle; //calculate shaded area

    new_height = height;
    //Intialize variable prior to a conditional entry into a loop
    while (Calculated_Area < 0 || new_height > height) //Enter loop if shaded
calculated area is less than zero

    {
        //or new traingle height is larger than previous entry
        printf("\nTriangle too high please make it smaller: ");
        scanf("%f", &new_height);
    }
}
```

```

        while (new_height<=0)
        //loop to ensure positive number is entered
        {
            printf("\nPlease enter a valid height:");
            scanf("%f", &new_height);
        }
        Area_Triangle = (Base*new_height) / 2; //Recalculate
Area of the triangle
        Calculated_Area = Area_circle - Area_Triangle; //Recalculate the
shaded area
        height = new_height; //Reset
value of height to new height value
    }

    printf("\nThe area of the Triangle is %.2fcm squared\n\n", Area_Triangle);
//Output Area of the Triangle
    printf("\nThe area of the Circle is %.2fcm squared\n\n", Area_circle);
//Output the area of the triangle
    printf("\nThe area of your circle minus the area of the triangle is %.2fcm
squared\n\n", Calculated_Area); // Output the value of the shaded area
}

```

Exercises 2. Initial testing with normal values

Microsoft Visual Studio Debug Console

```

Enter the radius of the circle:4.5
Enter the length of the base of the triangle:3

Enter the height of the Triangle:5

The area of the Triangle is 7.50cm squared

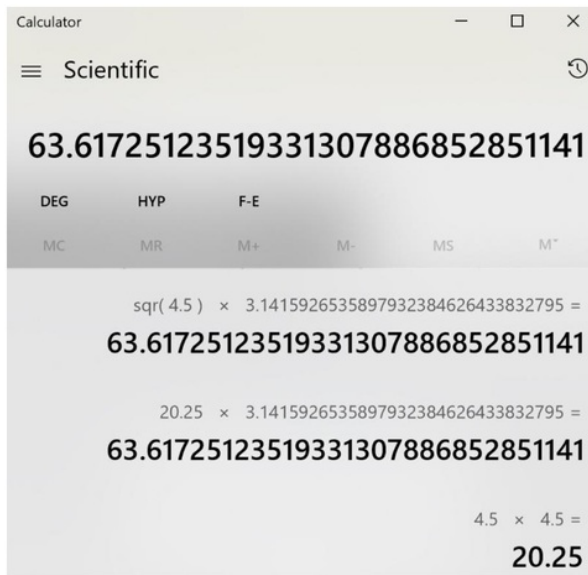
The area of the Circle is 63.63cm squared

The area of your circle minus the area of the triangle is 56.13cm squared

C:\Users\brian\source\repos\AreaofCircleBasedOnHeightofTriangle\x64\Debug\Are
ss 17600) exited with code 0.

```

Whilst manually calculating the values they were off slightly as the constant defined Pi as 3.142 even when accounting for rounding. The constant value was amended to a slightly more granular 3.141592653. Revised testing produced a result that mirrored the manual calculations.



```
Microsoft Visual Studio Debug Console

Please enter the radius of the circle:4.5
Please enter the length of the base of the triangle:3
Enter the height of the Triangle:5
The area of the Triangle is 7.50cm squared

The area of the Circle is 63.62cm squared

The area of your circle minus the area of the triangle is 56.12cm squared
```

Exercise 2. Testing the validation loops

A number of loops were added to validate input values against negative numbers and steer the input to produce a valid response to the answer for the shaded area. Without these loops it was possible to input negative lengths or produce a triangle that was larger than the circle resulting in a negative value for the shaded area.

Microsoft Visual Studio Debug Console

```
Please enter the radius of the circle:-9
Please enter the radius of the circle:0
Please enter the radius of the circle:9
Please enter the length of the base of the triangle:-9
Please enter the length of the base of the triangle:0
Please enter the length of the base of the triangle:8
Enter the height of the Triangle:-9
Enter the height of the Triangle:100
Triangle too high please make it smaller: -90
Please enter a valid height:62
The area of the Triangle is 248.00cm squared

The area of the Circle is 254.47cm squared

The area of your circle minus the area of the triangle is 6.47cm squared

C:\Users\brian\source\repos\AreaofCircleBasedOnHeightofTriangle\x64\Debug\Ass 7960) exited with code 0.
```

Exercise 2. Testing with impossible values

Entering non numerical values produce an expected failure: -

[illegible]

Exercise 3:

Write a program that allows the user to enter the price of 5 products into an array. Once all the prices are entered calculate the total price, and then calculate the total price plus 20% VAT.

Exercise 3. Sudo code

Sudo code

Define value of VAT

Declare an array for Price

Declare variables for price,total,total+vat ,counter

Declare a String Array for products //not required but I added anyway

Initialize array of products with data.

Initialize counter variable

Set total variable

Create counter loop for a 5 count

 Request prices

 Input prices

 Add input to total cost

Increment loop

Calculate vat costs

Output results

Exercise 3. Annotated Code

```
#include <stdio.h>
#define vat 1.2 //define constant for VAT
void main()
{
    float Price[5],total,total_plusvat; //declare float array for prices total and total+vat
    char Products[5][20] = { "Tin of Peas", "Frozen Chips", "4 Cans of Beer", "Ready made Curry", "Fresh Chicken"}; //declare string array for products

    int counter, i; //declare intergers for loop counter
    total = 0; //initialize variable
    for (counter = 0; counter < 5; counter++) //create a counted loop starting at 0 ending at 5 and incrementing by 1 each time its looped
    {
        printf("Please enter a price for %s: ", Products[counter]); //call string value by position in array
        scanf("%f", &Price[counter]);
        //store corresponding price in array
        while (Price[counter] < 0) //check that a non negative value is not entered
        {
            printf("\nPlease re-enter a valid price for %s: ", Products[counter]); //overwrite array value if required
            scanf("%f", &Price[counter]);
        }
        total = total + Price[counter]; //calculate total value by previous total value and price stored in Price[counter]
    }

    total_plusvat = total * vat; //calulate total inclusive of vat
    printf("\nThe total of the bill is:          %0.2f", total_plusvat);
    //output totals
    printf("\nThe total excluding vat is:      %0.2f", total);
    //output total ex vat
    printf("\nVat charge:                          %0.2f", total_plusvat - total);
    //vat charge
}
```

Exercise 3. Testing with normal values

Microsoft Visual Studio Debug Console

```
Please enter a price for Tin of Peas: .16
Please enter a price for Frozen Chips: 1.25
Please enter a price for 4 Cans of Beer: 3
Please enter a price for Ready made Curry: 4.99
Please enter a price for Fresh Chicken: 3.99

The total of the bill is: 16.07
The total excluding vat is: 13.39
Vat charge: 2.68
```

Exercise 3. Testing with extreme numbers

Testing with large numbers didn't crash the system but did produce issues with rounding producing an inflated price;

Microsoft Visual Studio Debug Console

```
Please enter a price for Tin of Peas: 99999999
Please enter a price for Frozen Chips: 99999999
Please enter a price for 4 Cans of Beer: 99999999
Please enter a price for Ready made Curry: 99999999
Please enter a price for Fresh Chicken: 99999999

The total of the bill is: 600000000.00
The total excluding vat is: 500000000.00
Vat charge: 100000000.00
```

Exercise 3. Testing with negative values to enter the validation loop

 Microsoft Visual Studio Debug Console

```
Please enter a price for Tin of Peas: -0.16

Please re-enter a valid price for Tin of Peas: .16
Please enter a price for Frozen Chips: -2.99

Please re-enter a valid price for Frozen Chips: -2

Please re-enter a valid price for Frozen Chips: 2.99
Please enter a price for 4 Cans of Beer: 79
Please enter a price for Ready made Curry: 4
Please enter a price for Fresh Chicken: -9

Please re-enter a valid price for Fresh Chicken: 1.679

The total of the bill is:                105.39
The total excluding vat is:            87.83
Vat charge:                           17.57
```

Exercise 3. Testing with negative values to ensure the loop isn't tripped

 Microsoft Visual Studio Debug Console

```
Please enter a price for Tin of Peas: -1

Please re-enter a valid price for Tin of Peas: -1

Please re-enter a valid price for Tin of Peas: -1

Please re-enter a valid price for Tin of Peas: -1

Please re-enter a valid price for Tin of Peas: -1

Please re-enter a valid price for Tin of Peas: -1

Please re-enter a valid price for Tin of Peas: -1

Please re-enter a valid price for Tin of Peas: -1

Please re-enter a valid price for Tin of Peas: 1
Please enter a price for Frozen Chips: 1
Please enter a price for 4 Cans of Beer: 1
Please enter a price for Ready made Curry: 1
Please enter a price for Fresh Chicken: 1

The total of the bill is:                6.00
The total excluding vat is:            5.00
Vat charge:                            1.00
```

Exercise 4:

Build a fruit machine game which represents the 3 reels of symbols using numbers ranging from 1 to 5. Allow the user to have as many goes as they like (the numbers are chosen randomly each time).



The value of 5 will be treated as the “Jackpot” symbol. The scoring will be as follows:

- Any three of the same numbers (excluding fives) will pay a bonus of 20 points
e.g.
4 4 4
3 3 3
1 1 1, etc.
- Any combination of two fives will pay out 100 points e.g.
5 5 3
5 4 5
5 5 1, etc.
- Three fives pay out the big jackpot of 2000 points.

The points the player has scored should be displayed and the player should be asked if they wish to make another spin. It could look something like this:

Non-cumulative score version

Cumulative score version

```
Spin? Y
2 2 2
You win, and your points are 100
Spin? Y
1 5 4
You lose, and your points are 0
Spin? Y
5 5 5
You win, and your points are 2000
```

```
Spin? Y
2 2 2
You win, and your points are 100
Spin? Y
1 5 4
You lose, and your points are 100
Spin? Y
5 5 5
You win, and your points are 2100
```

Exercise 4. Requires the the following:-

- 1.Include libraries to assist in generating numbers
- 2.Declare integer variables for the numbers
- 3.Declare integer variables for the extremes of the numbers
- 4.Declare Char variable for yes no response
- 5.Declare and initialize integer variable for the running total
- 6.Create a seed for random number
- 7.Ask user to play
 - 7.1 Begin while choice is yes loop
 - 7.2 Generate three separate numbers
 - 7.3 Check for winning numbers
 - 7.4 Add to running total
 - 7.5 Ask user to play until loop is broken
8. Output final running score

Exercise 4. Annotated Code

```
#include <stdio.h>
#include <stdlib.h> //include additional libraries
#include <time.h>   //include additional libraires

void main()
{
    int rand_1,rand_2,rand_3;           //declare three interger variables for random
    numbers 1-3
    int low = 1;                       //declare interger and set value for
    lowest number in the range
    int high = 5;                      //declare interger variable and set
    highest number in the range
    char choice;                       //decalre a Character variable for
    user Yes/No Response
    int score = 0;                     //declare and initialize
    interger for running score total
    srand(time(NULL));                 //seed randomness

    printf("SPIN?");
    scanf(" %c", &choice);

    while (choice == 'y')
    {
        // printf("\nYour Spin results are");
        rand_1 = rand() % (high - low + 1) + low; //get random number 1
        rand_2 = rand() % (high - low + 1) + low; //get random number 2
        rand_3 = rand() % (high - low + 1) + low; //get random number 3
        printf("%d" " " "%d" " " "%d\n", rand_1,rand_2,rand_3); //Output result

        if (rand_1 == 5 && rand_2 == 5 && rand_3 == 5)
        {
            score = score + 2000;
            printf("You win, and your points are %d \n", score);
        }
        else if (rand_1 == 5 && rand_2 == rand_1)
        {
            score = score + 20;
            printf("You win, and your points are %d \n", score);
        }
        else if (rand_2 == 5 && rand_2 == rand_3)
        {
            score = score + 20;
            printf("You win, and your points are %d \n", score);
        }
        else if (rand_3 == 5 && rand_3 == rand_1)
        {
            score = score + 20;
            printf("You win, and your points are %d \n", score);
        }
        else if (rand_1 != 5 && rand_2 == rand_1 && rand_1 == rand_3)
        {
            score = score + 100;
            printf("You win, and your points are %d \n", score);
        }

        else if (rand_1 != rand_2)
        {
            score = score + 0;
        }
    }
}
```

```

        printf("You lose, your points are %d \n", score);
    }
    else if (rand_2 != rand_3 )
    {
        score = score + 0;
        printf("You lose, your points are %d \n", score);
    }
    else if (rand_1 != rand_3)
    {
        score = score + 0;
        printf("You lose, your points are %d \n", score);
    }
    else if (rand_1 != rand_2)
    {
        score = score + 0;
        printf("You lose, your points are %d \n", score);
    }
    else if (rand_2 != rand_3)
    {
        score = score + 0;
        printf("You lose, your points are %d \n", score);
    }

    // printf("Your Running score is: %d \n", score);
    printf("SPIN?");
    scanf(" %c", &choice);
}
printf("\nAnother time, maybe? Bye!\n\n");
printf("\nYour final score was: %d \n", score);
}

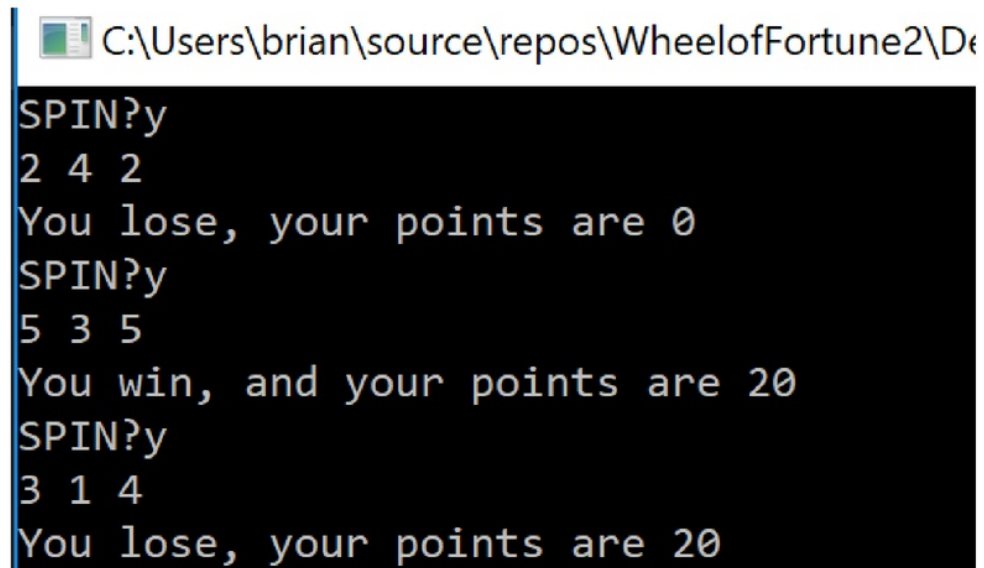
```

Exercise 4 .Testing with valid input

Exercise 4. First points

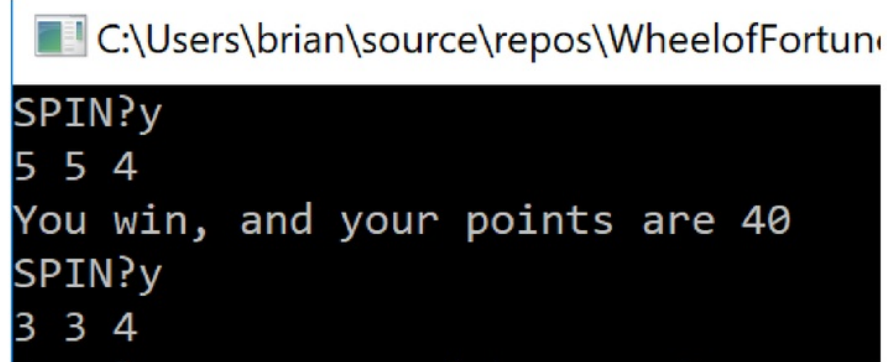
Screen shots are from one game.

0 points from first spin but second spin has two 5s so adds 20 points to the score.



```
C:\Users\brian\source\repos\WheelofFortune2\De
SPIN?y
2 4 2
You lose, your points are 0
SPIN?y
5 3 5
You win, and your points are 20
SPIN?y
3 1 4
You lose, your points are 20
```

Exercise 4. Another combination of 5s produce an additional score



```
C:\Users\brian\source\repos\WheelofFortune
SPIN?y
5 5 4
You win, and your points are 40
SPIN?y
3 3 4
```

Exercise 4. A different combination of 5s produce an additional score

```
C:\Users\brian\source\repos\WheelofFortu
You lose, your points are 40
SPIN?y
4 5 5
You win, and your points are 60
SPIN?y
```

Exercise 4. Three numbers in a row

Further on into the game a run of three numbers but not three fives produce a win and add to the running total.

```
C:\Users\brian\source\repos\WheelofFortu
SPIN?y
3 3 3
You win, and your points are 280
SPIN?y
```

Exercise 4. A different three numbers in a row

A run of three different numbers but not three fives add to the running total

```
C:\Users\brian\source\repos\WheelofFortu
You lose, your points are 540
SPIN?y
1 1 1
You win, and your points are 640
SPIN?y
```

Another three number win

 C:\Users\brian\source\repos\WheelofFortun

```
1 5 5
You win, and your points are 960
SPIN?y
4 4 4
You win, and your points are 1060
SPIN?y
```

Exercise 4. Jackpot score

Finally, a Jackpot win.

 C:\Users\brian\source\repos\WheelofFortu

```
4 1 2
You lose, your points are 1280
SPIN?y
5 5 5
You win, and your points are 3280
SPIN?
```

Exercise 4. Clean Exit

Quitting the game and outputting the final score



Microsoft Visual Studio Debug Console

```
You lose, your points are 3300
SPIN?y
5 2 1
You lose, your points are 3300
SPIN?y
4 4 4
You win, and your points are 3400
SPIN?y
4 2 3
You lose, your points are 3400
SPIN?y
3 4 3
You lose, your points are 3400
SPIN?y
2 5 2
You lose, your points are 3400
SPIN?n

Another time, maybe? Bye!

Your final score was: 3400
```

Exercise 4. Quitting the game on initial run with expected input

 Microsoft Visual Studio Debug Console
SPIN?
n
Another time, maybe? Bye!
Your final score was: 0

Exercise 4. Quitting the game with a non 'y' answer

 Microsoft Visual Studio Debug Console
SPIN?
y
2 5 2
You lose, your points are 0
SPIN?
1
Another time, maybe? Bye!
Your final score was: 0

Bibliography

- [1] K. Pal, "The History of the C Programming Language," [Online]. Available: <https://www.techopedia.com/the-history-of-the-c-programming-language/2/32996>. [Accessed 16 January 2019].
- [2] D. M. Ritchie, "The Development of the C Language*," [Online]. Available: <https://www.bell-labs.com/usr/dmr/www/chist.html>. [Accessed 16 January 2016].

Brian Morris Programs in C and Documentation

GRADEMARK REPORT

FINAL GRADE

/100

GENERAL COMMENTS

Instructor

Part A

Variables A good explanation, you could also have mentioned that different variable types take up different amounts of memory 4.5/5

Counting Loops - A good explanation and contrast with the while loop, include a coded example here 4/5

Arrays - A good explanation, don't forget to state that arrays in most languages start from 0 - 4/5

Part B

Exercise 1

Code-Annotation - I'm not sure that anyone lives much beyond 120, but a good answer with suitable validation 4/5

Documentation - In the breakdown you haven't stated that the user is entering the values or anything about the validation? I'm not sure why because you have gone on to code it. A wide variety of tests included. 4/5

Exercise 2

Code-Annotation - A good implementation with attention to validation. The only point to note is that the code is slightly confused on the height of

triangle validation and not entirely efficient. 4.5/5

Documentation - Include the validation in the step breakdown (although you have included it in the explanatory notes) Testing is fairly comprehensive 4/5

Exercise 3

Code-Annotation - A good implementation with suitable validation, nothing to fault! 5/5

Documentation Good breakdown, don't forget to state that the user is entering prices and number the stages for easy reading. Testing shows a wide range. 4.5/5

Exercise 4

Code-Annotation - A good effort with a running total implemented. Some parts of your if statements are redundant, and some could have been combined into a for loop for counting the number of 5s. Nevertheless, a good implementation overall 8.5/10

Documentation - You don't need to refer to including libraries, create a seed for random number etc, just the logic of the program. "Generate a random number" is enough here. You also need to include the win scenarios, they are part of the logic of the program. I think you've shown all the possibilities with your testing, well done! 8/10

Total: 55/65 (85%)

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

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29
