

Programs in C and documentation (M4X01508)

by Jaromir Pauer

FILE	11677_JAROMIR_PAUER_PROGRAMS_IN_C_AND_DOCUMENTATION__M4X01508__173935_1050903537.DOCX (411.21K)		
TIME SUBMITTED	16-APR-2019 02:11PM (UTC+0100)	WORD COUNT	4046
SUBMISSION ID	104857219	CHARACTER COUNT	24599

University of Wales Trinity Saint David

School of Applied Computing

Foundation Computing

Jaromír Pauer (1801641)

Programs in C and documentation (M4X01508)

Contents

Part A.....	3
A1. What is a variable and why are they important in writing programs?.....	3
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.	3
A3. Explain what an array is, what their features are, and give an example of when you would use one.	4
Part B.....	7
Exercise 1:	7
Identifying problems in the exercise:	7
Flowchart for exercise 1:.....	7
This is the C code for exercise 1:.....	8
Screenshot showing that the programme works as intended.....	9
Exercise 2:	10
Identifying problems in the exercise:	10
Flowchart for exercise 2:.....	11
This is the C code for exercise 2:.....	11
Screenshots showing that the programme works as intended	14
Exercise 3:	15
Identifying problems in the exercise:	15
Flowchart for exercise 3:.....	16
This is the C code for exercise 3:.....	16
Screenshot showing that the programme works as intended.....	17
Exercise 4:	18
Identifying problems in the exercise:	18
Flowchart for exercise 4:.....	19
.....	19
The C code for exercise 4:.....	20
Screenshots showing that the programme works as intended	22

Part A.

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

Essentially, variables are used in programming to store information, which can then be accessed and manipulated. Generally, [1] the variable consists of a type of variable (such as: integer, float, char and string); variable name; data location; and the value of the variable. An example of a variable is: float myNumber = 0; float would be the type of variable, myNumber would be the variable name and 0 would be the value of that variable. There are multiple reasons why variables are important in programming, for example, they can help us organize our code in a certain way that is familiar not only to us but also to other programmers. It also makes it easier to return to the programme at a later stage. Therefore, it is essential to be consistent throughout the code while using, for example, variable names.

Additionally, there are certain rules and naming conventions that it is recommended that users follow. For example, [2] variable name can only begin with a letter, an underscore and a digit. Additionally, spaces between words are not allowed. Alternatively, it is possible to use camel case style, for instance, variableName. [2] Constants are typically in uppercase letters. [2] Finally, there are some other things to remember such as some words are reserved as keywords and these can't be used as variable names. For example, int, float, if, else, while, for, do, break.

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.

[3] A For-loop, sometimes known as counting-loop, is a variable that performs predefined tasks while or until a predetermined condition is met. In other words, the variable takes on a range of integer values in some orderly sequence until a limit is reached, for example starting at 0 and increasing in increments of 1 until the limit of 10 reached. [4] A for-loop syntax can typically read: for(initialiser ; test-expression ; incrementer) { statements }.

An example of a for-loop (counting loop).

```
#include <stdio.h>

void main()
{
    int i;

    for (i=0; i < 10; i++)
    {
        printf("i is:%d\n", i);
    }
}
```

In this example above the initialiser is set to i; i is then given an initial value of 0, the test expression is set to i being less than 10 in which case the incrementer is set to + and will increase by 1 until the condition is met, after which the loop will end.

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

[5] An array is a data structure that can store a fixed sized sequential collection of elements of the same type. In an array the data items, all of the same type, are accessed using a common name. [6] A one dimensional array is like a list; whereas a two-dimensional array is like a table. The features of an array include: a structure that holds elements that have the same data type; the elements of an array are stored in subsequent memory locations; two-dimensional array elements are stored row by row in subsequent memory locations, and the array name represents the address of the starting element. [7] An array size should be mentioned in the declaration and the size must be a constant expression and not a variable.

An example using 10 variables individually:
#include <stdio.h>

```
void main()
{
```

```
int number1, number2, number3, number4, number5, number6, number7, number8,
number9, number10;
number1 = 1, number2 = 2, number3 = 3, number4 = 4, number5 = 5, number6 = 6,
number7 = 7, number8 = 8, number9 = 9, number10 = 10;
```

```
    printf("number1: %d\n", number1);
    printf("number2: %d\n", number2);
    printf("number3: %d\n", number3);
    printf("number4: %d\n", number4);
    printf("number5: %d\n", number5);
    printf("number6: %d\n", number6);
    printf("number7: %d\n", number7);
    printf("number8: %d\n", number8);
    printf("number9: %d\n", number9);
    printf("number10: %d\n", number10);
}
```

It is evident that the above example looks cumbersome and that such an approach is not especially practical or efficient, particularly if large numbers have to be stored. Instead it would be better to adopt an array approach.

The example below uses an array to store 10 numbers:

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

```
void main()
```

```
{
```

```
    int number[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }, i; // declare array
```

```
    for (i = 0; i < 10; i++) // for loop set to go all the way to number10 and then stop, as the  
    conditions specified will be met.
```

```
    {
```

```
        printf("%d \n", number[i]); // display the values throughout the code execution
```

```
    }
```

```
}
```

References for Part A:

- [1] "What is a variable in programming? – Definition from Techopedia", [Techopedia.com](https://www.techopedia.com/definition/25647/variable-programming), N/a, [Online] Available at: <https://www.techopedia.com/definition/25647/variable-programming> [Accessed: 16/03/2019]
- [2] Identifier/Variable naming conventions in C language [Rules and Recommendations] - IncludeHelp", [Includehelp.com](https://www.includehelp.com/c/identifier-variable-naming-conventions.aspx), N/a, [Online] Available at: <https://www.includehelp.com/c/identifier-variable-naming-conventions.aspx> [Accessed: 25/03/2019]
- [3] "What is for", Computerhope, Computerhope.com , N/a [Online] Available at: <https://www.computerhope.com/jargon/f/for.htm> [Accessed: 28/03/2019]
- [4] M. McGrath ,C programming in easy steps, 4th ed. Computer step, 2002.
- [5] "Computer programming arrays", [Tutorialspoint.com](https://www.tutorialspoint.com/computer_programming/computer_programming_arrays.htm), N/a [Online] Available at: https://www.tutorialspoint.com/computer_programming/computer_programming_arrays.htm [Accessed: 16/03/2019]
- [6] "C Programming Course Notes – Arrays", Department of Computer Science, College of Engineering, [Online] Available at: https://www.cs.uic.edu/~jbell/CourseNotes/C_Programming/Arrays.html [Accessed: 28/03/2019]
- [7] D. Thakur, "Characteristics of Arrays in C, Ecomputer Notes, Ecomputernotes.com, [Online] Available at: <http://ecomputernotes.com/what-is-c/array/what-are-the-characteristics-of-arrays-in-c> [Accessed: 28/03/2019]

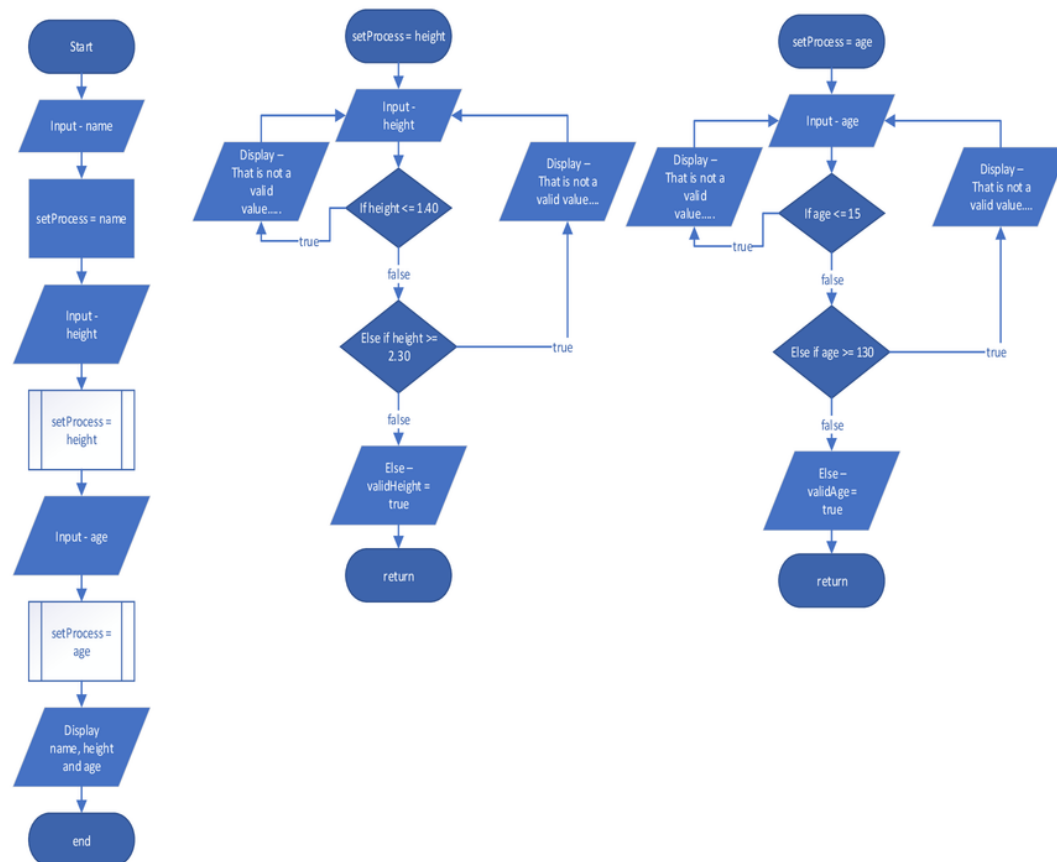
Part B.

Exercise 1:

Identifying problems in the exercise:

This exercise requires the user to input a person's name, height and age. For the name a string is used with at least 20 characters to avoid errors, especially when the user has a long name. For height use is made of a float with 2 decimal places, which allows the programme to record accurately the values entered by the user. Additionally, with the help of a bool command it is possible to check the validity of the height value entered so ensuring that only realistic values are input. In this case a decision was made to set the minimum height at 1.40 centimetres and the maximum height at 2.30 centimetres. Again, to avoid inaccuracies an age integer is used simply because there are only whole numbers to work with in this exercise. However, it remains important to set limits for the age values entered. Again, a bool command is useful to minimise the user entering invalid age. In this case the max low is set at 15 years of age and the max high is set at 120 years of age. The values of this exercise are depicted in the flowchart and the c code below:

Flowchart for exercise 1:



This is the C code for exercise 1:

```
#include <stdio.h> // standard input and output
#include <string.h> // Need to include that ( not finished)

void main()
{
    int age = 0; // integer age will be sufficient here
    float height = 0; // float height, using float there instead of integer because
    of numbers with decimal places.
    bool validHeight = false; //check for height range validity
    bool validAge = false; //check for age range validity
    char myName[20]; // leaving 20 characters for name, just in case the user has a
    long name to avoid errors.

    printf("Please enter your name: "); // user input for name
    scanf("%s", &myName); // scanf of the entered value

    while (validHeight != true) //while loop check for validity of height
    initialization
    {
        printf("\nPlease enter your height in centimetres: "); // user input for
        height
        scanf("%f", &height); // scanf of the entered value

        if (height <= 1.40) //if statement to check for value entered being or
        not being less or equal to 1.40
        {
            validHeight = false;
            printf("\nThat is not a valid value, Please try again!");
        }
        else if (height >= 2.30) //else if statement to check for value entered
        being or not being more or equal to 2.30
        {
            validHeight = false;
            printf("\nThat is not a valid value, Please try again!");
        }
        else // else statement to end the loop when the conditions are met.
        {
            validHeight = true;
        }
    }

    while (validAge != true) //while loop check for validity of age initialization
    {

        printf("\nPlease enter your age: "); // user input for age
        scanf("%d", &age); // scanf of the entered value
```

```

    if (age <= 15) // if statement to check for value entered being less or
equal to 15
    {
        validAge = false;
        printf("\nThat is not a valid value, Please try again!");
    }

    else if (age >= 120) // else if statement to check for value entered
being more or equal to 120
    {
        validAge = false;
        printf("\nThat is not a valid value, Please try again!");
    }

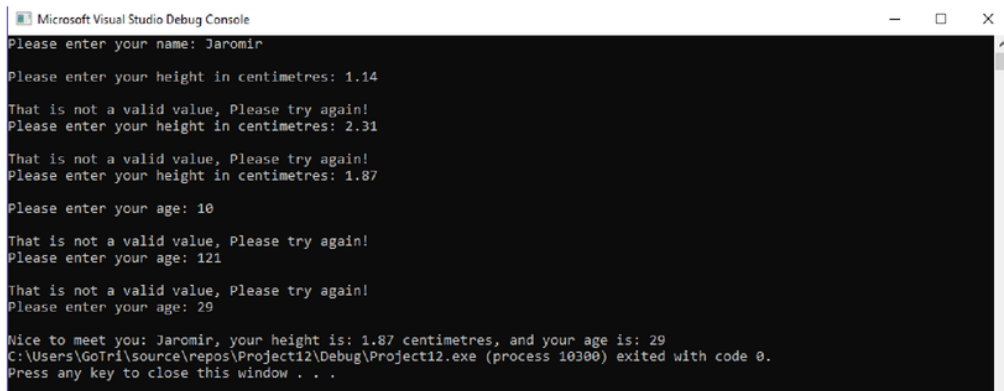
    else // else statement to end the loop when the conditions are met.
    {
        validAge = true;
    }

}
printf("\nNice to meet you: %s, your height is: %.2f meters, and your age is:
%d", myName, height, age); // display the values that were entered
}

```

Screenshot showing that the programme works as intended

The screenshot below shows the user entering name, then the user tries to enter too small height, and is prompted to enter the height again, upon trying to enter too big value he is prompted to enter again, after which the programme accepts a valid value entered. The same process is repeated for the age. Finally, the programme shows all the values entered by the user in a logical order.



```

Microsoft Visual Studio Debug Console
Please enter your name: Jaromir
Please enter your height in centimetres: 1.14
That is not a valid value, Please try again!
Please enter your height in centimetres: 2.31
That is not a valid value, Please try again!
Please enter your height in centimetres: 1.87
Please enter your age: 10
That is not a valid value, Please try again!
Please enter your age: 121
That is not a valid value, Please try again!
Please enter your age: 29
Nice to meet you: Jaromir, your height is: 1.87 centimetres, and your age is: 29
C:\Users\GoTri\source\repos\Project12\Debug\Project12.exe (process 10300) exited with code 0.
Press any key to close this window . . .

```

Exercise 2:

Identifying problems in the exercise:

The user is required to enter the width of the triangle, the height of the triangle and the radius of the circle. After that the program should calculate the area of the circle and the area of the triangle.

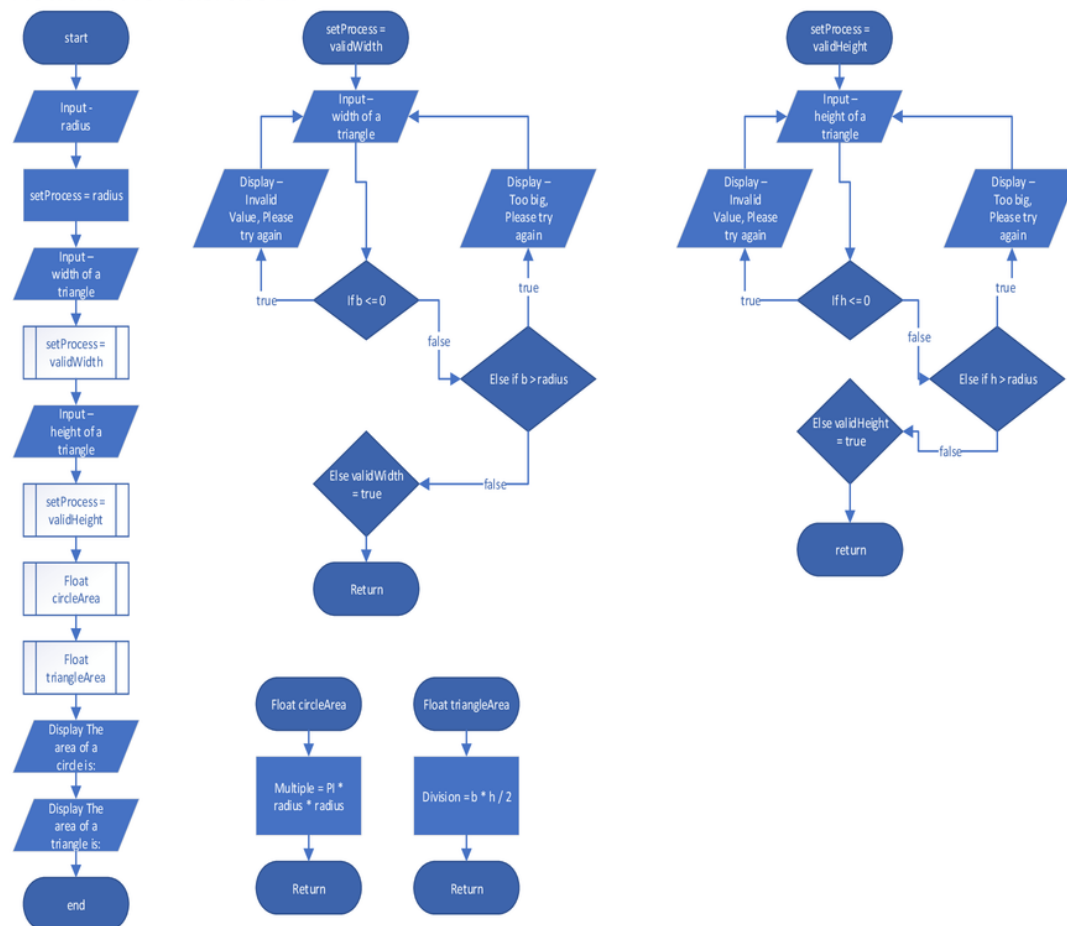
A number of problems may arise when completing this exercise such as:

1. It is necessary to ensure that the values of the width entered by the user should never be larger than the radius of the circle. It is essential to include a validity check of these matters in the C code.

2. The values of height entered by the user should not be larger than the radius of the circle. However, it can be equal to the size of a radius. Again, it is necessary to include a validity check of these values in the C code

3. The programme should be capable of validating the width and height entered by the user. To avoid complications the programme needs to ask the user to input the value of the radius first. Otherwise, the programme would not be able to check the values entered for the width and height.

Flowchart for exercise 2:



This is the C code for exercise 2:

```

#include <stdio.h> // standard input and output library
#include <math.h>
#define PI 3.142; // definition of a constant in this code as this value will be
constant throughout the code
float circleArea(float); //float statement for calculation that will be later used to
measure the area of the circle
float triangleArea(float b, float h); //float statement for calculation that will be
later used to measure the area of the triangle
bool validWidth = false; // bool statement to check the validity of the width, with
conditions specified in the code
bool validHeight = false; // bool statement to check the validity of the height, with
conditions specified in the code

float main()
{

```

```
float b, h, radius, multiple, division; // float of all the variables that will  
be used in this code
```

```
printf("Please enter the radius of the circle:"); // user input of the radius  
scanf("%f", &radius); // store the value entered by the user in the memory
```

```
while (validWidth != true)  
{  
the width    printf("\nPlease enter the width of the triangle:"); // user input of  
scanf("%f", &b); // store the value entered by the user in the memory  
if (b <= 0)  
{  
    printf("\nInvalid value, Please try again");  
    validWidth = false;  
}  
else if (b > radius)  
{  
    printf("\nToo big, Please try again");  
    validWidth = false;  
}  
else  
{  
    validWidth = true;  
}  
}
```

```
while (validHeight != true)  
{  
the height   printf("\nPlease enter the height of the triangle:"); //user input of  
scanf("%f", &h); // store the value entered by the user in the memory
```

```
if (h <= 0)  
{  
    printf("\nInvalid value, Please try again");  
    validHeight = false;  
}
```

```
else if (h > radius)  
{  
    printf("\nToo big, Please try again");  
    validHeight = false;  
}
```

```
else  
{  
    validHeight = true;  
}
```

```
}
```

```
multiple = circleArea(radius); // identification of result, that was calculated  
outside the main of the C code  
printf("\n\tThe area of a circle is: %.2f", multiple); // display the  
calculated value
```

```

        division = triangleArea(b, h); // identification of a result that was
        calculated outside the main of the C code
        printf("\n\tThe area of a triangle is: %.2f", division); // display the
        calculated value

    }

    float circleArea(float radius) // float outside the main of the C code, to calculate
    the area of the circle.
    {
        float multiple; // initialization of the float multiple to do the calculation
        in the next step

        multiple = 3.142 * radius * radius; // calculation using mathematical formula
        with the values that were entered by the user.
        return multiple; //return the result to the main part of the C code
    }

    float triangleArea(float b, float h) // float outside the main of the C code, to
    calculate the area of the triangle.
    {
        float division; // initialization of the float division to do the calculation
        in the next step

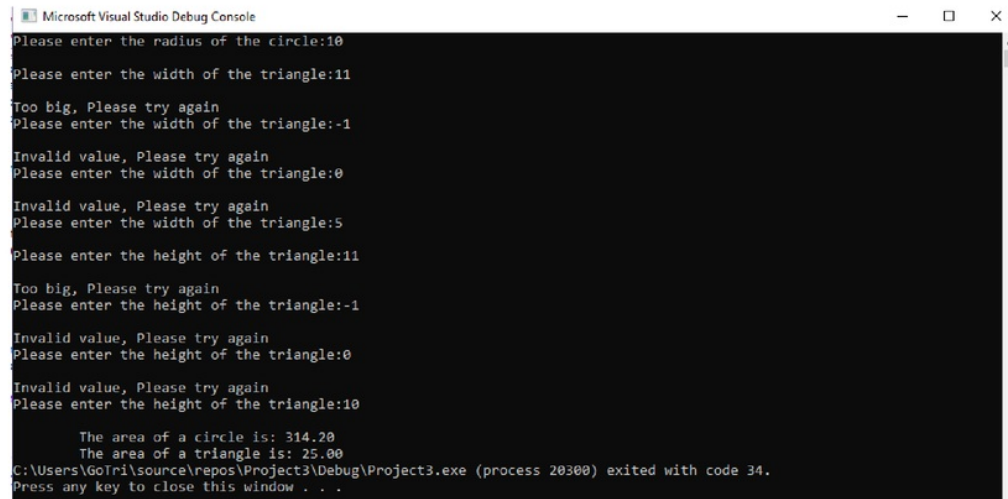
        division = b * h / 2;
        return division; // return the result to the main part of the C code
    }

```

Screenshots showing that the programme works as intended

Screenshot 1:

The screenshot below shows the user attempting to enter either too small or too big width of a triangle, in which case he is prompted to enter the value again. The same process is repeated for height. When valid values are entered the programme continues on and calculates the area of a circle and the area of a triangle and shows the calculated values for each.

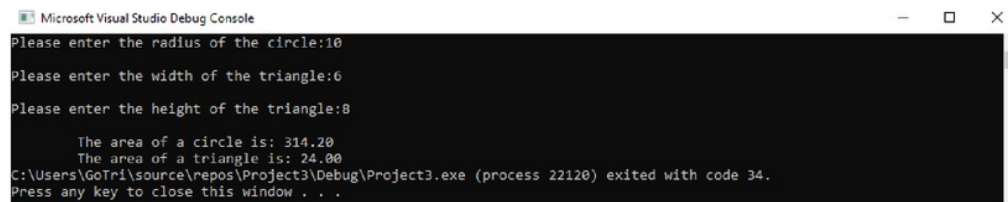


```
Microsoft Visual Studio Debug Console
Please enter the radius of the circle:10
Please enter the width of the triangle:11
Too big, Please try again
Please enter the width of the triangle:-1
Invalid value, Please try again
Please enter the width of the triangle:0
Invalid value, Please try again
Please enter the width of the triangle:5
Please enter the height of the triangle:11
Too big, Please try again
Please enter the height of the triangle:-1
Invalid value, Please try again
Please enter the height of the triangle:0
Invalid value, Please try again
Please enter the height of the triangle:10

    The area of a circle is: 314.20
    The area of a triangle is: 25.00
C:\Users\GoTri\source\repos\Project3\Debug\Project3.exe (process 20300) exited with code 34.
Press any key to close this window . . .
```

Screenshot 2:

The screenshot below shows the process of user entering correct values for each radius, width and height, upon which the area of a circle and area of a triangle is calculated. And the results are shown as below.



```
Microsoft Visual Studio Debug Console
Please enter the radius of the circle:10
Please enter the width of the triangle:6
Please enter the height of the triangle:8

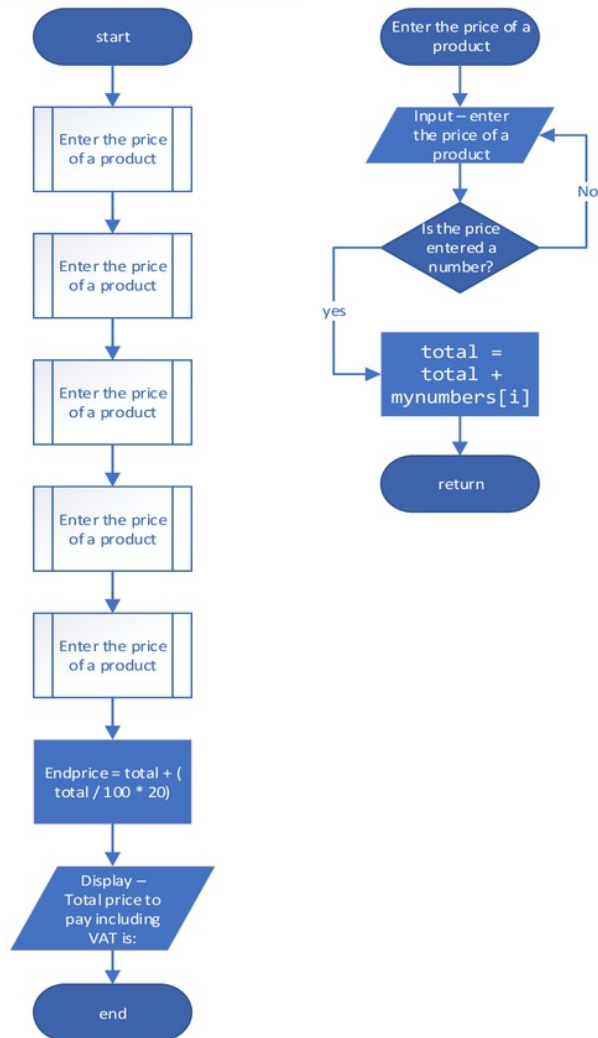
    The area of a circle is: 314.20
    The area of a triangle is: 24.00
C:\Users\GoTri\source\repos\Project3\Debug\Project3.exe (process 22120) exited with code 34.
Press any key to close this window . . .
```

Exercise 3:

Identifying problems in the exercise:

It is necessary to produce a programme, which allows the user to enter the price of 5 products into an array. A validity check is essential to ensure that the user only inputs numbers. As soon as the user has entered numbers for the 5 products the programme will be capable of calculating the provisional total value of these products. Subsequently, the programme then runs a mathematical calculation of the total provisional value divided by 100 and then multiplied by 20. The result is then added to the provisional value, which becomes the final end price. The array in this exercise is designed so that the loop runs until all relevant conditions are met, for example, until 5 valid product prices are entered by the user. Every time a product price is entered the programme checks whether the value entered is a number or a float number. If invalid information is entered by the user, the programme will ask the user to input the value again, until valid information has been entered. This process helps the programme to accurately calculate the total price to pay including VAT.

Flowchart for exercise 3:



This is the C code for exercise 3:

```

#include <stdio.h>

int main()
{
    float total = 0, division, vat, endprice; // using float because of decimal
    places, integer would be not suitable for this

    int i;
    float mynumbers[5] = { }; // declare array for the 5 numbers to be entered by
    the user

```

```

float input;

for (i = 0; i < 5; i++) // loop for the user to enter price of 5 products
{
    do
    {
        printf("\nPlease enter the price of a product: ");
    }
    while (!scanf("%f", &input) && getchar()); // checks for validity, so
the user has input a valid value, and if he did not it prompts him to enter the price
of a product again
    mynumbers[i] = input;
    total = total + mynumbers[i]; // total calculated in the for loop by
adding each number in the array to total
}

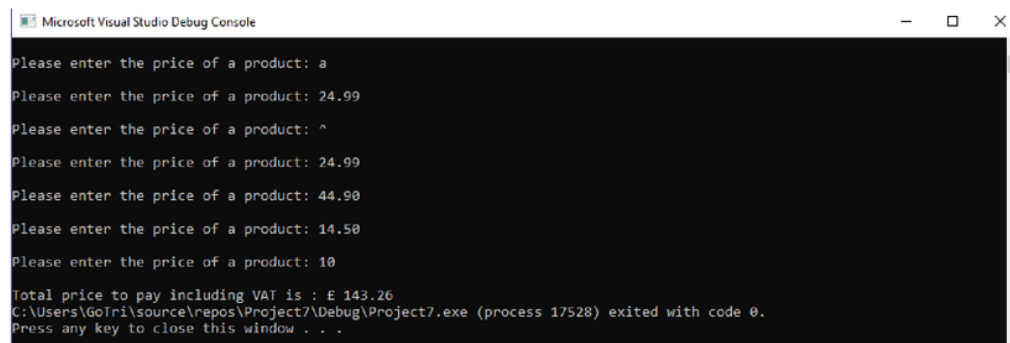
endprice = total + (total / 100 * 20); // calculation to add 20% of VAT to the
total price

printf("\nTotal price to pay including VAT is : \x9c %.2f", endprice); //
display the finale price to pay
}

```

Screenshot showing that the programme works as intended

In this screenshot it is possible to see the programme prompting the user to input 5 product prices, the user attempts to enter an invalid symbol in which case the programme does not accept that value and prompts the user to input a value again. After collecting 5 product prices the programme calculates the total price including VAT and displays the value on the screen.



```

Microsoft Visual Studio Debug Console
Please enter the price of a product: a
Please enter the price of a product: 24.99
Please enter the price of a product: ^
Please enter the price of a product: 24.99
Please enter the price of a product: 44.90
Please enter the price of a product: 14.50
Please enter the price of a product: 10
Total price to pay including VAT is : £ 143.26
C:\Users\GoTri\source\repos\Project7\Debug\Project7.exe (process 17528) exited with code 0.
Press any key to close this window . . .

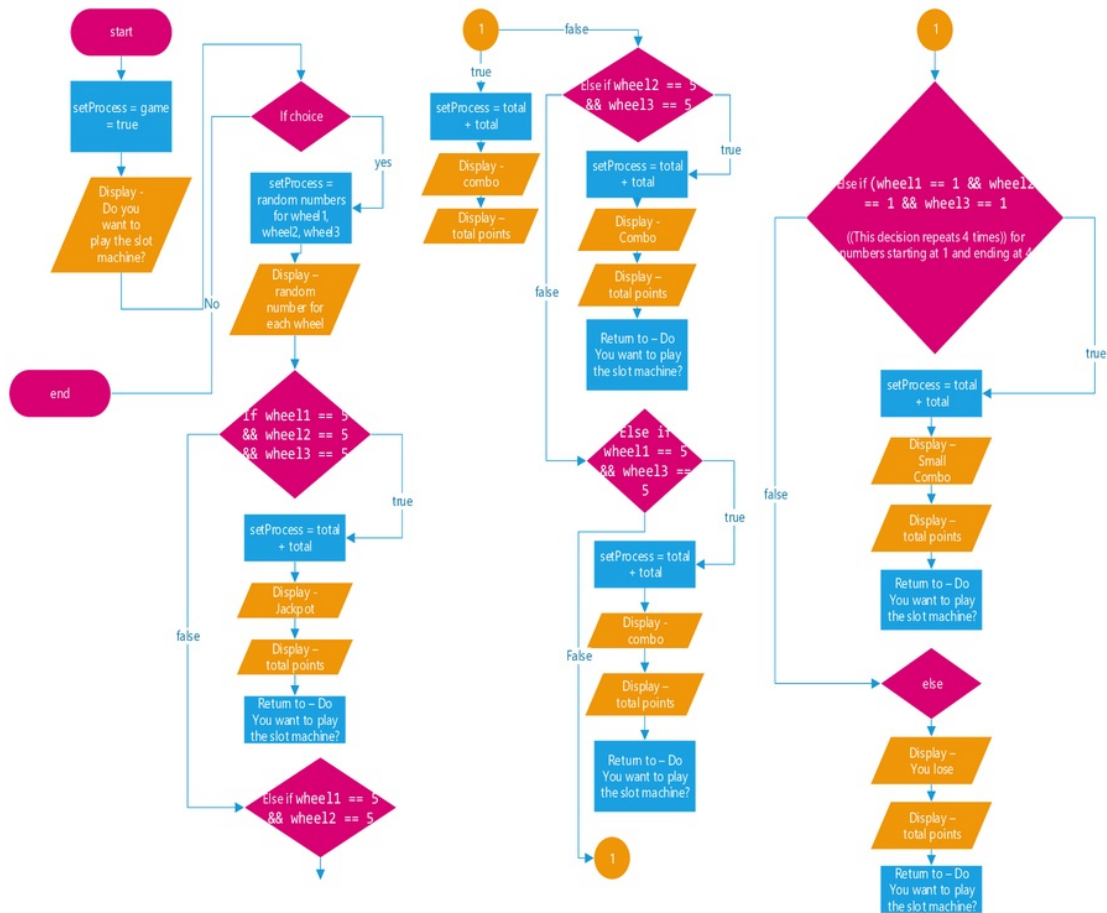
```

Exercise 4:

Identifying problems in the exercise:

In this exercise it is necessary to construct a slot machine programme capable of spinning or generating random numbers on each of the 3 wheels. To avoid the programme generating random numbers outside of the desired range, it is necessary to set a low limit of 1 and a high limit of 5. The slot machine game could be facilitated by using a long while loop, which would avoid unnecessary complications during the programming. Bool statements should be used to check the validity of the spins and to determine whether the user has won any points or has hit any combo. Again, to avoid problems, a time library should be used to generate authentic random numbers along with `strand(time(NULL))`. Multiple if, else if statements should be used to check the validity of jackpots and combos. An additional challenge is to ensure that the programme accurately records the total number of points scored by the user. To achieve this end, it should be noted that the user's total points are calculated after each spin. Every time a user has hit a winning combination, the points are added to his total number of points. Action was taken to set the total as an integer variable with an initial value of 0. Subsequently, each time the user spins the slot machine, the programme checks for all winning and losing combinations. If a winning combination is found the correct number of points is added to the total, which becomes a new total.

Flowchart for exercise 4:



The C code for exercise 4:

```
#include <stdio.h> // using the standard input and output library
#include <stdlib.h> // standard library
#include <time.h> // the use of time.h because of playing with random numbers and
generating them

int main()
{
    int low = 1; // setting the low limit of 1
    int high = 5; // setting the high limit of 5
    int wheel1 = 0, wheel2 = 0, wheel3 = 0; // giving an initial value to the
wheels to avoid possible errors
    int total = 0; // giving initial value of starting total amount of points
    char choice; // including choice for later use to want or not want to spin the
wheel

    srand(time(NULL)); // this is required to generate random number based on time

    bool jackpotKing = false, comboWombo = false, smallCombo = false; // bool
statements to check the validity of jackpots and combos
    bool game = true; // bool statement for the slot machine game

    while (game == true)
    {
        printf("\nDo you want to play the slot machine? <y/n>: "); // prompts
the user to ask whether he wants to play the slot machine
        scanf("%c", &choice); // scans the choice

        if (choice == 'y') // if choice yes

            wheel1 = rand() % (high - low + 1) + low;
            wheel2 = rand() % (high - low + 1) + low;
            wheel3 = rand() % (high - low + 1) + low;

            if (choice == 'n') // if choice no
            {
                printf("\n-----Ok maybe another time-----");
                return 0;
            }

            printf("\n-----%d %d %d-----\n", wheel1, wheel2, wheel3); // display
the random values that were generated on each wheel

            if (wheel1 == 5 && wheel2 == 5 && wheel3 == 5) // checks for whether the
user has won a jackpot
            {
                total = total + 2000; // adds the correct amount of points in the
case of jackpot

                jackpotKing = true; // if jackpot true
                printf("\n-----Jackpot!-----"); // displays jackpot
                printf("\nYour total points are: %d", total); // shows the total
amount of points the user has
            }
            else if (wheel1 == 5 && wheel2 == 5) // checks for smaller jackpot
            {
                total = total + 100; // adds the correct amount of points in the
case of smaller jackpot
                comboWombo = true; // if smaller jackpot true
            }
        }
    }
}
```

```

        printf("\n-----Combo!-----"); // displays the appropriate
message      printf("\nYour total points are: %d", total);
    }
    else if (wheel2 == 5 && wheel3 == 5)
    {
        total = total + 100;
        comboWombo = true;
        printf("\n-----Combo!-----");
        printf("\nYour total points are: %d", total);
    }
    else if (wheel1 == 5 && wheel3 == 5)
    {
        total = total + 100;
        comboWombo = true;
        printf("\n-----Combo!-----");
        printf("\nYour total points are: %d", total);
    }
    else if (wheel1 == 1 && wheel2 == 1 && wheel3 == 1) // check for mini
price      {
case of mini price    total = total + 20; // adds the correct amount of points in the
                        smallCombo = true; // if mini price true
message      printf("\n-----Small Combo!-----"); // displays appropriate
                        printf("\nYour total points are: %d", total);
    }
    else if (wheel1 == 2 && wheel2 == 2 && wheel3 == 2)
    {
        total = total + 20;
        comboWombo = true;
        printf("\n-----Small Combo!-----");
        printf("\nYour total points are: %d", total);
    }
    else if (wheel1 == 3 && wheel2 == 3 && wheel3 == 3)
    {
        total = total + 20;
        smallCombo = true;
        printf("\n-----Small Combo!-----");
        printf("\nYour total points are: %d", total);
    }
    else if (wheel1 == 4 && wheel2 == 4 && wheel3 == 4)
    {
        total = total + 20;
        smallCombo = true;
        printf("\n-----Small Combo!-----");
        printf("\nYour total points are: %d", total);
    }
    else // else statement in the case of user not winning anything
    {
message      printf("\n-----You lose-----\n"); // displays appropriate
                        printf("\nYour total points are: %d", total);
    }
}

}

```

Screenshots showing that the programme works as intended

Screenshot 1:

The screenshot below shows correct addition of points into user's total points, upon successfully hitting a winning combination

```
C:\Users\GoTri\source\repos\Project11\Debug\Project11.exe
Your total points are: 100
Do you want to play the slot machine? <y/n>: y
-----1 5 3-----
-----You lose-----
Your total points are: 100
Do you want to play the slot machine? <y/n>: y
-----5 3 4-----
-----You lose-----
Your total points are: 100
Do you want to play the slot machine? <y/n>: y
-----2 5 5-----
-----Combo!-----
Your total points are: 200
Do you want to play the slot machine? <y/n>: y
-----3 3 4-----
-----You lose-----
Your total points are: 200
Do you want to play the slot machine? <y/n>: y
-----4 2 1-----
-----Small Combo!-----
Your total points are: 120
Do you want to play the slot machine? <y/n>: y
-----2 5 5-----
-----Combo!-----
Your total points are: 220
Do you want to play the slot machine? <y/n>: y
-----4 5 2-----
-----You lose-----
Your total points are: 220
Do you want to play the slot machine? <y/n>: n
-----Ok maybe another time-----
C:\Users\GoTri\source\repos\Project11\Debug\Project11.exe (process 10712) exited with code 0.
Press any key to close this window . . .
```

Screenshot 2:

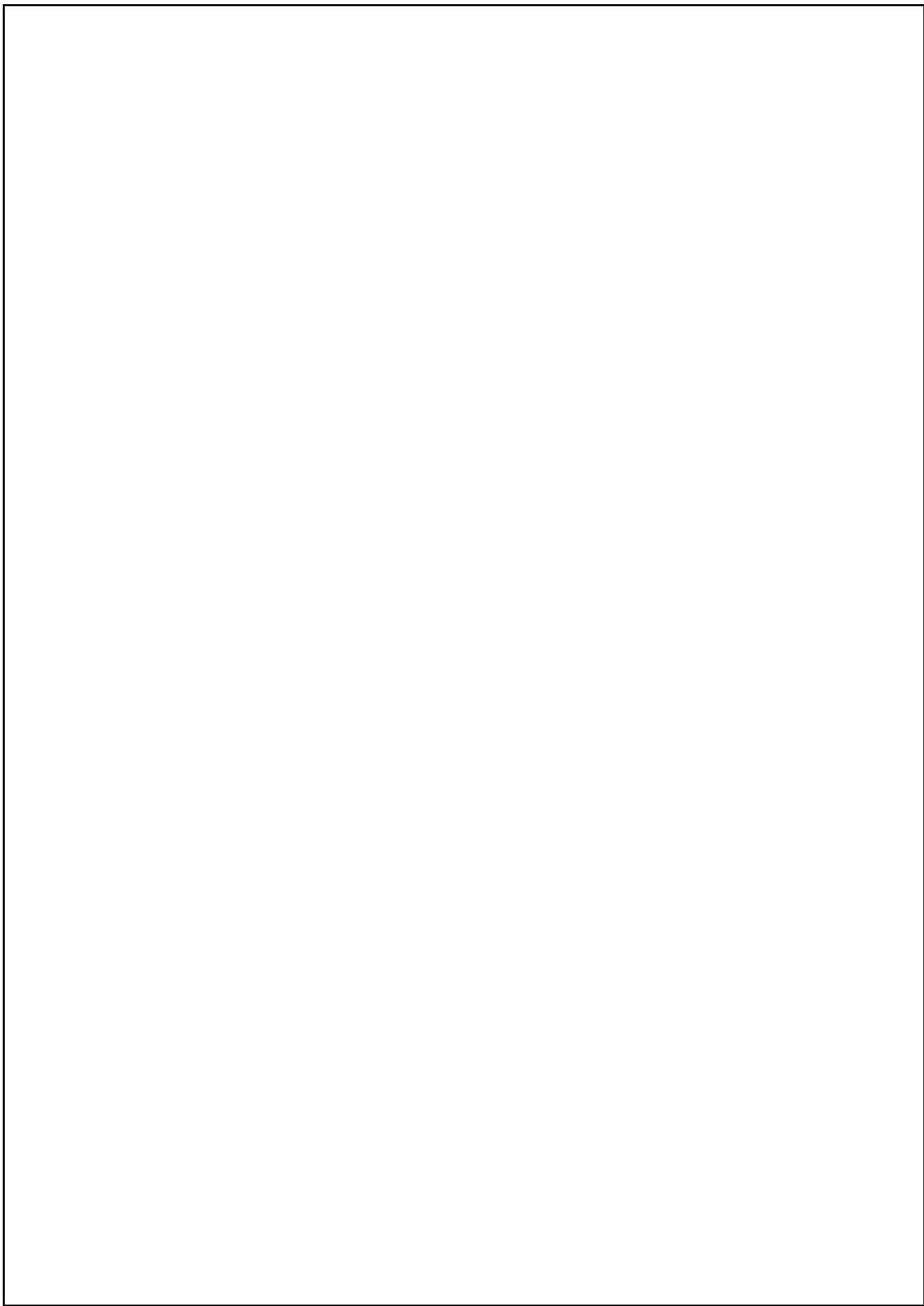
This screenshot below shows the correct addition of points into user's total points, upon successfully hitting another winning combination.

```
Microsoft Visual Studio Debug Console
-----4 2 1-----
-----You lose-----
Your total points are: 100
Do you want to play the slot machine? <y/n>: y
-----1 1 1-----
-----Small Combo!-----
Your total points are: 120
Do you want to play the slot machine? <y/n>: y
-----2 5 5-----
-----Combo!-----
Your total points are: 220
Do you want to play the slot machine? <y/n>: y
-----4 5 2-----
-----You lose-----
Your total points are: 220
Do you want to play the slot machine? <y/n>: n
-----Ok maybe another time-----
C:\Users\GoTri\source\repos\Project11\Debug\Project11.exe (process 10712) exited with code 0.
Press any key to close this window . . .
```

Screenshot 3:

The final screenshot shows the user hitting a Jackpot winning combination, upon which the correct amount of points is added to the user's total amount of points. Additionally, all these 3 screenshots shows that the user can spin as many times as he wishes to do so, which was required to achieve in this exercise.

```
C:\Users\GoTri\source\repos\Project11\Debug\Project11.exe
-----Small Combo!-----
Your total points are: 980
Do you want to play the slot machine? <y/n>: y
-----4 2 3-----
-----You lose-----
Your total points are: 980
Do you want to play the slot machine? <y/n>: y
-----2 3 4-----
-----You lose-----
Your total points are: 980
Do you want to play the slot machine? <y/n>: y
-----5 5 5-----
-----Jackpot!-----
Your total points are: 2980
Do you want to play the slot machine? <y/n>: y
-----1 1 3-----
-----You lose-----
Your total points are: 2980
Do you want to play the slot machine? <y/n>:
```



Programs in C and documentation (M4X01508)

GRADEMARK REPORT

FINAL GRADE

84/100

GENERAL COMMENTS

Instructor

Part A

Variables - Good overall - you need to state what sorts of information are stored in the various types eg char is one character. good discussion of the naming conventions and reserved words 9/10

Counting Loops - The point with a for loop against a while loop is that the number of iterations is known before the loop starts - you sort of skirt around this, but you've explained it fairly well. You have included a good example too 8/10

Arrays - Good explanation and very good example. You could mention that in C arrays are counted from 0, although the example demonstrated that you know that 9.5/10

Part B

Exercise 1

Code-Annotation Good overall, not entirely efficient (you could have combined your if and else if into a double condition, as the contents of the statement is identical) Very good validation, 4.5/5

Documentation - Flowchart is good - on the testing, don't forget to test exactly on the boundaries of your if statements ie 120 and 15 for age 4/5

Exercise 2

Code-Annotation - Good validation and good use of functions, well done. 10/10

Documentation - Good flowchart and testing has shown all possibilities 5/5

Exercise 3

Code-Annotation Good overall, you need to check if any number entered are negative because this would not be possible for a price 8/10

Documentation You could have shown the "Enter the price of a product" as a loop rather than repeating it 5 times. Also, you have not mentioned validating for negative numbers. Testing is a bit short, you need to test for the full range of inputs including negative numbers 3/5

Exercise 4

Code-Annotation A few minor things - you've left out the brackets on your `if(choice=='y')` if statement which could make the code unstable. Some of your if statements could have been combined. You don't really need to test for all 1s, all 2s etc, once you know that they're not 5s, it's enough to check if they're all equal - in this way you could have used fewer if statements. A good effort overall though and accumulating points implemented. 16 /20

Documentation - A bit too much code in your flowchart - make sure you're showing just the logic rather than implementation details. Overall you have shown the logic well though. For testing, you need to show all combinations as you have an if statement for each eg all 2s, all 3s, 5s in all positions etc. 7/10

Total: 84%

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
