

Richard Hankinson p137603 software development part A & B

by Richard Peter Hankinson

FILE	12382_RICHARD_PETER_HANKINSON_RICHARD_HANKINSON_P137603_SOFTWARE_DEVELOPMENT_PART_A__B_157877_1289697578.DOCX (1.62M)		
TIME SUBMITTED	24-FEB-2019 11:24AM (UTC+0000)	WORD COUNT	2715
SUBMISSION ID	101091423	CHARACTER COUNT	12802

PART A (UNDERSTANDING) (due date: February 25th 2019, 3pm)

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

A variable is a storage area in a program created by a programmer in which data can be assigned and stored inside. The variable is assigned a name, it can be called anything (not starting with a number or containing any special characters) but it is good practice to call it something related to its intended use [1].

The variable can be called upon anywhere in the program depending if it is global (defined before the program, after the libraries) or local (defined inside a function).

The data inside the variable can be changed by either the user or the program. This is the opposite of a constant which is also defined locally or globally but has a fixed value.

The data type expected to be assigned and stored to a variable must be declared for the variable. The data types are:

Int = Whole number

Float = Number with 6 decimal places

Char = Single character (e.g the letter 'p' or 'a')

The use of variables is important for writing programs because it allows the data to be manipulated by the user or the program whilst making sense to the programmer. If data was being manipulated without the use of a variable, and the programmer had data changing, without being fully aware of what the data doing, it would get very confusing.

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.

The for-loop can also be referred to as a counting-loop because the programmer places a condition on the for-loop, stating how many times the loop will run for. This is done by first creating a counter variable as an integer (int "i" is commonly used for this). Then create the condition (i=0; i<=4; i++) for example.

The steps taken by the program are as follows:

1. if i is equal to 0, and i is less than equal to 4, and add 1 to i,
2. if this is true
3. run the loop.
4. If this is false
5. End the loop

By adjusting the value of i<=4, the number of cycles of the loop can be stated.

A time where a for-loop comes in handy is when using arrays. By using a for-loop to change to the next element within an array when obtaining data to be stored within it. This allows the programmer to only have to type the code to obtain the data, and assign it to the array, once. Then use a counter to incrementally change to the next element in the array. This saves time and makes the program look much neater.

For an example of an array in use, see section B, exercise 3.

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

An array allows multiple values of the same data type, to be stored in a single variable. An array is created by defining a variable and then including square brackets after the variable name []. Inside the square brackets, the programmer specifies how many memory spaces will be allocated to this array [1]. For example this could look like (int myarray [5];).

(Note that because computers like to count from 0, 5 memory spaces in an array will be assigned from 0-4 and not 1-5.)

As with variables, arrays can be int, float, char or string. Arrays can also become multi-dimensional by having two or more index [][] locations assigned to them. This becomes useful when displaying tables of information or coordinates.

For an example of an array in use see section B, exercise 3.

PART B (IMPLEMENTATION) (due date: February 25th 2019, 3pm)

The following exercises become increasingly more challenging. You don't need to complete all the exercises to pass this assignment. Marks will be awarded for attempting the exercises – a partial solution will still gain marks.

For each exercise below:

- Identify specific problems in completing the exercise.
 - You may also constrain a problem (simplify it) to produce a simpler solution that may act as a stepping stone to solving the real problem
- Write a clear analysis of the solutions (the thought process behind them) – what logical stages are required. You could use a flowchart or pseudocode.
- Show a C code listing with annotations
- Show testing (screenshots) that clearly demonstrate that the solutions work under all circumstances.

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.d

The specific problems with setting up this programme are collecting information from the user and then displaying them on the screen once all inputs have been entered.

1. Setup programme library
2. Declare variables: name = char[10], height = float, age = int
3. Prompt user to input name
4. Collect string
5. Prompt user to input height
6. Collect float to 2 decimal places
7. Prompt user to input age
8. Collect int
9. Display collected information onscreen with description to user
10. End program

```

524 // Part B exercise 1: three variables
525
526 #include <stdio.h>
527
528 int main()
529 {
530     // variables defined below: name, height and age
531     char name[10]; //number of characters for string defined by number
532     float height; //float because of decimal point
533     int age; // int because we are expecting a whole number
534
535     printf("Please enter your name "); // prompt for user to enter their name
536     scanf("%s", name); // collect entered letters as a string and store in name
537
538     printf("Please enter your hight in meters "); // prompt for user to enter their height
539     scanf("%f", &height); // collect entered numbers as a float and store in height
540
541     printf("Please eneter your age "); // prompt for user to enter their age
542     scanf("%d", &age); // collect entered numbers as a int and store in age
543
544     // display input information to the user in written english:
545
546     // ("%s", name) will display info stored in name variable
547     printf("\nYour name is %s \n", name);
548
549     // ("%f", height) will display info stored in hight float variable. %.2f means display to only 2
550     // decimal places.
551     printf("Your height is %.2f meters\n", height);
552
553     // ("%d", age) will display info stored in age int variable
554     printf("Your age is %d \n", age);
555
556     return 0;
557 }

```

The first test of the programme with expected values entered by the user.

```

Please enter your name Richard
Please enter your hight in meters 1.77
Please eneter your age 30

Your name is Richard
Your height is 1.77 meters
Your age is 30
Program ended with exit code: 0

```

Second test shows height entered to more than 2 decimal places. Programme output to only two decimal places to user.

```

Please enter your name Simon
Please enter your hight in meters 2.345
Please eneter your age 34.3

Your name is Simon
Your height is 2.35 meters
Your age is 34
Program ended with exit code: 0

```

To limit the user inputting floats and integers of ridiculous numbers a while loop can be added to set parameters to re-prompt the user to input a correct value.

```

541 while (height>3.5) // if user inputs vaule higher than 3.5 run loop to ask for another answer untill
    its a useable value.
542
543 {
544     printf("Please enter your correct hight in meters "); // re-prompt for user to enter their
        height
545     scanf("%f", &height); // collect entered numbers as a float and store in height
546
547 }
548
549 printf("Please eneter your age "); // prompt for user to enter their age
550 scanf("%d",&age); // collect entered numbers as a int and store in age
551
552 while (age>110) //repromt user to enter a value below 110
553 {
554     printf("Please eneter your correct age "); // prompt for user to enter their correct age
        scanf("%d",&age); // collect entered numbers as a int and store in age
555
556 }
557
558 while (age<1) //repromt user to enter a value of 1 or above
559 {
560     printf("Please eneter your correct age "); // prompt for user to enter their correct age
        scanf("%d",&age); // collect entered numbers as a int and store in age
561
562 }
563

```

For the age part, if a value of over 3.5 is entered the while loop will run and "please enter your correct height in meters" will be printed on the screen. This will repeat until a value of under 3.5 is entered. The same with the age section but this time there are two while loops. The first loop will run if a value over 110 has been entered resulting in the message "Please enter your correct age" being displayed to the user. The second loop will run displaying the same message if a value of less than 1 is entered.

Results when entering a number above 3.5 in height section.

```

Please enter your name Richard
Please enter your hight in meters 3.7
Please enter your correct hight in meters 5.4
Please enter your correct hight in meters 30
Please enter your correct hight in meters 3.4
Please eneter your age 30

Your name is Richard
Your height is 3.40 meters
Your age is 30
Program ended with exit code: 0

```

Results entering above the age range. User prompted to enter correct age. Program continue only when number enter between 1 and 110.

```

Please enter your name Richard
Please enter your hight in meters 1.77
Please eneter your age 111
Please eneter your correct age 0
Please eneter your correct age 30

Your name is Richard
Your height is 1.77 meters
Your age is 30
Program ended with exit code: 0

```

Results entering value with a decimal point below the age range.

```

Please enter your name Richard
Please enter your hight in meters 1.77
Please eneter your age 0.9
Please eneter your correct age Please eneter your correct age
Please eneter your correct age Please eneter your correct
age Please eneter your correct age Please eneter your
correct age Please eneter your correct age Please eneter
your correct age Please eneter your correct age Please
eneter your correct age Please eneter your correct age
Please eneter your correct age Please eneter your correct
age Please eneter your correct age Please eneter your
correct age Please eneter your correct age Please eneter
your correct age Please eneter your correct age Please

```

With the user entering a float value of 0.9 it has sent the while loop into an infinite loop. Scanf, can not use a number of an incorrect data type, which sends it into this loop. To combat this the following command can be added "scanf("%*^\n\n");" [2]. This stops the loop reading nothing and going into an infinity loop. It tells the loop to discard the input.

```
Please enter your name Richard
Please enter your hight in meters 1.77
Please eneter your age 0.9
Please eneter your correct age 30
Please eneter your correct age
Your name is Richard
Your height is 1.77 meters
Your age is 30
Program ended with exit code: 0|
```

Below is a screenshot of the programme with annotations:

```
554 // Part 8 excercise 1: three variables
555
556 #include <stdio.h>
557
558 int main()
559 {
560     // variables defined below: name, height and age
561     char name[10]; //number of characters for string defined by number
562     float height; //float because of decimal point
563     int age; // int because we are expecting a whole number
564
565     printf("Please enter your name "); // prompt for user to enter their name
566     scanf("%s", name); // collect entered letters as a string and store in name
567
568     printf("Please enter your hight in meters "); // prompt for user to enter their height
569     scanf("%f", &height); // collect entered numbers as a float and store in height
570
571     while (height>3.5) // if user inputs vaule higher than 3.5 run loop to ask for another answer untill its a useeble value.
572     {
573         printf("Please enter your correct hight in meters "); // re-prompt for user to enter their height
574         scanf("%f", &height); // collect entered numbers as a float and store in height
575     }
576
577     printf("Please eneter your age "); // prompt for user to enter their age
578     scanf("%d",&age); // collect entered numbers as a int and store in age
579
580     while (age>110) //reprompt user to enter a value below 110
581     {
582         printf("Please eneter your correct age "); // prompt for user to enter their correct age
583         scanf("%d",&age); // collect entered numbers as a int and store in age
584         scanf("%*^\n\n"); // stop infinity loop
585     }
586
587     while (age<1) //reprompt user to enter a value of 1 or above
588     {
589         printf("Please eneter your correct age "); // prompt for user to enter their correct age
590         scanf("%d",&age); // collect entered numbers as a int and store in age
591         scanf("%*^\n\n"); // stop infinity loop
592     }
593
594     // display input information to the user in writen english:
595
596     // ("%s", name) will display info stored in name variable
597     printf("\nYour name is %s \n",name);
598
599     // ("%f", height) will display info stored in hight float variable. %.2f means display to only 2 decimal places.
600     printf("Your height is %.2f meters\n",height);
601
602     // ("%d", age) will display info stored in age int variable
603     printf("Your age is %d \n", age);
604
605     return 0;
606 }
```

Code Below:

```
// Part B excersise 1: three variables
```

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

```
int main()
```

```
{
    char name[10];
    float height;
    int age;

    printf("Please enter your name ");
    scanf("%s", name);

    printf("Please enter your hight in meters ");
    scanf("%f", &height);

    while (height>3.5)

    {
        printf("Please enter your correct hight in meters ");
        scanf("%f", &height);
    }

    printf("Please eneter your age ");
    scanf("%d",&age);

    while (age>110)
    {
        printf("Please eneter your correct age ");
        scanf("%d",&age);
        scanf("%*^\n\n");
    }

    while (age<1)
    {
        printf("Please eneter your correct age ");
        scanf("%d",&age);
        scanf("%*^\n\n");
    }

    printf("\nYour name is %s \n",name);
    printf("Your height is %.2f meters\n",height);
    printf("Your age is %d \n", age);

    return 0;
}
```

Exercise 2:

Question:

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.

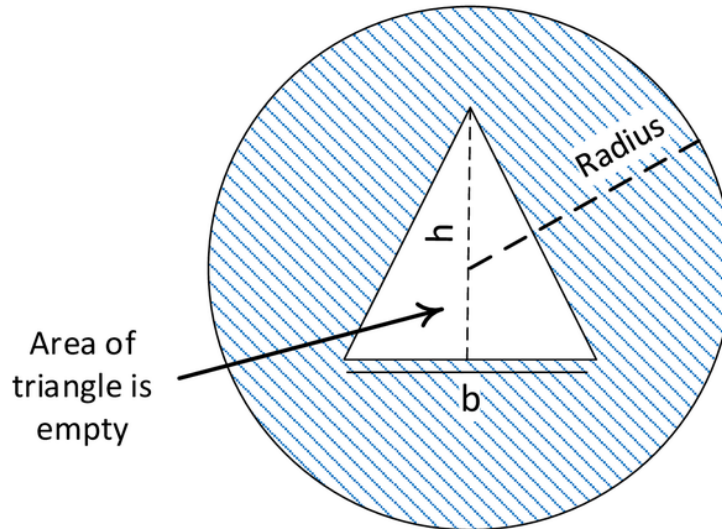


Figure 1 Calculate Shaded Area

Note:

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

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

Answer:

The specific problems with setting up this programme are collecting input data from user. Then using math functions to simulate the given equations and then output data to user.

1. Set up programme library
2. Define constant (PI = 3.141)
3. Define variables: radius, bvalue, hvalue, circlearea, trianglearea, totalarea
4. Prompt user to input Radius value
5. Prompt user to input value h
6. Prompt user to input value b
7. Calculate area of circle
8. Calculate area of triangle
9. Subtract area of triangle from area of circle
10. Display results to user

There is no squared math function in C language, so to get the correct answer, the radius squared will have to become radius times by radius.

Below is a screenshot of the programme with annotations:

```
586 //Part B excersise 2: area of circle with triangle inside
587
588 #include <stdio.h>
589 #define PI 3.142
590
591 int main()
592 {
593
594     float radius, bvalue, hvalue, circlearea, trianglearea, totalarea; //defined variables
595
596     printf("Please enter the radius of the circle "); //promt radius value from user
597     scanf("%f", &radius); //collect radius value and store in radius
598
599     printf("Please enter the value of base value b "); //promt base value from user
600     scanf("%f", &bvalue); //collect base value and store in bvalue
601
602     printf("Please enter the vaule of height value h "); //promt rheight value from user
603     scanf("%f", &hvalue); //collect height value and store in hvalue
604
605     circlearea=(radius*radius)*PI; //calculate area of the circle
606
607     trianglearea=bvalue*hvalue/2; //calculate the area of the triangle
608
609     totalarea=circlearea-trianglearea; //subtract the triangle area from the circle area
610
611     printf("The area of the shades shape is %.3f", totalarea); //display the reseult to the user
612
613     return 0;
614 }
```

Test screen shot inserting floating values.

```
Please enter the radius of the circle 10.23
Please enter the value of base value b 3.76
Please enter the vaule of height value h 4.98
The area of the shades shape is 319.457Program ended with exit
code: 0
```

Code below:

```
//Part B excersise 2: area of circle with triangle inside

#include <stdio.h>
#define PI 3.142

int main()
{
    float radius, bvalue, hvalue, circlearea, trianglearea, totalarea;

    printf("Please enter the radius of the circle ");
    scanf("%f", &radius);

    printf("Please enter the value of base value b ");
    scanf("%f", &bvalue);

    printf("Please enter the vaule of height value h ");
    scanf("%f", &hvalue);

    circlearea=(radius*radius)*PI;
    trianglearea=bvalue*hvalue/2;
    totalarea=circlearea-trianglearea;

    printf("The area of the shades shape is %.3f", totalarea);

    return 0;
}
```

Exercise 3:

Question:

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.

Answer:

The specific problems with setting up this programme are creating an array to store the prices entered by the user. A counter (for-loop) to automatically move to the next storage location in the array (otherwise the entered information will keep on overwriting the stored price values). Setting up the loop to only run the same amount of times as the number of storage locations in the array. The maths on how to calculate the VAT. Divide the total by 100 (gives 1% value) and times by the percentage (20) and add the original total onto this.

1. Setup programme library
2. Setup an array to hold 5 sets of floating numbers
3. Declare variable "int i" to use as counter
4. Set up for loop to collect price information from the user and store in array
5. Declare variables for array data to be assigned to
6. Assign data from array to variables
7. Set up loop to add variables together
8. Display total to the user
9. Calculate VAT at 20% and add it to the total
10. Display total with added VAT to the user

Below is a screenshot of the programme with annotations:

```
621 //Part B exercise 3: 5 product array with total and VAT calculated
622
623 #include <stdio.h>
624
625 int main()
626 {
627     float productname[5]; //declare array name and number
628     float productname1, productname2, productname3, productname4, productname5, totalexvat = 0.0, totalincvat; //defined
        variables
629     int i; // counter
630
631     productname1 = productname[0]; // stored array values assigned to variables
632     productname2 = productname[1];
633     productname3 = productname[2];
634     productname4 = productname[3];
635     productname5 = productname[4];
636
637     for (i=0; i<=4; i++) // loop to store data in array. starting point, ending point and plus 1 counter
638     {
639         printf("Product %d price: ", i+1); // prompt user and counter for number displayed to user
640         scanf("%f", &productname[i]); // collect data and assign to array value of i (increases by one each loop)
641     }
642
643     for (i=0; i<=4; i++) //loop for adding stored array values. Starting and counting same as before
644     {
645         totalexvat += productname[i]; // adding values stored in array together and storing in totalexvat
646     }
647
648     printf("Your total ex VAT is: %.2f\n", totalexvat); // displaying total price excluding VAT to user
649
650     totalincvat = totalexvat / 100 * 20 + totalexvat; // calculating VAT at 20% and adding it to excluding VAT total
651
652     printf("Your total inc VAT is: %.2f\n", totalincvat); // displaying total price including VAT
653
654     return 0;
655 }
```

Below is a very simple test that is very easy to do mentally to see if the programme is working.

```
Product 1 price: 5.00
Product 2 price: 5.00
Product 3 price: 5.00
Product 4 price: 5.00
Product 5 price: 5.00
Your total ex VAT is: 25.00
Your total inc VAT is: 30.00
Program ended with exit code: 0
```

Below is another simulation with the correct results.

```
Product 1 price: 2.99
Product 2 price: 4.53
Product 3 price: 7.99
Product 4 price: 30.99
Product 5 price: 55
Your total ex VAT is: 101.50
Your total inc VAT is: 121.80
Program ended with exit code: 0|
```

Below, a real-life test to work out the VAT on a laser CNC machine. Inserting only one value to see how the programme will run. Results are correct.

```
Product 1 price: 2691.00
Product 2 price: 0
Product 3 price: 0
Product 4 price: 0
Product 5 price: 0
Your total ex VAT is: 2691.00
Your total inc VAT is: 3229.20
Program ended with exit code: 0|
```

Below a test with no values added. Results are expected.

```
Product 1 price: 0
Product 2 price: 0
Product 3 price: 0
Product 4 price: 0
Product 5 price: 0
Your total ex VAT is: 0.00
Your total inc VAT is: 0.00
Program ended with exit code: 0
```

Below, inserting one price for product three and zero for the others.

```
Product 1 price: 0
Product 2 price: 0
Product 3 price: 2691
Product 4 price: 0
Product 5 price: 0
Your total ex VAT is: 2691.00
Your total inc VAT is: 3229.20
Program ended with exit code: 0|
```

Code below:

```
//Part B exercise 3: 5 product array with total and VAT calculated

#include <stdio.h>

int main()
{
    float productname[5];
    float productname1, productname2, productname3, productname4, productname5, totalexvat = 0.0, totalincvat;
    int i;

    productname1 = productname[0];
    productname2 = productname[1];
    productname3 = productname[2];
    productname4 = productname[3];
    productname5 = productname[4];

    for (i=0; i<=4; i++) // loop to store data in array. starting point, ending point and plus 1 counter
    {
        printf("Product %d price: ", i+1);
        scanf("%f", &productname[i]);
    }

    for (i=0; i<=4; i++) //loop for adding stored array values. Starting and counting same as before
    {
        totalexvat += productname[i];
    }

    printf("Your total ex VAT is: %.2fn", totalexvat);
    totalincvat = totalexvat / 100 * 20 + totalexvat;
    printf("Your total inc VAT is: %.2fn", totalincvat);

    return 0;
}
```

Exercise 4:

Question:

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

```
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
```

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 100
Spin? Y
5 5 5
You win, and your points are 2100
```

Answer:

The specific problems with setting up this programme are generating three random numbers that are different from each other and are completely unpredictable, ranging between 1 and 5. Creating a yes/no function to end the code when the user wants to. Unlimited amount of game plays and showing the user their score total.

1. Set up function libraries and main

2. Set up a random number generator using rand
3. Use srand(time(NULL)) to generate truly random number using time clock as its parameter
4. Set up wheel variables for counter wheels; wheel1, wheel2, wheel3
5. Generate a random number between 1 and 5 to store in each wheel variable
6. Set up winning conditions with if, else if and else
7. All the same number = 20 points
8. Two fives = 100 points
9. Three fives = 2000 points
10. None of those = 0 points
11. Create variable for score and send and accumulate results
12. Display results to user
13. Ask user if they want to play again
14. Create a yes or no scenario using a do while loop
15. Create variable for playagain
16. Keep running program unless play again is storing y from the user
17. End program

Below is a screenshot of the programme with annotations:

```

709 // Section B Example 4: Fruit Machine
710
711 #include <stdio.h>
712 #include <stdlib.h>
713 #include <time.h>
714
715 int main()
716 {
717     char playagain; // used to store char 'y' specified later to end game
718     int score = 0; // counts score until game is ended
719
720     do // run this program as long as long as playagain variable is not equal to y
721     {
722         srand((unsigned int)time(0)); //seed random number using time as its parameter which is constantly changing. unsigned int stops warning about int precision.
723
724         int wheel1 = rand() % 5 + 1; // random number divided by 5, output remainder (0-4) need to add 1 to make 1-5
725         int wheel2 = rand() % 5 + 1; // random number divided by 5, output remainder (0-4) need to add 1 to make 1-5
726         int wheel3 = rand() % 5 + 1; // random number divided by 5, output remainder (0-4) need to add 1 to make 1-5
727
728         printf("%d %d %d\n", wheel1, wheel2, wheel3); // print out turn values to user
729
730         if ((wheel1 == 1 && wheel2 == 1 && wheel3 == 1) || (wheel1 == 2 && wheel2 == 2 && wheel3 == 2) || (wheel1 == 3 && wheel2 == 3 && wheel3 == 3) || (wheel1 == 4
&& wheel2 == 4 && wheel3 == 4)) //Run if all three numbers of the same number are generated (excluding fives)
731         {
732             printf("You won 20 points!\n"); // tell user they have won twenty points
733             score += 20; //add twenty to score
734         }
735
736         else if ((wheel1 == 5 && wheel2 == 5 && wheel3 == 1) || (wheel1 == 5 && wheel2 == 5 && wheel3 == 2) || (wheel1 == 5 && wheel2 == 5 && wheel3 == 3) || (wheel1 == 5 &&
wheel2 == 5 && wheel3 == 4) || (wheel1 == 1 && wheel2 == 5 && wheel3 == 5) || (wheel1 == 2 && wheel2 == 5 && wheel3 == 5) || (wheel1 == 3 && wheel2 == 5 && wheel3
== 5) || (wheel1 == 4 && wheel2 == 5 && wheel3 == 5) || (wheel1 == 5 && wheel2 == 1 && wheel3 == 5) || (wheel1 == 5 && wheel2 == 2 && wheel3 == 5) || (wheel1 == 5 &&
wheel2 == 3 && wheel3 == 5) || (wheel1 == 5 && wheel2 == 4 && wheel3 == 5)) //Run if there are two fives generated
737         {
738             printf("You won 100 points!\n"); // tell user they have won one hundred points
739             score += 100; //add one hundred to score
740         }
741
742         else if (wheel1 == 5 && wheel2 == 5 && wheel3 == 5) //Run if three fives are generated
743         {
744             printf("You won the BIG JACKPOT 2000 points!\n"); // tell user they have won two hundred points
745             score += 2000; //add two hundred to score
746         }
747
748         else
749         {
750             printf("You lose!\n"); //tell user they lost that turn
751         }
752
753         printf("SCORE: %d\n", score); //output score to user after every turn
754         printf("Play again press enter: \n End game? Press y then enter: \n"); // Prompt user to play again or exit game
755         scanf("%c", &playagain); //collects value and stores in playagain. Anything except a y will continue the game
756     }
757
758     while (playagain != 'y'); // run this program as long as long as playagain variable is not equal to y.
759     // When playagain is equal to y the while loop ends and the programme goes to return and ends
760     return 0; // ends programme
761 }

```

By adding an or || in between each condition it is possible to have multiple condition in statement [1]. This is important because before knowing this many else if statements were used to complete the same task taking up lots of space and looking bulky.

Below is the code for the fruit machine:

```

// Section B Example 4: Fruit Machine

#include <stdio.h>

```

```

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

int main()
{
    char playagain;
    int score = 0;

    do
    {
        srand ((unsigned int)time(0));

        int wheel1 = rand() % 5 + 1
        int wheel2 = rand() % 5 + 1
        int wheel3 = rand() % 5 + 1

        printf("%d %d %d\n", wheel1, wheel2, wheel3);

        if ((wheel1 == 1 && wheel2 == 1 && wheel3 == 1)|| (wheel1 == 2 && wheel2 == 2 && wheel3 ==
2) || (wheel1 == 3 && wheel2 == 3 && wheel3 == 3) || (wheel1 == 4 && wheel2 == 4 && wheel3 == 4))
        {
            printf("You won 20 points!\n");
            score += 20;
        }

        else if ((wheel1 == 5 && wheel2 == 5 && wheel3 == 1)||(wheel1 == 5 && wheel2 == 5 && wheel3
== 2)||((wheel1 == 5 && wheel2 == 5 && wheel3 == 3)||(wheel1 == 5 && wheel2 == 5 && wheel3 ==
4)||((wheel1 == 1 && wheel2 == 5 && wheel3 == 5)||((wheel1 == 2 && wheel2 == 5 && wheel3 ==
5)||((wheel1 == 3 && wheel2 == 5 && wheel3 == 5)||((wheel1 == 4 && wheel2 == 5 && wheel3 ==
5)||((wheel1 == 5 && wheel2 == 1 && wheel3 == 5)||((wheel1 == 5 && wheel2 == 2 && wheel3 ==
5)||((wheel1 == 5 && wheel2 == 3 && wheel3 == 5)||((wheel1 == 5 && wheel2 == 4 && wheel3 == 5))

        {
            printf("You won 100 points!\n");
            score += 100;
        }

        else if (wheel1 == 5 && wheel2 == 5 && wheel3 == 5)
        {
            printf("You won the BIG JACKPOT 2000 points!\n");
            score += 2000;
        }

        else
        {
            printf("You lose\n");
        }

        printf("SCORE: %d\n", score);
        printf("Play again press enter: \n End game? Press y then enter: \n");
        scanf("%c", &playagain);
    }

    while (playagain != 'y');
    return 0;
}

```

Works Cited

- [1] M. McGrath, C Programming in easy steps, Southam , 2002-2004.
- [2] "Infinite loop when assigning a float to an integer," [Online]. Available:
<https://stackoverflow.com/questions/20593961/infinite-loop-when-assigning-a-float-to-an-integer>.
[Accessed 21 Feb 2019].

FINAL GRADE

/100

GENERAL COMMENTS

Instructor

Part A

Variables - A good explanation, you could also have talked about the different amounts of memory taken up by different types of variable 4.5/5

Counting Loops - You haven't quite said that the number of iterations is known in advance, although you have alluded to it. I would have liked to see a coded example here 3.5/5

Arrays - You have covered the main points, please include a coded example here, not just refer to the exercises elsewhere! 4/5

Part B

Exercise 1

Code-Annotation - A good attempt, but you need to pay attention to validation. The age validation is not quite correct. You need one while loop with 2 conditions, otherwise out of range values can slip through. Similarly, height can be negative. 3.5/5

Documentation You don't need to refer to language-specific things like "Set up programme library". The pseudo code should just refer to the logic of the program. Age is not fully tested, but generally good 3/5

Exercise 2

Code-Annotation - OK, but a bit limited as no validation of the values - first that they can't be negative and second that the sizes of the triangle and circle are suitable relative to each other, otherwise the triangle will exceed the bounds of the circle 3/5

Documentation - Breakdown is OK although again no mention of validation and the testing is limited to the "happy path" only, a wider range of test values is needed 2.5/5

Exercise 3

Code-Annotation - I'm mystified by the need for productname1, productname2 variables?! They don't appear to be used for anything?! Good otherwise save for the lack of validation on the price input - it shouldn't be allowed to be negative. 3.5/5

Documentation - Do not refer to implementation decisions eg int i etc. or for loop. Just "User enters 5 values" is enough. Testing is OK, but you have not tested negative values. 3 /5

Exercise 4

Code-Annotation

srand should not be within the loop it is only needed once per programme. Your if statements are crazily long(!) and could have been condensed. For example, if there are 2 5s, we don't care what the other entry is. Testing for the jackpot first would help you here. Nevertheless, a working implementation with a score accumulator 7.5/10

Documentation Same comment as before about including program implementation details like srand etc. Just break down the logic. No testing provided?! 4.5/10

Total: 42.5 /65 (65 %)

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
