Oliver Legg – 201244658 sgolegg
University of Liverpool

COMP281
Assignment 1 (2018)

1. Record marks (Problem 1013)

For this problem, I dealt with the scores by taking a user input in a while loop. If the input equals to 0, it exits the loop. If it doesn't equal to 1, it loops again. Instead of saving the scores, I saved the score boundaries and stored the in an array with a size of 3. The way this array works is like so:

If the score is in the upper boundary, the 0^{th} element is incremented.

>=85 = upper boundary
60-84 = medium boundary
<60 = lower boundary

Starting array is like so: [0, 0, 0].
Each position of the array works like so: [upper boundary, medium boundary, lower boundary]
If the user inputted 82, this would be in the upper boundary and the array would look like so [1, 0, 0]
This would continue forever until the user inputs 0.

On every loop of the while loop, an input() function is called which takes the input of the user and returns which index this is supposed to go in the array. The input function contains if statements which checks if the score is greater than 84, greater than 59 and lesser than 85 or lesser than 60.

After 0 has been inputted, I use a function which takes an int list (which is the scoreBoundary list) and outputs the list like so:

>=85:2
60-84:7
<60:2

2. Area and circumference of circles (Problem 1014)

I implemented this problem by getting 2 inputs and taking them as floats. I had to make sure that range 1 should be smaller than range 2 so I switched these around. Afterwards, I took the looped from range 1 to range 2 calculating the area sum and circumference sum like so:

$$area\ sum = \sum_{n=range1}^{range2} \pi \times n^2$$

$$circumference\ sum = \sum_{k=range1}^{range2} 2 \times \pi \times n$$

Afterwards, I outputted the answer to 3 significant figures just like the example shown in the assignment requirements.

Oliver Legg – 201244658 sgolegg
University of Liverpool

3. ASCII Code (Problem 1015)

Knowing how to store the values inputted into the program was the main difficulty of this problem. This was because the user was able to keep inputting anything until they inputted an EOF. Because of this, I was unable to use an int list or char list as this would require knowing the list size. To solve this, I thought it was necessary to use a linked list which I learnt how to implement from https://www.learn-c.org/en/Linked_lists from this, I stored the integer values inputted into the program and ran through the list outputting the char. The linked list required me to import `<stdlib.h>`. Outputting was easily done with the printf function as the %c had the ability of converting denary to ascii.

I had the outputting in a separate function which took the parameters of the head of the linked list and returned nothing but gave side effects of the printf. EOFs were given by ctrl+z on windows as learnt from this stacked overflow post: https://stackoverflow.com/questions/11968558/how-to-enter-the-value-of-eof-in-the-terminal

4. Largest common factor and smallest common multiple (Problem 1025)

I began to solve this problem by taking 2 integer numbers, I then made 2 functions. One of the functions return the largest common factor and the second returns the smallest common multiple. Both these functions took 2 integer numbers and returned 1 integer number. The largest common factor uses a while loop to whittle down the integer number lower and lower until num2 modulus num1 equals to 0. The smallest common multiple function divides num1 by the lcm times by num 2. Like so:

$$\text{scm}(\text{num1}, \text{num2}) = \frac{num1}{lcf(num1, num2)} \times num2$$

The lcf and scm is then outputted using the printf() function.

5. Precise division (Problem 1030)

This problem was solved by getting 3 inputs from the user. 2 of the numbers are divided, lets call them number1 and number2. I stored them as a data type ‘double’ as variables. Then there is the digit number to get the n’th decimal point from the division of both numbers. The digit number only had to be an integer. I made a function to get the digit which took the answer of the division.

The function has a loop which loops the number of times according to the n’th decimal point minus 2. Every loop the following math is carried out:

```
        n = n * 10;
        n = n - ((int) n);
```

This removes a digit from the left of a number for example. 1 iteration of ‘0.234’ would be converted to ‘0.34’. This keeps going until the loop is finished. And then multiplies the final answer by 10 to get a positive number and is converted to an integer to get a whole number. The answer is then printed out as an integer.