13 Feb 2023

COMP281 2021-22 – Assignment 1

- In the following, you will find the problems that constitute Assignment 1. They also appear on the canvas site for COMP281 as Assignment 1.
- You need to write a C program (not C++ or C#) that solves each problem – it must read the input, as specified in the problem description then print the solution to the given problem for that input.
    - Note that code is "correct" only if it **correctly implements a solution to the problem stated** in the assignment, not "if CodeGrade accepts it".
    - That is, even if CodeGrade accepts your code, it could be wrong. Read the problems carefully.
    - Make certain that your coding is clear so that it may be easily understood by the assessor (i.e. use spaces and blank lines appropriately). The final mark for the assignment will be based on whether your programs pass Code Grade's auto tests, but also the assessor's judgement on the code you've submitted.

- Input is read from the standard input, in the same way that you read input from the keyboard as shown in lectures (e.g., using scanf). Output is also printed to the standard output, as you have seen (e.g., using printf).

- For this set of problems, you must not use C's string handling library string.h and the "types" library ctype.h

- Do not include any additional prompt messages in your programs. The appearance of such text in the output from your code will mean that it cannot possibly match the output that Code Grade is checking for. N.B. For the week 2 assignment, this was not the case (CodeGrade was set to ignore additional outputs. That feature has been switched off for this assignment).

- You can work on the programs using e.g. Cygwin, running them on your own or a departmental computer, or by developing them directly on Code Grade using its interactive editor. When you are satisfied that your C programs work correctly, you must submit them through Code Grade. Even if they do not work fully it is worth submitting something in order to obtain some marks when the assessors look at them.

- You must also submit a brief report describing your solutions to the problems. This should be maximum two sides of A4 paper and should give a description of how each of your solutions works. This should include describing the algorithm used to reach the solution, describing your use of any C language features (that were not discussed in lectures) and identifying any resources that you have used to help you solve the problems. A separate canvas assignment called "Assignment 1 brief report" has been setup for this.

- This assignment is worth 50% of the total mark for COMP281.
    - All five problems in this assignment are weighted equally.
    - For each problem, you can earn a total of 20 points
        - 10 points for "Functionality and Correctness" awarded for programs that **correctly** solve the problem for all test cases.
        - 8 points for "Programming style, use of comments, indentation and identifiers" awarded depending on the style, comments, efficiency of the solution and use of appropriately named variables etc.
        - 2 points for the quality and depth of the accompanying report
    - The final grade results from normalising the earned points to a scale of 100.
    - See separate "comp281-detailed-marking-guidelines.pdf" for more details.

Submission Instructions

- Submit your solution to each part of the assignment via Canvas and don't forget to include your brief report.
- The **deadline** for this assignment submission is **27-Feb-2023 at 12:00.**
- Penalties for late submission apply in accordance with departmental policy as set out in the student handbook, which can be found at: http://intranet.csc.liv.ac.uk/student/ug-handbook.pdf

Part 1

Title: Area and circumference of circles

Description

In this exercise you must compute the area and circumference of a series of circles and output their sum. Specifically, the program will take the radius of two circles as input (r1 <= r2, both integers) and will output the sum of the areas and the circumferences of all circles starting with r1 and increasing at each step the radius by '1' until radius r2 has been reached, i.e. circles of radii r1, r1+1, …, r2.

Remember that the area of a circle equals Pi*r^2 and the circumference equals 2Pi*r.

Set Pi to 3.14

Input

Two integers r1 and r2 with r1<=r2.

Output

Two floats, sum_of_areas and sum_of_circumferences.

The result should be to 3 digits precision.

Sample Input

```
3 4
```

Sample Output

```
78.500
43.960
```

Part 2

Title: Count characters in a string

Description

Input a sequence of ASCII characters (aka a string). Count the numbers of 1) English characters; 2) digits; 3) spaces; 4) other characters. Note: Do not use the functions provided in string.h or ctype.h

Input

String

Output

number_of_english_characters number_of_digits number_of_spaces number_of_other_characters

Sample Input

```
aklsjflj123 sadf918u324 asdf91u32oasdf/.';123
```

Sample Output

```
23 16 2 4
```

Hint

Like all functions in C, scanf returns a value. In fact it returns the number of items read in successfully. When there is no more input, scanf returns the value EOF, which you can check for in your code.

If you're testing this yourself directly on your own computer, you will find it easier to put the input data into a file and redirect it into the program when you wish to test it e.g. ./assignment1_2 < inputdata.txt

Be aware of reading in and processing any newline character at the end of the input string. When a single line of input data is provided by Code Grade it will usually not include a newline, but on your own computer, it can be hard to avoid having one. If you are using a file containing input data to test your program, avoid a newline by not pressing return at the end of the line of characters you put into the file.

Part 3

Title: A+B For Large Integers

Description

Input 2 positive integers a and b (one each per line). Both a and b can be very large (up to 99 digits). Output a+b (you are not allowed to use any floating point data types, including float, double and long double). Do not use any functions in string.h

Input

Two integers a and b

Output

The sum of a+b

Sample Input

```
11111111111111111111111111110123
222222222222222
```

Sample Output

```
11111111111111111113333333333332345
```

Part 4

Title: Precise division

Description

8/13=0.615384615384615384615384...

For 8/13, the 6-th digit after the decimal point is 4.

Given three positive integers a, b, and n (all at most 60000), you are asked to compute a/b and print out the n-th digit after the decimal point.

Input

a b n

Output

The n-th digit after the decimal point of a/b.

Sample Input

```
8 13 6
```

Sample Output

```
4
```

Part 5

Title: ASCII code

Description

Convert a series of integers (representing ASCII codes) to characters.

You are not allowed to use system functions other than those in stdio.h

Input

A list of positive integers separated by whitespaces (spaces, newlines, tabs). Input ends with EOF.

Output

A list of characters.

Sample Input

```
72 101 108 108 111 44 32 119 111
114   108 100 33
```

Sample Output

```
Hello, world!
```

Hint

Remember that scanf returns a value (usually discarded in our programs) which indicates how many items were successfully scanned. You can use this to check for EOF (EOF is end-of-file and means no more input).