

# COMP281 2024-25 – Assignment 1

- In the following, you will find the tasks 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 task – 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 tasks 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 to improve legibility). 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 tasks, you must not use C’s string handling library `string.h` and the “types” library `ctype.h`. Certain tasks may stipulate additional limitations. Read the task definition carefully.
- **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 university 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.
- This assignment is worth 50% of the total mark for COMP281.
  - All five tasks in this assignment are weighted equally.
  - For each task, 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.
    - 10 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.
  - 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 as a separate file e.g. assignment1\_1.c, assignment1\_2.c ... assignment1\_5.c
- The **deadline** for this assignment submission is **16-Feb-2026 at 23:59**.
- The standard UoL policy regarding late submissions applies to this assessment.
  - Every student may submit up to 7 calendar days after the deadline, without penalty.
  - Students who have a Student Support Information Sheet (SSIS) that includes the "coursework deadlines extensions allowed" adjustment can make use of a further 7 day extension period (total of 14 calendar days after the deadline).
  - Any attempt to submit more than 7 days (or 14 days with an adjustment) after the deadline will be classed as non-submission and will receive a mark of zero.

## Task 1

### Title: Area and circumference of circles & The Area of Their Inscribed Squares

#### Description

In this exercise you must compute the area and circumference of a series of circles, the area of the associated inscribed squares and output their sums. Specifically, the program will take the radius of two circles as input ( $r_1 \leq r_2$ , both integers) and will output the sum of the areas and the circumferences of all circles and the sum of the areas of all the inscribed squares starting with  $r_1$  and increasing at each step the radius by '1', until radius  $r_2$  has been reached, i.e. circles of radii  $r_1, r_1+1, \dots, r_2$ .

In your program, set the value of  $\pi$  to 3.14

Note: You cannot use the maths library (and hence do not include the maths header file `<math.h>`).

#### Input

Two integers  $r_1$  and  $r_2$  with  $r_1 \leq r_2$ .

$1 < r_1 < 200$     $1 < r_2 < 200$     $r_2$  is guaranteed to be greater than or equal to  $r_1$

#### Output

Three floats, `sum_of_areas`   `sum_of_circumferences`   `sum_of_areas_of_inscribed_squares`

Each result should be to 3 digits precision.

#### Sample Input

```
3 4
```

#### Sample Output

```
78.500
```

```
43.960
```

```
50.000
```

## Task 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.

```
gcc -o assignment1_2 assignment1_2.c
./assignment1_2.exe < inputdata.txt
```

(i.e. you feed the data into your program, once it's compiled).

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 typically not include a newline at the end, but on your own computer, it can be hard to avoid having one – which will lead to an incorrect character count. 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 string of characters you put into the file.

## Task 3

### Title: Reverse String

#### Description

Reverse a string. Use as few char arrays in your solution as possible to achieve the highest mark.

The length of the string is at most 200 characters.

Do not use any functions in string.h!

#### Input

A string of characters.

#### Output

The reversed version of the string.

#### Sample Input

```
I am a student
```

#### Sample Output

```
tneduts a ma I
```

## Task 4

### Title: Precise division

#### Description

$8/13=0.615384615384615384615384\dots$

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
```

## Task 5

### 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 100 digits). Output a+b (you are not allowed to use any floating-point data types, including float, double and long double or libraries other than stdio.h).

#### Input

Two integers a and b.

#### Output

The sum of a+b

#### Sample Input

```
111111111111111111111111111111110123
2222222222222222
```

#### Sample Output

```
111111111111111111111133333333332345
```