Block 1
Lab 2 – Selection & Iteration

It is highly recommended that you get your solutions checked (continuously).

Read through the tasks carefully! The solutions must follow the specifications/requirements in the tasks to be approved.

None of the tasks in lab 2 requires type checking of inputted data. That is, if the program should read two integers you can assume that the user enters integers and not for example characters.

Global variables, **`goto`**, `exit()` or **`static`** are not allowed in any of the labs (if you don't know what these things mean, you don't have to worry).

Remember to test your solutions properly.

When you are done with the labs you should get them checked by a lab assistant or teacher (highly recommended) as well as upload tasks 2.1, 2.2 and 2.3 to Canvas. The submission is done individually, and it is only the .c files that should be handed in (not the entire project).

Task 1 – The angle

Write a program that asks the user for a positive angle in degrees (integers). The program should tell if the angle is acute (less than 90 degrees), obtuse (more than 90 degrees) or right angle (exactly 90 degrees). If the angle is more than 180 degrees or less than 0 degrees, the program should print that the angle is incorrectly given.

Example run (user input in **bold**)

```
Enter a positive angle in degrees: 98
The angle is obtuse!
```

Another example (user input in **bold**)

```
Enter a positive angle in degrees: -2
Incorrect input.
```

Task 2 – Morse code

Write a program that asks for a number between 0 and 5 and prints the corresponding Morse code. Use a **`switch`**-statement. The program should print an error if a number outside of the interval 0-5 is read from the user.

Morse codes for 0 to 5 are:

1) `-----`
2) `.----`
3) `..---`
4) `...--`
5) `....-`
6) `.....`

<u>Example run (user input in **bold**)</u>

```
Enter a number: 3
Morse code: ...--
```

Task 3 – The number counter

Task 3.1

Write a program that repeatedly asks for positive integers until the user enters a negative integer. When this happens, the program should print the smallest and the largest of the entered numbers, the sum of all numbers and the average of all numbers. See the example run below.

The requested results should be calculated successively after each new value has been entered. Note that the final negative number should not be calculated in the results.

Tip: **Do one thing at a time**. If you try to think about everything that should be done you'll be overwhelmed. Remember to break down problems into smaller ones. A suggestion is to start by making the input loop (that quits when the user enters a negative value). Try and test that it works until you start calculating values. Carefully add one functionality at a time and test before moving on.

If something does not work as you expect, try the debugger and try to follow what happens.

(since we have not yet talked about arrays in the course - you are not allowed to use arrays in this solution)

Example run (user input in **bold**)

```
Welcome!
Enter a negative number to stop.

Enter an integer: 4
Enter an integer: 2
Enter an integer: 66
Enter an integer: 633
Enter an integer: 8
Enter an integer: -1

The smallest number is: 2
The biggest number is: 633
The sum of the numbers is: 713
The average value is: 142.600000
```

Task 3.2

Now extend the program so that the user can continue or exit the program. You can do this by putting a loop around everything you have written so far. The user should be able to type '1' to run it one more time or '0' to exit.

Example run (user input in **bold**)

```
… (from before)
The sum of the numbers is: 713
The average value is: 142.600000

Would you like to run the program again (1 for yes, 0 for no)? 1

Enter a negative number to exit input and print the result.

Enter an integer: 8
Enter an integer: 14
… (etc)

Would you like to run the program again (1 for yes, 0 for no)? 1
```