# C++ Programming: Judge Assignment 1 (JA1)

The following tasks should be submitted to the SoftUni Judge system, which will be open starting Saturday, 18 March 2017, 10:00 (in the morning) and will close on Sunday, 26 March 2017, 23:59. You will be provided with a link to the “contest” (where you will submit the assignment) later.

For this assignment, the code for each task should be a single C++ file, the contents of which you copy-paste into the Judge system.

Please be mindful of the strict input and output requirements for each task, as well as any additional requirements on running time, used memory, etc., as the tasks are evaluated automatically and not following the requirements strictly may result in your program’s output being evaluated as incorrect, even if the program’s logic is mostly correct.

You can use C++03 and C++11 features in your code.

Unless explicitly stated, any integer input fits into int and any floating-point input can be stored in double

The tasks here do have memory and time restrictions, but that is just to introduce the exam format. In this first assignment, the input data is small enough and the tasks simple enough that you don’t need to think about optimizing your code too much. Focus on code that works and is easy to read and modify vs. super-optimized code.

## Task 4 – Named Operations (JA1-Task-4-Named-Operations)

We have an imaginary machine that has 4 operations, each of which works on an array of positive integers, more specifically on a range of elements of that array, defined by an inclusive start and an exclusive end index, and returns an integer. In C++, each of the operations might look similar to this:

int operation(int array[], int startIndex, int endIndex);

Each of the operations has a number, the first being 0, the second 1, the third 2 and the fourth – 3. The operations are the following:

1. Calculates the sum of the elements from array[startIndex] to array[endIndex - 1]
2. Calculates the average of the elements from array[startIndex] to   
   array[endIndex - 1] as an integer, by ignoring the floating-point part.
   1. E.g. the average of the numbers **4** and **5** would be calculated as 4
3. Finds the minimum (smallest) of the elements from array[startIndex] to array[endIndex - 1]
4. Finds the maximum (largest) of the elements from array[startIndex] to array[endIndex - 1]

To use the operations, the user must first input an array of positive integers. Then, the user must define names for the operations. A user can define multiple names for the same operation. When names have been given for the operations, the user can start calling operations, by typing in the name of an operation followed by two numbers, defining the start and end index for the operation. When the user wants to stop doing operations, they input the word end and the machine prints out a summary of the operations and their results

Write a program which simulates our imaginary machine.

### Input

The first line will be the array of integer numbers, separated by single spaces.

The second line will contain a single number **P** – the number of operation names the user wants to input.

Each of the following **P** lines will contain an operation name, written with lowercase English letters (a-z) and the number of the operation which is being named, separated by a single space.

Each of the following lines will contain a single operation call. Each operation call will begin with a name of an operation (which was entered previously), a start index and an end index, separated by single spaces.

The last line of input will contain only the word **end**

### Output

A single line listing all the operations done, with the following format, where **C** is the number of all operation calls done and **operationString** is the info of an operation call (see below)

* [C]{operationString,operationString,…,operationString}
* i.e. [C]{ followed by an operationString for each call, in the order they were called from the console, comma-separated (if more than one), followed by }

The info of an operation call, called an **operationString**, has the following format, where **name** is the name of the operation, **startIndex** is the start index number, **endIndex** is the end index number and **result** is the integer value of the result of the operation:

* name(startIndex,endIndex)=result

### Restrictions

**0 < C < 10**

**0 < P < 20**

The input array will be at least **1** and at most **20** elements long

startIndex < endIndex

**startIndex** and **endIndex** will always be correct indices inside the input array

Operation names will be at least **1** and at most **20** lowercase English letters (**a-z**)

### Example I/O

|  |  |
| --- | --- |
| Example Input | Expected Output |
| 5 4 3 2 1  4  avg 1  sum 0  min 2  max 3  max 0 5  sum 0 5  sum 0 1  min 0 5  avg 0 2  sum 0 5  end | [6]{max(0,5)=5,sum(0,5)=15,sum(0,1)=5,min(0,5)=1,avg(0,2)=4,sum(0,5)=15} |
| 1 2 3 4  3  sumo 0  sum 0  maximum 3  sum 0 4  sumo 0 4  end | [2]{sum(0,4)=10,sumo(0,4)=10} |