# Hints for Project 1 B.

## Problem description

## Task B. Calc: A calculator program.



We want to make a **simple calculator that can add and subtract integers**, and will accept arbitrarily long mathematical formulas composed of symbols +, -, and non-negative integer numbers.

Imagine you have a file formula.txt with the summation formula such as:

100 + 50 - 25 + 0 + 123 - 1

If you redirect the file into your program, it should compute and print the answer:

$ ./calc < formula.txt

247

It may sound tricky, but it is actually easy to write such a program, and you already know all the needed tools. Just think carefully how to put it all together.

Specifically, write a program calc.cpp that reads from the cin a sequence of **one or more non-negative integers** written to be **added or subtracted**. Space characters can be anywhere in the input. After the input ends (end-of-file is reached), the program should compute and print the result of the input summation.

### Possible input for your program may look like this:

15

10 + 3 + 0 + 25

5+6- 7 -8 + 9 + 10 - 11

1 + 1 + 1 + 1 +

1 + 1 + 1 + 1 +

1 + 1 + 1 + 1 +

1 + 1 + 1 + 1

(Each of the inputs above is a separate file containing one single formula, even if it spans multiple lines.)  
The corresponding outputs should be: 15, 38, 4, and 16.

### A hint on how to handle possible space characters in the input:

You can use >> operator to read the numbers and the +/- characters, because >> will be skipping all spaces between the input terms. It is also suggested to use the char type for reading the +/- operator characters, not string, because it will work well even when numbers and the operator symbol are adjacent and not separated by spaces (such as in 10+5+3).

## Reading int from cin in C++

C++ will skip all spaces before an int to put value into an integer variable.

For example, given the following code:

#include<iostream>

using namespace std;

int main()

{

    //Prompt user to enter age.

    cout << "Enter your age: ";

    int age; //declare age as an int.

    cin >> age; //Initialize age by reading input from cin.

    //Output age.

    cout << "Your age is " << age << endl;

    //main method needs to return an int.

    //Since the code runs smoothly to the end,

    //return 0 to the caller of current method main.

    return 0;

}

Here is a sample run:

Enter your age:      35

Your age is 35

Note from that above run, there are spaces before 35 but they are (rightfully) omitted by cin. Hence we do not need to worry about spaces before integers; we only need to read and skip spaces **before** character ‘+’ or ‘-‘.

## Analysis of the problem

There are two types of symbols: one is operator, the other are operands.

For example, in 5+6- 7 -8 + 9 + 10 - 11, operands are 5, 6, …, 11, operator are + and -.

Let *operand* be an integer variable. To read an int into *operand* in C++, we use

cin >> operand;

Let ch (warning: operator is a keyword in C++ and cannot be a variable name) be a char variable. To read a character in ch in C++, we use

cin >> ch;

For consistence, we can rewrite the original formula as

**0+**5+6- 7 -8 + 9 + 10 - 11

Let variable sum be the current sum and is initialized to be **zero (see the first 0?)**.

Let ch be the character before the first original operand. Variable ch is initialized to be **‘+’**.

|  |  |  |  |
| --- | --- | --- | --- |
| round | read next operand | if ch is ‘+’, add operand to sum, otherwise,  if ch is ‘-‘, subtract operand from sum | skip all spaces until ‘+’ or ‘-‘ is reached, save it in ch |
| 1 | 5 | ch is ‘+’ (by initialization), run sum += operand; so sum is 0 + 5 = 5 | + |
| 2 | 6 | ch is ‘+’, run sum += operand; so sum is 5 + 6 = 11 | - |
| 3 | 7 | ch is ‘-‘, run sum -= operand; so sum is 11 – 7 = 4 | - |
| 4 | 8 | ch is ‘-‘, run sum -= operand; so sum is 4 - 8 = -4 | + |
| 5 | 9 | ch is ‘+’, run sum += operand; so sum is -4 + 9 = 5 | + |
| 6 | 10 | ch is ‘+’, run sum += operand; so sum is 5 + 10 = 15 | - |
| 7 | 11 | ch is ‘-‘, run sum -= operand; so sum is 15 - 11 = 4 |  |

## Pseudocode

Each round is a loop. Operations in each round are loop statements. Here is a pseudocode.

int sum = 0;

char ch = ‘+’;

int operand;

while (cin >> operand)

{

if ch is ‘+’

sum += operand;

else if ch is ‘-‘

sum -= operand;

**skip all spaces (if any) until a non-space character is encountered**

//comments: In this problem, non-space character other than digit can only be ‘+’ or ‘-‘.

}

report sum;

So, the remaining job is to implement the following code:

**skip all spaces (if any) until a non-space character is encountered.**

Hint: you can use cin >> ch; to read a char into ch. You need to use a repetition statement since there might be more than one space.

**Warning**: the file name needs to be named as calc.cpp.