# Hints for Project 1C and 1D

Project 1 is in <https://maryash.github.io/135/projects/project_01.html>.

Note that cin automatically skips spaces. That is, suppose ch is a char, with cin >> ch; and a space is **not** read into ch.

1. Project 1C needs to calculate values for multiple formulas separated by semicolon.

We may encounter a symbol with three possible values: '+', '-', and ';'.

* If it is ';', this means we can prepare for the next formula, so print out the sum for the current formula, reset sum to be zero, set ch to be '+'.
* If the character is '+' or '-', we need to read the following operand first before working on that operand. Hence, the handling of ‘+’ or ‘-‘ happens AFTER we extract the next operand.

1. Project 1D needs to work on +, -, and ^ (square operations).

The difficult part is to handle ^, which decides whether to add an operand by itself or by its square (if ^ follows the operand).

After reading an integer in variable operand (you may use a different name, so modify the code correspondingly), read the next character into variable ch.

* if ch is ‘^’, then add operand \* operand to sum.
* if ch is not ‘^’, it may be ‘+’, ‘-‘, ‘;’, or space characters.
* If we run code like 1C, then if we read the next symbol to check whether it is ‘^’ or not, if it is ‘^’, then we process it, otherwise, we will miss a valid symbol ‘+’.

Now ch is ‘+’ and is pointed by file read head.

5+ 6;

After putting ch back to input stream, file read head is pointing to 5.

5+ 6;

To make our code work no matter whether spaces follow operand or not, we backtrack one letter from ch, using putback method in cin. You may think this operation is like to rewind the file read head one letter to the left. Google “putback cin c++” to see how to use this method.

* After putting back ch to input stream, we can continue to search the input stream to get next non-space character, which is ‘+’ in the above example.
* Another approach is to peek a character first before removing it from input stream. Google “peek char c++”. Using this approach, you can avoid backtracking a character.
* Warning: need to use a different variable, say ch2, than the variable to hold operator ‘+’ or ‘-‘ before the current operand, to hold the possible ‘^’ after the current operand, otherwise, ‘^’, if exists, will replace the previously read ‘+’ or ‘-‘ symbol. However, ‘^’ cannot be used to work with an operand.
* The type of variable ch2 must be **char**, not int, otherwise, cin >> ch2; fails, since ‘^’ cannot be interpreted as an integer and the code exits.