HOMEWORK #5:

Brain Calculator

Due Thursday, April the 17th, 11:59.59pm

For this assignment, your '`main()`' function should be in a C++ file called '`braincalc.cpp`'. Remember to put your name and section at the top of your program file.
Your program should expect all input to come from '`cin`', and all your output should be to '`cout`'.

Problem

The Brains are superior! The Brains have better technology!. As an example, when dealing with simple calculations, the Brains do not need to waste their neurons with silly parentheses and baroque operator precedence. Instead, they use "Brain Notation" [1], in which the operator follows the operands. For example, instead of:

```
2 - (3 * 8) + (4 + (12 / 5)) * 6
```

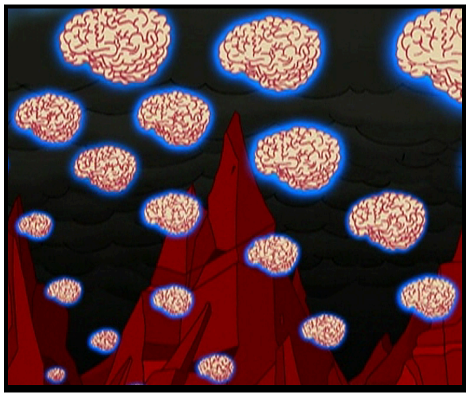
the Brains would use:

```
2 3 8 * - 4 12 5 / + 6 * +
```

saving themselves the energy needed to think about the parenthesis.

This super-human ability is possible because Brains organize their short term memory as a **stack**. When Brains read an *operand* they push it into their short-term memory stack. When they read an *operator*, they pop the necessary operands and push the intermediate result into the **stack**.

In order to defeat the Brains the next time they decide to invade Earth, is is important to study how their neurons work. The "Earth Defense Forces" have asked you to write a program that simulates the way Brains process integer expressions and create a "Brain Calculator". Your program will use a stack to simulate a Brains' brain.



Know thy enemy

Input

The input will consist of a series of integer expressions in Brain notation. Elements of the expression are separated by spaces. the character ‘$’ will denote the end of an expression. The character ‘#’ will denote the end of the input.

Output

For each expression, output the contents of the simulator stack (formated as in the sample). Note that when your program reads an operator, it should pop the corresponding number of operands off the stack, apply the operator, and push the result into the stack.
You will implement the following integer operators:

- Binary operators `+, -, *, /, %` with their usual meanings.
- Unary operator `!` negation... (Example... `5 !` produces `-5`)

Implementation Requirements / Details.

- Use a stack Data Structure to simulate the Brain’s computational processes.
- Your stack implementation should be a subclass of the provided “`AbstractStack`” class.
- All operands are integers.
- An expression may take more than one line.
- All expression elements are separated by at least a single space.

.

Sample

Input	Output
`4 3 + $`	`[ 7 ]`
`20 3 / $`	`[ 6 ]`
`5 2 - 3 * 99 $`	`[ 99, 9 ]`
`62`	`[ 2 ]`
`5 % $`	`[ 14 ]`
`2 3 8 * -`	`[ -22 ]`
`4 12 5 / +`	
`6 * + $`	
`22 ! $`	
`#`	

[1]
http://en.wikipedia.org/wiki/Reverse_Polish_notation

Hints:

Study the following code sample for handling the input:

```
#include <string>
#include <cstdlib>
#include <iostream>
using namespace std;
int main ()
{
  string s1, s2;
  int x, y;

  cin >> s1 >> s2;
  x = atoi( s1.c_str() );
  y = atoi( s2.c_str() );
  cout << s1 << " * " << s2 << " = " << x*y << endl;

  return 0;
}
```