Problem 2 - By Hugo Barahona

Prefix Stack in a Calculator

```
string operation[] = {"-", "+", "25", "30", "/", "*", "20", "50", "5"};

calculate(operation, 9);
```

For this problem I make use of a String array that contains the operators and the numbers in the prefix expression, and they are placed (individually) in each place within the array. Then I make use of a function called calculate, which parameters are a String array and the size of the given array.

```
void calculate(string op[], int size){
    stack<int> calc;//stack that will control all the numbers and pop them when necessary.
    int number1;//numbers to be used in every calculation.
    int number2;
    for(int i = size-1; i > -1; i--){
        if (op[i] == "+" || op[i] == "-" || op[i] == "/" || op[i] == "*") {
            number1 = calc.top();
            calc.pop();
            number2 = calc.top();
            calc.pop();
            if(op[i] == "+"){
                calc.push(number1+number2);
            } else if (op[i] == "-"){
                calc.push(number1-number2);
            } else if (op[i] == "/"){
                calc.push(number1/number2);
            } else {
                calc.push(number1*number2);
            }
        } else {
            calc.push(stoi(op[i]));
        }
    }
    cout << "The Result is: " << calc.top() << endl;
}
```

Within the calculate function we make use of a stack of int type, two int variables for the numbers that will be operated on, and the algorithm is basically the same as the one of postfix. However, this algorithm is reversed, instead of reading the expression from left to right, we are reading it from right to left. We push numbers in the stack and once we find an operator, we pop two numbers in the stack and those become our operands. Then, the result is pushed again within the stack and the process continues until we only have one number in the stack, which is the result.

Note: since the numbers are stored in a String format at first, we convert them into integers to store them in the stack.