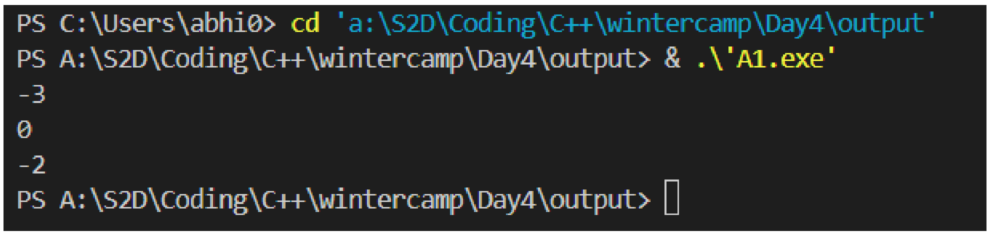
Winter Domain Camp Day-1

Name-Abhiraj Patel UID-22BCS11329 Date-19-12-24

Q.1. Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

```
#include <stack>
#include <iostream>
using namespace std;
class MinStack
{
private:
   stack<int> mainStack;
   stack<int> minStack;

public:
   MinStack() {}

   void push(int val)
   {
      mainStack.push(val);
      if (minStack.empty() || val <= minStack.top())
      {
         minStack.push(val);
      }
   }

   void pop()
   {
      if (mainStack.top() == minStack.top())
      {
            minStack.pop();
        }
        mainStack.pop();
    }

    int top()
    {
        return mainStack.top();
    }

    int getMin()
    {
        return minStack.top();
    }
};

int main()
{
    MinStack minStack;
    minStack.push(-2);
    minStack.push(0);
    minStack.push(-3);
    cout << minStack.getMin() << endl;
    minStack.pop();
    cout << minStack.top() << endl;
    cout << minStack.getMin() << endl;
    return 0;
}
```

Output



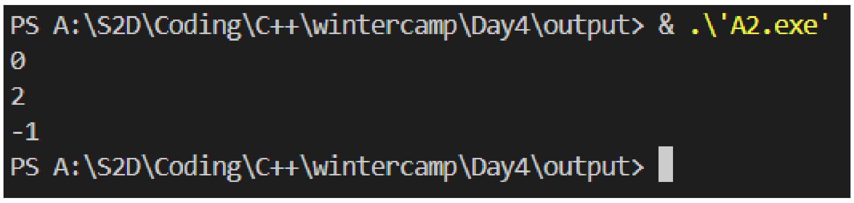
```
PS C:\Users\abhi0> cd 'a:\S2D\Coding\C++\wintercamp\Day4\output'
PS A:\S2D\Coding\C++\wintercamp\Day4\output> & .\'A1.exe'
-3
0
-2
PS A:\S2D\Coding\C++\wintercamp\Day4\output>
```

Q2 Given a string s, find the first non-repeating character in it and return its index.

If it does not exist, return -1.

```
#include <iostream>

#include <unordered_map>

#include <string>

using namespace std;


int firstUniqChar(string s) {

    unordered_map<char, int> freq;

    for (char c : s) {

        freq[c]++;

    }

    for (int i = 0; i < s.size(); i++) {

        if (freq[s[i]] == 1) {

            return i;

        }

    }

    return -1;

}


int main() {

    string s = "leetcode";

    cout << firstUniqChar(s) << endl;

    s = "loveleetcode";

    cout << firstUniqChar(s) << endl;
    s = "aabb";

    cout << firstUniqChar(s) << endl;

    return 0;

}
```

Output:



```
PS A:\S2D\Coding\C++\wintercamp\Day4\output> & .\'A2.exe'
0
2
-1
PS A:\S2D\Coding\C++\wintercamp\Day4\output>
```

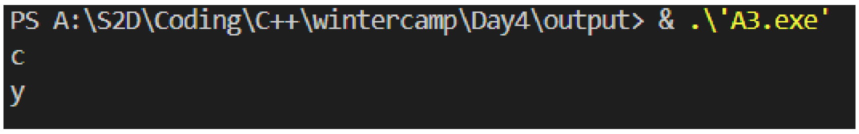
Q.3 Implement a simple text editor. The editor initially contains an empty string, S.Perform Q operations of the following 4 types:

- **append(W) - Append string W to the end of S.**
- **delete (k)- Delete the last k characters of S.**
- **print (k)- Print the k^th character of S.**
- **undo() - Undo the last (not previously undone) operation of type 1 or 2, reverting S to the state it was in prior to that operation.**

```
#include <iostream>

#include <vector>

#include <stack>

#include <string>

using namespace std;


class TextEditor {

private:

    string s;

    stack<string> undoStack;


public:
    TextEditor() {}

    void append(string w) {
        undoStack.push("1 " + w);
        s += w;
    }

    void deleteChars(int k) {
        string deleted = s.substr(s.size() - k);
        undoStack.push("2 " + deleted);
        s = s.substr(0, s.size() - k);
    }

    void print(int k) {
        cout << s[k - 1] << endl;
    }

    void undo() {
        if (undoStack.empty()) return;

        string operation = undoStack.top();
        undoStack.pop();

        if (operation[0] == '1') {
            s = s.substr(0, s.size() - operation.substr(2).size());
        } else if (operation[0] == '2') {
            s += operation.substr(2);
        }
    }
};
```

```
int main() {
    TextEditor editor;
    editor.append("abc");
    editor.print(3);   // c
    editor.deleteChars(3);
    editor.append("xy");
    editor.print(2);   // y
    editor.undo();
    editor.print(1);   // a
    return 0;
}
```

Output:



```
PS A:\S2D\Coding\C++\wintercamp\Day4\output> & .\'A3.exe'
c
y
```

Q.4 Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).

Implement the MyQueue class:

void push(int x) Pushes element x to the back of the queue.

int pop() Removes the element from the front of the queue and returns it.

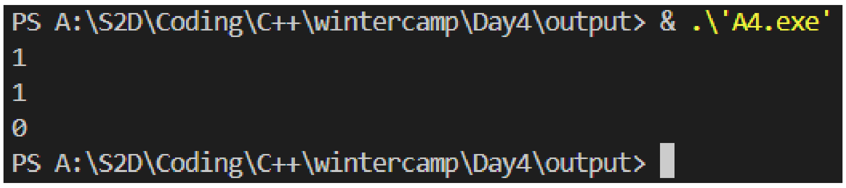
int peek() Returns the element at the front of the queue.

boolean empty() Returns true if the queue is empty, false otherwise.

```
#include <stack>
#include <iostream>
using namespace std;

class MyQueue {
private:
    stack<int> stack1, stack2;
```

```
public:
    MyQueue() {}

    void push(int x) {
        stack1.push(x);
    }

    int pop() {
        if (stack2.empty()) {
            while (!stack1.empty()) {
                stack2.push(stack1.top());
                stack1.pop();
            }
        }
        int val = stack2.top();
        stack2.pop();
        return val;
    }

    int peek() {
        if (stack2.empty()) {
            while (!stack1.empty()) {
                stack2.push(stack1.top());
                stack1.pop();
            }
        }
        return stack2.top();
    }

    bool empty() {
        return stack1.empty() && stack2.empty();
    }
};

int main() {
    MyQueue queue;
    queue.push(1);
    queue.push(2);
    cout << queue.peek() << endl;
    cout << queue.pop() << endl;
    cout << queue.empty() << endl;
    return 0;
}
```

Output:



```
PS A:\S2D\Coding\C++\wintercamp\Day4\output> & .\'A4.exe'
1
1
0
PS A:\S2D\Coding\C++\wintercamp\Day4\output>
```

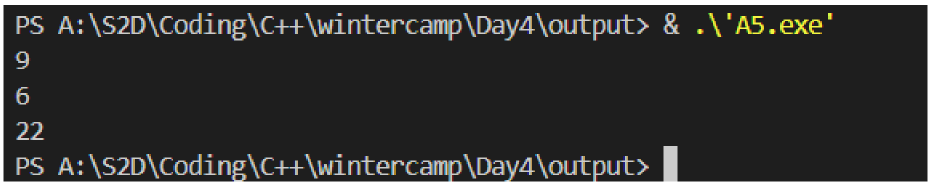
Q5. You are given an array of strings tokens that represents an arithmetic expression in a Reverse Polish Notation.

Evaluate the expression. Return an integer that represents the value of the expression.

```
#include <iostream>

#include <stack>

#include <vector>

#include <string>

#include <cstdlib>

using namespace std;


int evalRPN(vector<string>& tokens) {

   stack<int> stk;

   for (string& token : tokens) {

      if (token == "+" || token == "-" || token == "*" || token == "/") {

         int b = stk.top();

         stk.pop();

         int a = stk.top();

         stk.pop();


         if (token == "+") stk.push(a + b);

         else if (token == "-") stk.push(a - b);

         else if (token == "*") stk.push(a * b);

         else if (token == "/") stk.push(a / b);

      } else {

         stk.push(atoi(token.c_str()));

      }

   }

   return stk.top();
```

```cpp
}

int main() {
    vector<string> tokens = {"2","1","+","3","*"};
    cout << evalRPN(tokens) << endl;

    tokens = {"4","13","5","/","+"};
    cout << evalRPN(tokens) << endl;

    tokens = {"10","6","9","3","+","-11","*","/","*","17","+","5","+"};
    cout << evalRPN(tokens) << endl;

    return 0;
}
```

Output:



```
PS A:\S2D\Coding\C++\wintercamp\Day4\output> & .\'A5.exe'
9
6
22
PS A:\S2D\Coding\C++\wintercamp\Day4\output>
```