MAX

4.5 + 11 = 15.5/17

Excellent ☆

National University of Computer and Emerging Sciences (Lahore)			
Course:	Applied Programming		
Section:	MCS-1A	Semester:	Spring 2024
Duration:	40 minutes	TotalMarks:	~~20~~ 17
Name:	Muhammad Arslan Masroor	Roll no:	24L-7808

Question 1: Write a C++ function *printFibonacci(int n)* which prints the Fibonacci numbers till the *nth* term inclusively by using FIFO Queue data structure. You may assume that *Queue* data structure is already implemented.

Example:
Input: n = 7
Output: 0 1 1 2 3 5 8

```
void PrintFibonacci(int n)
{
    Queue<int> q;          (1)
    int num1 = -1;
    int num2 = 1;
    int num3 = 0;
    for(int i=0; i<n; i++)   (1)
    {
        num3 = num1 + num2;   (0.5)
        q.enqueue(num3)
        cout << q.dequeue() << " ";   (0.5)
        num1 = num2;
        num2 = num3;          (1)
    }
}
```

q

0	1	1	2	3	5	8

4.5/6

Algorithm is correct but does not use queues to output fibonacci as it should

Question 2: Write a C++ function *evalRPN(char tokens[], int n)*, where *n* is the size of the 1-D character array *tokens* of tokens that represents an arithmetic expression in Reverse Polish Notation (RPN) a.k.a. Postfix Notation. The function evaluates the given RPN expression and then returns an integer that represents the value, that is the answer, of that expression. You may use any number of helper functions and may assume the implementation of the LIFO *Stack* data structure.

Note:

- The valid operators are '+', '-', '*' and '/'.
- Each operand may be a single-digit integer or another expression.
- The division between two integers always truncates toward zero.
- There will not be any division by zero.
- The input represents a valid arithmetic expression in a reverse polish notation.

Example:

Input: tokens[5] = ["2","1","+","3","*"], n = 5
Output: 9
Explanation: ((2 + 1) * 3) = 9

```
int evalRPN(char tokens[], int n)
{
    Stack<int> st;
    for(int i=0; tokens[i] != '\0'; i++)
    {
        char ch = tokens[i];
        if( ch >= '0' && ch <= '9')
            st.push(ch - '0');
        else if ( ch == '+' || ch == '-' || ch == '*' || ch == '/')
        {
            int value2 = st.top();
            st.pop();
            int value1 = st.top();
            st.pop();
            switch( ch )
            {
                case '+':
                    st.push (value1 + value2);
                    break;
                case '-':
                    st.push (value1 - value2);
                    break;
                case '*':
                    st.push (value1 * value2);
                    break;
                case '/':
                    if (value2 != 0)
                        st.push (value1/value2);
                    break;
            }
        }
    }
    return st.top();
}
```

This is not a character array.

As a reverse for this.

next below box