

DATA STRUCTURES AND ALGORITHMS

Assignment # 2

Expression Evaluation using Stack

Student:

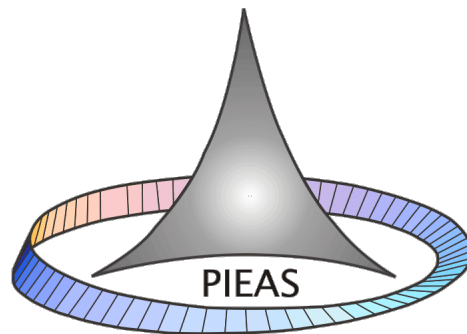
Assad Sultan

Ayesha Ali

Uswa Fatima

Instructor:

Mr. Tanveer Ali



April 28, 2020

Statement

You must implement Expression Evaluation System for space coordinate system $\langle x, y, z \rangle$ where x, y, z is of type real/float.

You must implement stack using STL. For driver/main function use expression, $((A-(B+C)*(A+C/A))^A)+B'$. Lastly you must generate two tables as per slides 53 and 69 against above said expression. Use $A=\langle 1, 2, 3 \rangle$, $B=\langle 1.1, 2.2, 3.3 \rangle$ and $C=\langle 3.3, 4.4, 9.9 \rangle$

Also

$$\langle a, b, c \rangle + \langle x, y, z \rangle = \langle a+x, b+y, c+z \rangle,$$

$$\langle a, b, c \rangle - \langle x, y, z \rangle = \langle a-x, b-y, c-z \rangle,$$

$$\langle a, b, c \rangle / \langle x, y, z \rangle = \langle a/x, b/y, c/z \rangle,$$

$$\langle a, b, c \rangle * \langle x, y, z \rangle = \langle a*x, b*y, c*z \rangle,$$

$$\langle a, b, c \rangle ^ \langle x, y, z \rangle = \langle a^x, b^y, c^z \rangle.$$

Solution

Source Code

```
#include <iostream>
#include <conio.h>
#include <vector>
#include <math.h>
#include <string>
using namespace std;

template <class TV> class Node
{
public:
    TV data;
    Node <TV>*link;
    Node <TV>()
    {
    };
};

//We are using template so that character stack and vector stack can be
implemented simultaneously
template <class T> class Stack {
private:
    Node<T> *top = NULL;

public:
    Stack()
    {
```

```

        top = NULL;
    }
    ~Stack()
    {
        delete top;
    }
    bool isempty()
    {
        if (top == NULL)
            return true; else
            return false;
    }
    void push(T value)
    {
        Node<T> *ptr = new Node<T>();
        ptr->data = value;
        ptr->link = top;
        top = ptr;
    }
    T pop()
    {
        if (isempty())
            cout << "Stack is Empty";
        else
        {
            Node<T> *ptr = top;
            top = top->link;
            return ptr->data;
            delete(ptr);
        }
    }
    void showTop()
    {
        if (isempty())
            cout << "Stack is Empty";
        else {
            for (int i = 0; i < 3; i++)
                std::cout << top->data.at(i) << ' ';
        }
    }
    T sTop()
    {
        if (isempty())
            cout << "Stack is Empty";
        else
        {
            return top->data;
        }
    }
    void displayStack()
    {
        if (isempty())
            cout << "Stack is Empty\n";
        else
        {
            Node<T> *temp = top;
            while (temp != NULL)
            {
                cout << "";
            }
        }
    }

```

```

        for (int i = 0; i < 3; i++)
            cout << "\t" << temp->data.at(i);
        cout << "\t|";
        temp = temp->link;
    }
    cout << "\n";
}

};
//operator precedence
int prec(char c)
{
    if (c == '^')
        return 3;
    else if (c == '*' || c == '/')
        return 2;
    else if (c == '+' || c == '-')
        return 1;
    else
        return -1;
}

//Function to convert Infix string to Postfix string
string infixToPostfix(string s)
{
    Stack<char> stackOfChar;
    stackOfChar.push('n');
    int l = s.length();
    string arrangedStack;
    for (int i = 0; i < l; i++)
    {
        if ((s[i] >= 'A' && s[i] <= 'Z') || (s[i] >= 'a' && s[i] <= 'z'))
            arrangedStack += s[i];
        else if (s[i] == '(')
        {
            stackOfChar.push('(');
        }
        else if (s[i] == ')')
        {
            while (stackOfChar.sTop() != 'n' && stackOfChar.sTop() != '(')
            {
                char c = stackOfChar.sTop();
                stackOfChar.pop();
                arrangedStack += c;
            }
            if (stackOfChar.sTop() == '(')
            {
                char c = stackOfChar.sTop();
                stackOfChar.pop();
            }
        }
        else {
            while (stackOfChar.sTop() != 'n' && prec(s[i]) <=
prec(stackOfChar.sTop()))
            {
                char c = stackOfChar.sTop();
                stackOfChar.pop();
                arrangedStack += c;
            }
            stackOfChar.push(s[i]);
        }
    }
}

```

```

    }
    while (stackOfChar.sTop() != 'n')
    {
        char c = stackOfChar.sTop();
        stackOfChar.pop();
        arrangedStack += c;
    }
    cout << "infix expression is: "<< arrangedStack << "\n\n";
    return arrangedStack;
}

int main()
{
    int choice, flag = 1;
    vector<double> A = { 1,2,3 }, B = { 1.1,2.2,3.3 }, C = { 3.3,4.4,9.9 },
ans{ 0,0,0 };
    Stack <char> ch;
    Stack <vector<double>> s;

    string expression = "((A-(B+C)*(A+C/A))^A)+B";
    string postfixExpression;
    postfixExpression = infixToPostfix(expression); //Conversion of Expression
from Infix to Postfix

    //Postfix String to char array conversion
    char e[16];
    for (int i = 0; i < sizeof(e); i++)
        e[i] = postfixExpression[i];

    //Postfix Array Evaluation
    cout << "Stack Flow:\n";
    for (int i = 0; i < 15; ++i)
    {
        s.displayStack();
        if (e[i] == 'A')
            s.push(A);
        else if (e[i] == 'B')
            s.push(B);
        else if (e[i] == 'C')
            s.push(C);
        else
        {
            vector<double> op1 = s.pop();
            vector<double> op2 = s.pop();
            switch (e[i])
            {
                case '+':
                    for (int i = 0; i<3; i++) {
                        op2[i] = op2[i] + op1[i];
                    }
                    s.push(op2);
                    break;
                case '-':
                    for (int i = 0; i < 3; i++) {
                        op2[i] = op2[i] - op1[i];
                    }
                    s.push(op2);
                    break;
                case '*':

```

```

        for (int i = 0; i < 3; i++) {
            op2[i] = op2[i] * op1[i];
        }
        s.push(op2);
        break;
    case '/':
        for (int i = 0; i < 3; i++) {
            op2[i] = op2[i] / op1[i];
        }
        s.push(op2);
        break;
    case '^':
        for (int i = 0; i < 3; i++) {
            op2[i] = pow(op2[i], op1[i]);
        }
        s.push(op2);
        break;
    }
}
s.displayStack();
ans = s.sTop();
cout << "\n\nInfix expression " << expression << " is: " << endl;
cout << "\nIt becomes postfix expression as " << e << " is: " << endl;
cout << "\nSolution is: " << e << " is: ";
for(int i=0;i<3;i++)
    cout << ans[i] << "\t";

_getch();
return 0;
}

```

Output

```

Infix expression is: ABC+ACA/+*-A^B+
Stack Flow:
Stack is Empty
1      2      3      |      1      2      3      |      1      2      3      |      1      2      3      |
1.1    2.2    3.3      |      1.1    2.2    3.3      |      1      2      3      |      1      2      3      |
3.3    4.4    9.9      |      1      2      3      |      4.4    6.6    13.2     |      4.4    6.6    13.2     |
4.4    6.6    13.2     |      1      2      3      |      1      2      3      |      1      2      3      |
1      2      3        |      4.4    6.6    13.2     |      1      2      3      |      4.4    6.6    13.2     |
3.3    4.4    9.9      |      1      2      3      |      4.4    6.6    13.2     |      1      2      3      |
1      2      3        |      3.3    4.4    9.9      |      1      2      3      |      4.4    6.6    13.2     |
3.3    2.2    3.3      |      1      2      3      |      4.4    6.6    13.2     |      1      2      3      |
4.3    4.2    6.3      |      4.4    6.6    13.2     |      1      2      3      |      1      2      3      |
18.92  27.72  83.16    |      1      2      3      |      1      2      3      |      1      2      3      |
-17.92 -25.72 -80.16   |      -17.92 -25.72 -80.16   |      -17.92 -25.72 -80.16   |      -17.92 -25.72 -80.16   |
1      2      3        |      1      2      3      |      1      2      3      |      1      2      3      |
-17.92 661.518 -515078 |      -17.92 661.518 -515078 |      -17.92 661.518 -515078 |      -17.92 661.518 -515078 |
1.1    2.2    3.3      |      1      2      3      |      1      2      3      |      1      2      3      |
-16.82 663.718 -515075 |      -16.82 663.718 -515075 |      -16.82 663.718 -515075 |      -16.82 663.718 -515075 |

Infix expression ((A-(B+C)*(A+C/A))^A)+B is:
It becomes postfix expression as ABC+ACA/+*-A^B+ is:
Solution is: ABC+ACA/+*-A^B+ is: -16.82 663.718 -515075

```

Figure 1: Output of the program

Infix to Postfix:

$((A - (B + C) * (A + C / A))^A) + B$

Steps	Input	stack	Postfix
1	((
2	(((
3	A	((A
4	-	((-	A
5	(((-(A

6	B	((-(AB
7	+	((-(+	AB
8	C	((-(+	ABC
9)		ABC+
10	*	((-*(ABC+
11	(((-*(ABC+
12	A	((-*(ABC+A
13	+	((-*(+	ABC+A
14	C	((-*(+	ABC+AC
15	/	((-*(+/ 	ABC+AC
16	A	((-*(+/ 	ABC+ACA
17)		ABC+ACA/+
18)		ABC+ACA/+*-
19	^	(^	ABC+ACA/+*-
20	A	(^	ABC+ACA/+*-A
21)		ABC+ACA/+*-A^
22	+	+	ABC+ACA/+*-A^
23	B	+	ABC+ACA/+*-A^B
24	final		ABC+ACA/+*-A^B+

ABC+ACA/+*-A^B+

Solution:

Input	Op1	Op2	value	stack
A	--	--	--	A
B	--	--	--	A B
C	--	--	--	A B C
+	B	C	B+C	A (B+C)
A	--	--	----	A (B+C) A
C	--	--	--	A (B+C) A C
A	--	--	--	A (B+C) A C A
/	C	A	C/A	A (B+C) A C/A
+	A	C/A	A+C/A	A (B+C) (A+C/A)
*	B+C	A+C/A	(B+C)*(A+C/A)	A ((B+C)*(A+C/A))
-	A	(B+C)*(A+C/A)	A-(B+C)*(A+C/A)	A-(B+C)*(A+C/A)
A	--	--	--	A-(B+C)*(A+C/A) A
^	A	A-(B+C)*(A+C/A)	A^(A-(B+C)*(A+C/A))	(A-(B+C)*(A+C/A))^A
B	--	--	--	((A-(B+C)*(A+C/A))^A) B
+	B	A^ A- (B+C)*(A+C/A)	B+(A^(A- (B+C)*(A+C/A)))	((A- (B+C)*(A+C/A))^A)+B

Input	Op1	Op2	value	stack
1,2,3	--	--	--	1,2,3
1.1,2.2,3.3	--	--	--	1,2,3 1.1,2.2,3.3
3.3,4.4,9.9	--	--	--	1,2,3 1.1,2.2,3.3 3.3,4.4,9.9
+	1.1,2.2,3.3	3.3,4.4,9.9	4.4,6.6,13.2	1,2,3 4.4,6.6,13.2
1,2,3	--	--	----	1,2,3 4.4,6.6,13.2 1,2,3
3.3,4.4,9.9	--	--	--	1,2,3 4.4,6.6,13.2 1,2,3 3.3,4.4,9.9
1,2,3	--	--	--	1,2,3 4.4,6.6,13.2 1,2,3 3.3,4.4,9.9 1,2,3
/	3.3,4.4,9.9	1,2,3	3.3,2.2,3.3	1,2,3 4.4,6.6,13.2 1,2,3 3.3,2.2,3.3
+	1,2,3	3.3,2.2,3.3	4.3,4.2,6.3	1,2,3 4.4,6.6,13.2 4.3,4.2,6.3
*	4.4,6.6,13.2	4.3,4.2,6.3	18.92,27.72,83.16	1,2,3 18.92,27.72,83.16
-	1,2,3	18.92,27.72,83.16	-17.92,-25.72,-80.16	-17.92,-25.72,-80.16
1,2,3	--	--	--	-17.92,-25.72,-80.16 1,2,3
^	-17.92,-25.72,-80.16	1,2,3	-17.92,661.518,-515078	-17.92,661.518,-515078
1.1,2.2,3.3	--	--	--	-17.92,661.518,-515078 1.1,2.2,3.3
+	1.1,2.2,3.3	17.92,661.518,-515078	-16.82,663.718,-515075	-16.82,663.718,-515075