ASSIGNMENT 2 – STACK PREFIX AND POSTFIX

```
// Stack prefix and postfix stuff
#include <iostream>
using namespace std;

class stack
{
    char data[20];
    int top;

public:
    stack()
    {
        top = -1;
    }

    char topdata()
    {
        return (data[top]);
    }

    void push(char x)
    {
        data[++top] = x;
    }
```

```
    char pop()
    {
        return (data[top--]);
    }

    int empty()
    {
        if (top == -1)
            return 1;
        return 0;
    }

    int full()
    {
        if (top == 19)
            return 1;
        return 0;
    }
};

int precedence(char x)
{
    if (x == '(')
        return 0;

    if (x == '+' || x == '-')
        return 1;

    if (x == '*' || x == '/' || x == '%')
        return 2;
```

```
    return 3;
}

void infix_postfix(char infix[20], char
postfix[20])
{
    stack s;
    int i = 0;
    int j = 0;
    char token, x;

    for (i = 0; infix[i] != '\0'; i++)
    {
        token = infix[i];
        if (isalnum(token))
        {
            postfix[j] = token;
            j++;
        }

        else
        {
            if (token == '(')
            {
                s.push(token);
            }
            else if (token == ')')
            {
                while ((x = s.pop()) != '('
                {
```

```
                    postfix[j] = x;
                    j++;
                }
            }
            else
            {
                while ((s.empty()) != 1 &&
((precedence(token)) <=
(precedence(s.topdata()))))
                {
                    postfix[j] = s.pop();
                    j++;
                }
                s.push(token);
            }
        }
    }
    while ((s.empty()) != 1)
    {
        postfix[j] = s.pop();
        j++;
    }
    postfix[j] = '\0';
}

void reverse(char a[20], char b[20])
{
    int i, j = 0;
    for (i = 0; a[i] != '\0'; i++)
    {
    }
```

```
    i--;

    for (j = 0; i >= 0; j++, i--)
    {
        if (a[i] == '(')
        {
            b[j] = ')';
        }
        else if (a[i] == ')')
        {
            b[j] = '(';
        }
        else
        {
            b[j] = a[i];
        }
    }
    b[j] = '\0';
}

void infix_prefix(char infix[20], char
prefix[20])
{
    char pre[20], in[20];
    reverse(infix, in);
    infix_postfix(in, pre);
    reverse(pre, prefix);
}

int main()
{
```

```
    char infix[20], postfix[20], prefix[20];
    cout << "Enter infix: ";
    cin >> infix;

    infix_postfix(infix, postfix);
    cout << "\n postfix expression: ";
    cout << postfix;

    infix_prefix(infix,  prefix);
    cout << "\n prefix expression: ";
    cout << prefix;
}

//OUTPUT
Enter infix: 2*3/(2-1)+5*3

postfix expression: 23*21-/53*+
prefix expression: +*2/3-21*53
```

```
// Prefix and postfix expression evaluation
//ASSIGNMENT 2B

#include <iostream>
using namespace std;

class stack
{
    char data[20];
    int top;

public:
    stack()
    {
        top = -1;
    }
    char topdata()
    {
        return (data[top]);
    }
    void push(int x)
    {
        data[++top] = x;
    }
    char pop()
    {
        return (data[top--]);
    }
    int empty()
    {
        if (top == -1)
            return 1;
        return 0;
    }
```

```
    int full()
    {
        if (top == 19)
            return 1;
        return 0;
    }
};
int evaluate(int op1, int op2, char op)
{
    if (op == '+')
        return op1 + op2;
    if (op == '-')
        return op1 - op2;
    if (op == '*')
        return op1 * op2;
    if (op == '/')
        return op1 / op2;
    if (op == '%')
        return op1 % op2;
}

void evaluate_postfix(char postfix[20])
{
    stack s;
    int i, op1, op2, result;
    char token;
    int x;
    for (i = 0; postfix[i] != '\0'; i++)
    {
        token = postfix[i];
        if (isalnum(token))
        {
            cout << "ENTER THE VALUE OF " << token <<
": ";
```

```
            cin >> x;
            s.push(char(x));
        }
        else
        {
            op2 = s.pop();
            op1 = s.pop();
            result = evaluate(op1, op2, token);
            s.push(char(result));
        }
    }
    result = s.pop();
    cout << "result: " << result;
}
void evaluate_prefix(char prefix[20])
{
    stack s;
    int i, op1, op2, result;
    char token;
    int x;
    for (i = 0; prefix[i] != '\0'; i++)
    {
    }
    i--;
    for (; i >= 0; i--)
    {
        token = prefix[i];
        if (isalnum(token))
        {
            cout << "Enter the value of " << token <<
": ";
            cin >> x;
            s.push(char(x));
        }
```

```
        else
        {
            op1 = s.pop();
            op2 = s.pop();
            result = evaluate(op1, op2, token);
            s.push(char(result));
        }
    }
    result = s.pop();
    cout << "result: " << result;
}
int main()
{
    char infix[20], token, postfix[20], prefix[20];
    cout << "\n postfix:";
    cin >> postfix;
    evaluate_postfix(postfix);

    cout << " ----------";

    cout << "\n prefix:";
    cin >> prefix;
    evaluate_prefix(prefix);
}
```

```
OUTPUT:
postfix:53+82-*
ENTER THE VALUE OF 5: 5
ENTER THE VALUE OF 3: 3
ENTER THE VALUE OF 8: 8
ENTER THE VALUE OF 2: 2
result: 48----------
```

```
 prefix:-*+4325
Enter the value of 5: 5
Enter the value of 2: 2
Enter the value of 3: 3
Enter the value of 4: 4
result: 9
```