Jaskirat Singh

2020CSC1008

STACK USING STL

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



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

// The main function to convert infix expression
//to postfix expression
string infixToPostfix(string s) {

   stack<char> st; //For stack operations, we are using C++ built in stack
   string result;

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

```
char c = s[i];

// If the scanned character is
// an operand, add it to output string.
if((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9'))
   result += c;

// If the scanned character is an
// ‘(‘, push it to the stack.
else if(c == '(')
   st.push('(');

// If the scanned character is an ‘)’,
// pop and to output string from the stack
// until an ‘(‘ is encountered.
else if(c == ')') {
   while(st.top() != '(')
   {
      result += st.top();
      st.pop();
   }
   st.pop();
}

//If an operator is scanned
else {
   while(!st.empty() && prec(s[i]) <= prec(st.top())) {
      result += st.top();
      st.pop();
   }
   st.push(c);
```

```
        }
    }

    // Pop all the remaining elements from the stack
    while(!st.empty()) {
        result += st.top();
        st.pop();
    }
    cout<<"Expression in postfixed form : "<<result<<endl;
    return result;
}


// The function calculate_Postfix returns the final answer of the expression after calculation
int calculate_Postfix(string  post_exp)
{
    stack <int> stack;
    int len = post_exp.length();
    // loop to iterate through the expression
    for (int i = 0; i < len; i++)
    {
        // if the character is an operand we push it in the stack
        // we have considered single digits only here
        if ( post_exp[i] >= '0' &&  post_exp[i] <= '9')
        {
            stack.push( post_exp[i] - '0');
        }
        // if the character is an operator we enter else block
        else
        {
            // we pop the top two elements from the stack and save them in two integers
```

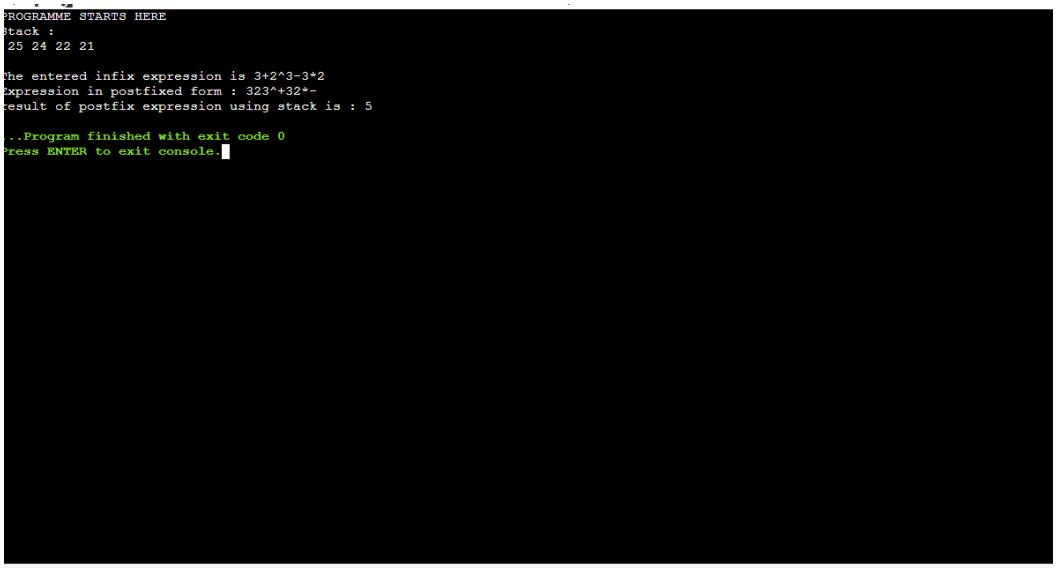
```
            int a = stack.top();
            stack.pop();
            int b = stack.top();
            stack.pop();
            //performing the operation on the operands
            switch (post_exp[i])
            {
               case '+': // addition
                        stack.push(b + a);
                        break;
               case '-': // subtraction
                        stack.push(b - a);
                        break;
               case '*': // multiplication
                        stack.push(b * a);
                        break;
               case '/': // division
                        stack.push(b / a);
                        break;
               case '^': // exponent
                        stack.push(pow(b,a));
                        break;
            }
         }
      }
      //returning the calculated result
      return stack.top();
}
//Driver program to test above functions
int main() {
  cout<<"PROGRAMME STARTS HERE"<<endl;
```

```
//Implementation of stack using STL
   stack<int> stack;
   stack.push(21);
   stack.push(22);
   stack.push(24);
   stack.push(25);
   cout<<"Stack : \n";
   while (!stack.empty()) {
      cout << ' ' << stack.top();
      stack.pop();
   }

   cout<<endl<<endl;

//functions to convert Infix to Postfix and evaluate Postfix using stl stack

   string exp = "3+2^3-3*2";
   cout<<"The entered infix expression is "<<exp<<endl;
   string str=infixToPostfix(exp);
   cout<<"result of postfix expression using stack is : "<<calculate_Postfix(str);
   return 0;
}
```


ROGRAMME STARTS HERE
tack :
25 24 22 21
he entered infix expression is 3+2^3-3*2
xpression in postfixed form : 323^+32*-
esult of postfix expression using stack is : 5
..Program finished with exit code 0
ress ENTER to exit console.