Instructions: Please read carefully

- Please rename this file as only your ID number **(e.g. 18-*****-1.doc or 18-*****-1.pdf).**
- Submit the file within the deadline in the Portal Lab Performance section labeled **Lab task 5. If you cannot complete the full task, do not worry. Just upload what you have completed.**

Code Instruction:

For both of the following problems, an operand is assumed to be a single digit. And an operator is limited to ‘**+**’, ‘**-**’, ‘*****’, ‘**/**’ (these 4 types). Also, for usage of parentheses, use only ‘(‘ for opening and ‘)’ for closing.

In light of these remarks, an algebraic expression for example can be written like below:

2*4+(6-3)/3

Follow the instructions from the next slide regarding how to approach the problems **1** and **2**.

1. Write C++ code to convert an infix algebraic expression to a postfix one using the help of Stack.

Your code here:

```
#include<iostream>
#include<conio.h>
#include<stack>
using namespace std;

bool isOperator(char ch)
{
 if(ch == '+' || ch=='-' ||ch == '*' || ch== '/' ||ch == '^')
    return true;
   else
   return false;
}

  int precedence(char ch)
   {

    if(ch == '*' || ch == '/')
    return 2;
    else if(ch == '+' || ch == '-')
    return 1;
    else
    return -1;
  }

  string convert(stack<char> s, string infix)
 {
   string postfix;
   for(int i=0;i<infix.length();i++)
   {
     /* s.push('(');
      int q=infix.length();
      infix[q]=')';*/
    if((infix[i] >= 'a' && infix[i] <= 'z')||(infix[i] >= 'A' && infix[i] <= 'Z') || (infix[i] >= '0' && infix[i] <= '9'))
   {
```

```
   postfix+=infix[i];
  }
  else if(infix[i] == '(')
  {
   s.push(infix[i]);
  }
  else if(infix[i] == ')')
  {
  while((s.top()!='(') && (!s.empty()))
 {
  char temp=s.top();
   postfix+=temp;
   s.pop();
  }
  if(s.top()=='(')
  {
  s.pop();
  }
  }
  else if(isOperator(infix[i]))
  {
   if(s.empty())
  {
  s.push(infix[i]);
  }
  else
  {
  if(precedence(infix[i])>precedence(s.top()))
  {
  s.push(infix[i]);
  }
  else if((precedence(infix[i])==precedence(s.top()))&&(infix[i]=='^'))
   {
    s.push(infix[i]);
   }
   else
  {
    while((!s.empty())&&( precedence(infix[i])<=precedence(s.top())))
      {
       postfix+=s.top();
        s.pop();
       }
       s.push(infix[i]);
     }
   }
 }
}
while(!s.empty())
{
postfix+=s.top();
s.pop();
}
```

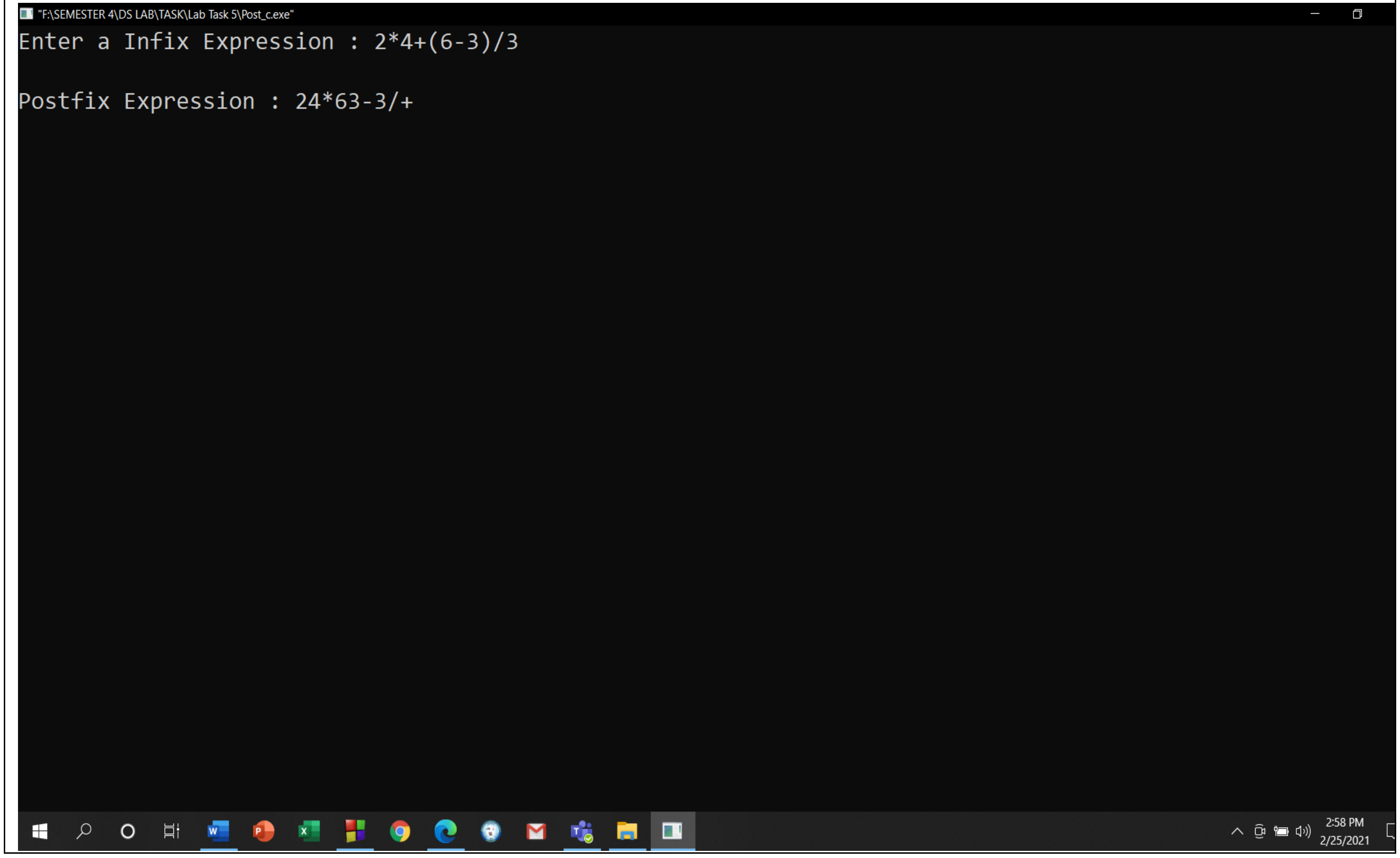
```
 return postfix;
}

int main()
{

            string infix, postfix;
            cout<<"Enter a Infix Expression : ";
            cin>>infix;
            stack <char> stack;
            cout<<endl<<"Postfix Expression : "<<convert(stack, infix);

 getch();
}
```

Your whole Screenshot here: (Console Output):



"F:\SEMESTER 4\DS LAB\TASK\Lab Task 5\Post_c.exe"

```
Enter a Infix Expression : 2*4+(6-3)/3

Postfix Expression : 24*63-3/+
```

2:58 PM
2/25/2021

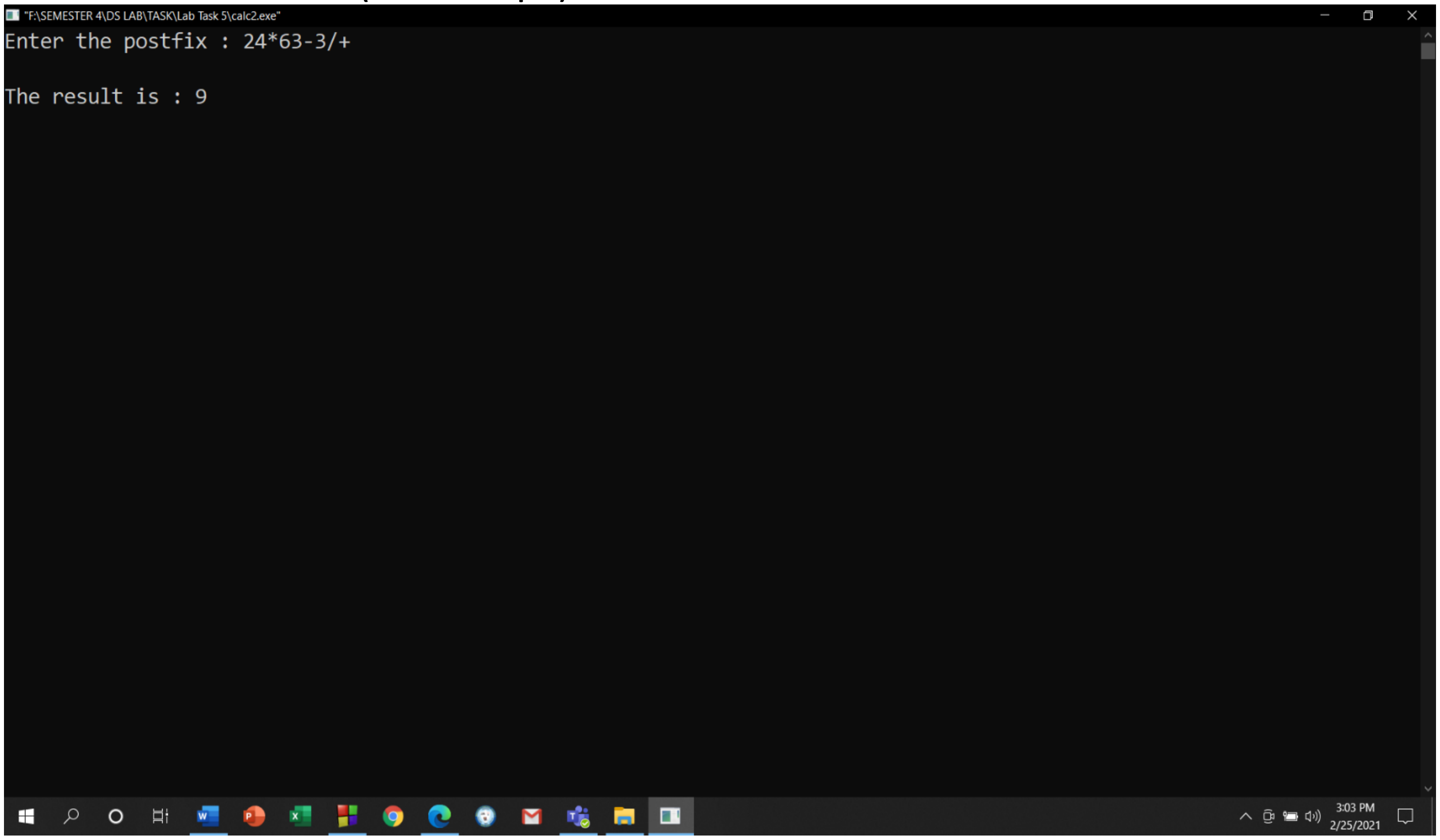
2. Write C++ code to evaluate a given postfix algebraic expression using the help of Stack.

Your code here:

```
#include<iostream>
#include<conio.h>
#define n 100
using namespace std;
class calculation {
	public:
		int st[n];
		int top;
		char str[n];
		calculation() {
			top = -1;
		}
		void push(int val) {
			top++;
			st[top] = val;
		}
		int pop() {
			int val = st[top];
			top--;
			return val;
		}
		int operation(int a,int b,char op) {
			switch(op) {
				case '+':return a+b;
				case '-':return a-b;
				case '*':return a*b;
				case '/':return a/b;
				default: return 0;
			}
		}
		int calc();
};
int calculation::calc() {
	int index = 0;
	while(str[index]!='\0') {
		if(isdigit(str[index])) {
			push(str[index]-'0');
		}
		else {
			int x = pop();
			int y = pop();
			int result = operation(y,x,str[index]);
			push(result);
		}
		index++;
	}
	return pop();
}
int main() {
	calculation cal;
```

```
        cout << "Enter the postfix : ";
        cin >> cal.str;
        cout << "The result is : " <<cal.calc();
        getch();
}
```

Your whole Screenshot here: (Console Output):



"F:\SEMESTER 4\DS LAB\TASK\Lab Task 5\calc2.exe"

```
Enter the postfix : 24*63-3/+

The result is : 9
```