Fatima Jinnah Women University

Name : Tanzeela Asghar
Roll no : 2021-BSE-032
Section : BSE (3A)
Submitted To : Sir Rehan Ahmed
Course : Data Structures And Algorithms

Lab # 04

TASK 1:

Give answers to the following.

1_Convert (manually) the following expressions to postfix.

• (A+B*D)/(E-F)+G:	ABD*EF-/G+
• A*(B+D)/E-F*(G+H/K) :	ABD+*E/FGHK/+*-

2. Convert the following infix expressions to prefix.

• A*B +(C/E) – (F+G)	+*AB /CE – +FG
• A+(B-D)/E-F*(G*H+K) :	+ A-/-BDE*F+*GHK

3. Evaluate the given Postfix expression and trace the contents of the Stack at each step using the standard evaluation algorithm.

2 7 3 - / 2 1 5 + * +

Symbols	Stack Content					
2						2
7					7	2
3				3	7	2

–					4	2
/						2
2					2	2
1				1	2	2
5			5	1	2	2
+				6	2	2
*					12	2
+						14

RESULT: 14

4. Convert the following expression from infix to postfix and show the contents of Stack and the output expression at each step.

(A+B) * C – D+F*G

Convert infix to postfix:

AB+ C – D*FG*+

SYMBOL infix	STACK CONTENT	OUTPUT EXPRESSION
(	(	
A		A
+	+	
B		B
)	)	
*	*	
C		C
-	-	
D		D
+	+	
F		F
*	*	
G		G

CODE 1:
Create a program to find weather a string in palindrome or not using stack
Palindrome is a word, phrase, or sequence that reads the same backwards as forwards
Example: pop, madam

```
#include<iostream>
#include<string>
using namespace std;
const int stack_capacity=100;
class stack
{
char array[stack_capacity];
int mytop;
public:
stack()
{
mytop=-1;
}
bool empty()
{
if(mytop==-1)
return true;
else
return false;
}
bool full()
{
if(mytop==stack_capacity)
return true;
else
return false; }
void push(int value) {
if (full())
cout<<"stack is full";
else
mytop++;
array[mytop] = value;
cout<< array[mytop]<<endl; }
char pop() {
if(empty())
{cout<<"stack is empty";
return mytop; }
else
cout<<"Pop value is :"<<array[mytop]<<endl;
mytop--; }
int top()
{
if(!empty())
```

```
{ return (array[mytop]); }
else
{ cout<<"stack is notempty return garbage value";
return (array[stack_capacity-1]);}}
void display()
{ cout<<"Stack element are :";
for(int i=mytop; i>=0; i--)
cout<<array[i]<<" ";
cout<<endl;
}};
int main()
{
int flag=0;
stack s;
string stg="ABC";
for(int i=0;i<stg.length();i++)
{
s.push(stg.at(i)); }
for(int i=0;i<stg.length();i++)
{
if(s.pop()==stg.at(i))
{flag=0;}
else
flag=1; }
if(flag==0)
cout<<"pal";
else
cout<<"not";
system("pause");
return 0;
 }
```

Output :

```
A
B
C
Pop value is :C
Pop value is :B
Pop value is :A
Press any key to continue . . .
not
[Program finished]
```

CODE 2:

Implement a program to convert the infix expression to postfix and display the result on screen.

```
#include<iostream>
#include<string>
using namespace std;
const int STACK_CAPACITY = 20;
class STACK
{
private:
int arr[STACK_CAPACITY];
int TOP_INDEX;
public:
STACK()
{
TOP_INDEX = -1;
}
bool EMPTY()
{
if (TOP_INDEX == -1)
return true;
else
return false;
}
void PUSH (int VALUE)
{
if (TOP_INDEX < STACK_CAPACITY - 1)
{
++TOP_INDEX;
arr[TOP_INDEX] = VALUE;
}
else
cout<<"STACK IS FULL, CAN'T ADD MORE VALUES"<<endl;
}
int TOP()
{
if (TOP_INDEX >= 0)
return arr[TOP_INDEX];
else
return 0;
}
int POP()
{
if (TOP_INDEX >= 0)
return arr[TOP_INDEX--];
else
return 0;
}
void DISPLAY()
```

```
{
if(TOP_INDEX >= 0)
{
cout<<"STACK ELEMENTS ARE:";
for (int i = TOP_INDEX; i >= 0; i--)
{
cout<<arr[i]<<" ";
}
}
else
cout<<"STACK IS EMPTY"<<endl;
}
};
string Infix_Postfix(string st);
bool Operator(char x);
bool Operand(char x);
int precedence (char x);
int main()
{
string abc;
cout<<"Enter Infix Expression \n";
cin>>abc;
string postfix = Infix_Postfix(abc);
cout<<"POSTFIX Expression = "<<postfix<<endl;
cout<<endl;
system("pause");
return 0;
}
string Infix_Postfix(string s)
{
STACK S;
string output;
for(int i = 0;i<s.length();i++)
{
if((s.at(i)>= 'a' && s.at(i) <= 'z')||(s.at(i)>= 'A' && s.at(i) <= 'Z'))
{
output+=s.at(i);
}
else if(s.at(i)=='(')
{
S.PUSH('(');
}
else if (Operator(s.at(i))==true)
{
if(precedence(s.at(i))<=precedence(S.TOP()))
{
char ch=S.TOP();
S.POP();
```

```
output+=ch;
}
S.PUSH(s.at(i));
}
else if(s.at(i)==')')
{
while(S.TOP()!='(')
{
char ch=S.TOP();
S.POP();
output+=ch;
}
if(S.TOP()=='(')
{
char c=S.TOP();
S.POP();
}
}
}
while(S.TOP()!='\0')
{
char ch=S.TOP();
S.POP();
output+=ch;
}
return output;
}
bool Operand(char x)
{
if(x >= '0' && x <= '9')
return true;
if(x >= 'a' && x <= 'z')
return true;
if(x >= 'A' && x <= 'Z')
return true;
return false;
}
bool Operator(char x)
{
if(x == '+' || x == '-' || x == '*' || x == '/' || x== '$')
return true;
return false;
}
int precedence(char op)
{
 if(op == '^')
return 3;
 else if(op == '*' || op == '/')
```

```
return 2;
 else if(op == '+' || op == '-')
return 1;
 else
return -1;
}
```

Output :