**Practical No: 03**

**Implement the following for Stack:**

1. **Write a program to implement the concept of Stack with Push, Pop, Display and Exit operations.**

**Code:**

#include<iostream>

#include<cstdlib>

using namespace std;

struct node{

int data;

struct node \*next;

};

struct node \*head=NULL;

void create(int v){

struct node \*new\_n=(struct node\*)malloc(sizeof(struct node));

new\_n->data=v;

new\_n->next=NULL;

if(head==NULL){

head=new\_n;

}

else{

new\_n->next=head;

head=new\_n;

}

}

void del(){

struct node \*t=head;

head=head->next;

free(t);

}

void display(){

struct node \*temp=head;

while(temp!=NULL){

cout<<temp->data<<" ";

temp=temp->next;

}

}

int main(){

int ch,c;

do{

cout<<"Enter\n1.Push\n2.Pop\n3.show\n4.Exit"<<endl;

cin>>ch;

switch(ch){

case 1:

int size,nm;

cout<<"Enter size of queue: ";

cin>>size;

for(int i=0;i<size;i++){

cout<<"Enter element: ";

cin>>nm;

create(nm);

}

cout<<"pushed element are: ";

display();

cout<<endl;

break;

case 2:

cout<<"After pop: ";

del();

display();

cout<<endl;

break;

case 3:

cout<<"Your final queue: ";

display();

break;

case 4:

exit(0);

break;

}

cout<<"Do you want to continue?(0/1)";

cin>>c;

}

while(c!=0);

}

**……………………………………………………………………………………….**

**Output:**

Enter

1.Push

2.Pop

3.show

4.Exit

1

Enter size of queue: 5

Enter element: 10

Enter element: 20

Enter element: 40

Enter element: 40

Enter element: 50

pushed element are: 50 40 40 20 10

Do you want to continue?(0/1)1

Enter

1.Push

2.Pop

3.show

4.Exit

2

After pop: 40 40 20 10

Do you want to continue?(0/1)1

Enter

1.Push

2.Pop

3.show

4.Exit

3

Your final queue: 40 40 20 10 Do you want to continue?(0/1)1

Enter

1.Push

2.Pop

3.show

4.Exit

4

**……………………………………………………………………………………….**

1. **Write a program to convert an infix expression to postfix and prefix conversion**
2. **Infix to postfix**

**Code:**

#include<iostream>

#include<stack>

#include<string>

using namespace std;

int checkOP(char op)

{

if(op=='+'||op=='-')

{

return 1;

}

if(op=='\*'||op=='/')

{

return 2;

}

return 0;

}

string infixtopostfix(string infix)

{

string postfix;

stack<char> st;

for(int ch=0;ch<infix.length();ch++)

{

char c=infix[ch];

if(isalnum(c))

{

postfix=postfix+c;

}

else if(c=='(')

{

st.push(c);

}

else if(c==')')

{

while(st.top()!='(')

{

postfix=postfix+st.top();

st.pop();

}

st.pop();

}

else{

while(!st.empty()&&checkOP(c)<=checkOP(st.top()))

{

postfix=postfix+st.top();

st.pop();

}

st.push(c);

}

}

while(!st.empty())

{

postfix=postfix+st.top();

st.pop();

}

return postfix;

}

int main()

{

cout<<"Your infix to postfix conversion is: "<<endl;

cout<<infixtopostfix("A+B\*C+D");

return 0;

}

Output:

**…………………………………………………………………………**

Your infix to postfix conversion is:

ABC\*+D+

**………………………………………………………………………….**

1. **Infix to prefix**

**Code:**

#include<iostream>

#include<stack>

#include<string>

#include<algorithm>

using namespace std;

string exp , postfix;

int checkOP(char op)

{

if(op=='+'||op=='-')

{

return 1;

}

if(op=='\*'||op=='/')

{

return 2;

}

return 0;

}

void infixtopostfix(string infix)

{

stack<char> st;

for(int ch=0;ch<infix.length();ch++)

{

char c=infix[ch];

if(isalnum(c))

{

postfix=postfix+c;

}

else if(c=='(')

{

st.push(c);

}

else if(c==')')

{

while(st.top()!='(')

{

postfix=postfix+st.top();

st.pop();

}

st.pop();

}

else{

while(!st.empty()&&checkOP(c)<=checkOP(st.top()))

{

postfix=postfix+st.top();

st.pop();

}

st.push(c);

}

}

while(!st.empty())

{

postfix=postfix+st.top();

st.pop();

}

}

int main()

{

cout<<"Your infix to prefix conversion is: "<<endl;

exp="A+B";

cout<<"Infix : "<<exp<<endl;

reverse(exp.begin(),exp.end());

infixtopostfix(exp);

reverse(postfix.begin(),postfix.end());

cout<<"Prefix : "<<postfix;

return 0;

}

**Output:**

**……………………………………………………………………………**

Your infix to prefix conversion is:

Infix : A+B

Prefix : +AB

**……………………………………………………………………………**

1. **Write a program to implement tower of Hanoi problem.**

**Code:**

#include<iostream>

using namespace std;

void toh(int n,char from\_rod,char to\_rod, char aux\_rod)

{

if(n==0)

{

return;

}

else

{

toh(n-1,from\_rod,aux\_rod,to\_rod);

cout<<"Move disk"<<" "<<n<<" "<<"from"<<" "<<from\_rod<<" "<<"to"<<" "<<to\_rod<<endl;

toh(n-1,aux\_rod,to\_rod,from\_rod);

}

}

int main()

{

int a,b,c;

toh(3,'a','b','c');

}

**Output:**

**………………………………………………………………………………..**

Move disk 1 from a to b

Move disk 2 from a to c

Move disk 1 from b to c

Move disk 3 from a to b

Move disk 1 from c to a

Move disk 2 from c to b

Move disk 1 from a to b

**……………………………………………………………………………….**