**MANGALORE UNIVERSITY**

**Bachelor of Computer Applications (BCA) Degree Programme**

**Choice Based Credit System (2019-2020 Onwards)**

**III Semester – Practicals**

**BCAP 234**

**Operating Systems and Data Structures Lab**

**Part - C:**

Implementations using C++

1. **Write a Program for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, \*, /, %( Remainder), ^ (Power) and alphanumeric operands.**

#include<iostream.h>

#include<conio.h>

#include<string.h>

int index=0,pos=0,top=-1,length;

char symbol,temp;

char infix[20],stack[20],postfix[20];

void push(char);

char pop();

void conversion(char[20],char[20]);

int preced(char);

void main()

{

clrscr();

cout<<"Enter the infix expression"<<endl;

cin>>infix;

conversion(infix,postfix);

cout<<"Infix expression is:"<<infix<<endl;

cout<<"Postfix expression is:"<<postfix<<endl;

getch();

}

void conversion(char infix[],char postfix[])

{

length=strlen(infix);

push('#');

while(index<length)

{

symbol=infix[index];

switch(symbol)

{

case'(':push(symbol);

break;

case')':temp=pop();

while(temp!='(')

{

postfix[pos]=temp;

pos++;

temp=pop();

}

break;

case'+':

case'-':

case'\*':

case'/':

case'%':

case'^':while(preced(stack[top])>=preced(symbol))

{

temp=pop();

postfix[pos]=temp;

pos++;

}

push(symbol);

break;

default:postfix[pos++]=symbol;

}

index++;

}

while(top>0)

{

temp=pop();

postfix[pos++]=temp;

}

return;

}

void push(char symb)

{

top++;

stack[top]=symb;

}

char pop()

{

char symb;

symb=stack[top];

top--;

return(symb);

}

int preced(char symb)

{

int p;

switch(symb)

{

case'^':p=3;

break;

case'\*':

case'/':

case'%':p=2;

break;

case'+':

case'-':p=1;

break;

case')':

case'(':p=0;

break;

case'#':p=-1;

break;

}

return(p);

}

/\*

**Output:**

Enter the infix expression

(a+b)\*(c+d)

Infix expression is:(a+b)\*(c+d)

Postfix expression is:ab+cd+\*

Enter the infix expression

a+b-c\*d/e%5^0

Infix expression is:a+b-c\*d/e%5^0

Postfix expression is:ab+cd\*e/50^%-

\*/

1. **Write a program to implement circular queue using array.**

#include<iostream.h>

#include<iomanip.h>

#include<conio.h>

#include<process.h>

#define ms 5

int cq[ms],front,rear;

class cqueue

{

public:

void insert();

void delet();

void display();

cqueue()

{

front=rear=-1;

}

~cqueue(){}

};

void cqueue::insert()

{

int ele;

if(front==(rear+1)%ms)

cout<<"Circular queue is full"<<endl;

else

{

cout<<"Enter the element:";

cin>>ele;

rear=(rear+1)%ms;

cq[rear]=ele;

if(front==-1)

front=0;

}

}

void cqueue::delet()

{

int ele;

if(front==-1)

cout<<"Queue is empty"<<endl;

else

{

ele=cq[front];

cout<<"Deleted element is "<<ele<<endl;

if(front==rear)

front=rear=-1;

else

front=(front+1)%ms;

}

}

void cqueue::display()

{

int i;

if(front==-1)

cout<<"Circular queue is empty"<<endl;

else

{

cout<<"Circular queue elements are :"<<endl;

if(front>rear)

{

for(i=front;i<=ms+1;i++)

{

cout<<cq[i]<<"\t";

for(i=0;i<=rear;i++)

cout<<cq[i]<<"\t";

}

}

else

{

for(i=front;i<=rear;i++)

cout<<cq[i]<<"\t";

}

cout<<endl;

}

}

void main()

{

cqueue obj;

int ch;

clrscr();

do

{

cout<<"\nCircular queue operation"<<endl;

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

cout<<"1.Insertion\n2.Deletion\n3.Display\n4.Exit"<<endl;

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

cout<<"Enter your choice"<<endl;

cin>>ch;

switch(ch)

{

case 1:obj.insert();

break;

case 2:obj.delet();

break;

case 3:obj.display();

break;

case 4:exit(0);

}

}while(ch>=1&&ch<=4);

getch();

}

/\*

**Output:**

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

1

Enter the element:1

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

1

Enter the element:2

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

1

Enter the element:3

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

3

1 2 3

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

2

Deleted element is 1

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

2

Deleted element is 2

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

3

Circular queue elements are :

3

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

1

Enter the element:4

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

1

Enter the element:5

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

1

Enter the element:6

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

1

Enter the element:7

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

1

Circular queue is full

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

3

Circular queue elements are :

3 4 5 6 7

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

4

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

3

Circular queue is empty

Circular queue operation

------------------------

1.Insertion

2.Deletion

3.Display

4.Exit

------------------------

Enter your choice

4

\*/

1. **Write a program to implement stack using linked list.**

#include<iostream.h>

#include<stdlib.h>

#include<conio.h>

#include<stdio.h>

#include<malloc.h>

typedef struct node

{

int data;

struct node \* link;

}node;

class stack

{

node \*top;

public:

stack()

{

top=NULL;

}

void push();

void pop();

void display();

};

void stack::push()

{

node\* temp;

temp= (node\*) malloc(sizeof(node));

cout<<"Enter the data: ";

cin>>temp->data;

temp->link= top;

top=temp;

}

void stack:: pop()

{

node \*temp;

temp= top;

if(top==NULL)

{

cout<<"Stack is empty"<<endl;

getch();

return;

}

temp= top;

top=top->link;

cout<<"Deleted item is "<<temp->data<<endl;

free(temp);

getch();

}

void stack::display()

{

node \*temp;

temp=top;

cout<<"Stack elements are "<<endl;

while(temp!=NULL)

{

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

temp=temp->link;

}

cout<<"NULL"<<endl;

getch();

}

int main()

{

stack s;

int choice=1,ch;

clrscr();

while(choice)

{

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

cout<<"Stack using linked list"<<endl;

cout<<"1.Push"<<endl;

cout<<"2.Pop"<<endl;

cout<<"3.Display"<<endl;

cout<<"Enter your choice: "<<endl;

cin>>choice;

switch(choice)

{

case 1:s.push();

break;

case 2:s.pop();

break;

case 3: s.display();

break;

default:

cout<<"Invalid choice"<<endl;

}

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

cin>>choice;

}

getch();

return 0;

}

/\*

**Otuput:**

----------

Stack using linked list

1.Push

2.Pop

3.Display

Enter your choice:

1

Enter the data: 1

Do you want to continue ? (1/0)

1

----------

Stack using linked list

1.Push

2.Pop

3.Display

Enter your choice:

1

Enter the data: 2

Do you want to continue ? (1/0)

1

----------

Stack using linked list

1.Push

2.Pop

3.Display

Enter your choice:

3

Stack elements are

2->1->NULL

Do you want to continue ? (1/0)

1

----------

Stack using linked list

1.Push

2.Pop

3.Display

Enter your choice:

2

Deleted item is 2

Do you want to continue ? (1/0)

1

----------

Stack using linked list

1.Push

2.Pop

3.Display

Enter your choice:

3

Stack elements are

1->NULL

Do you want to continue ? (1/0)

0

----------

Stack using linked list

1.Push

2.Pop

3.Display

Enter your choice:

2

Stack is empty

Do you want to continue ? (1/0)

1

----------

Stack using linked list

1.Push

2.Pop

3.Display

Enter your choice:

3

Stack elements are

NULL

Do you want to continue ? (1/0)

0

\*/

**4. Write a menu driven program for the following operations on Binary Search Tree (BST) of Integers (a) Create a BST of N Integers (b) Traverse the BST in Inorder, Preorder and Post Order (c) Search the BST for a given element (KEY) and report the appropriate message .**

#include<iostream.h>

#include<conio.h>

#include<iomanip.h>

#include<process.h>

struct node

{

int data;

struct node \*llink;

struct node \*rlink;

};

typedef struct node btree;

btree \*root;

class tree

{

btree \*newnode,\*cur,\*ptr;

public:

tree()

{root=NULL;}

void insert();

void inorder(btree\*);

void preorder(btree\*);

void postorder(btree\*);

void display(btree\*,int);

void search(btree\*);

};

void tree::insert()

{

int item;

newnode=new btree;

cout<<"Enter item to insert"<<endl;

cin>>newnode->data;

item=newnode->data;

newnode->llink=NULL;

newnode->rlink=NULL;

if(root==NULL)

root=newnode;

else

{

cur=root;

while(cur!=NULL)

{

ptr=cur;

if(item>=cur->data)

cur=cur->rlink;

else

cur=cur->llink;

}

if(ptr->data<=item)

ptr->rlink=newnode;

else

ptr->llink=newnode;

}

}

void tree::inorder(btree\*root)

{

if(root!=NULL)

{

inorder(root->llink);

cout<<root->data<<" ";

inorder(root->rlink);

}

}

void tree::preorder(btree\*root)

{

if(root!=NULL)

{

cout<<root->data<<" ";

preorder(root->llink);

preorder(root->rlink);

}

}

void tree::postorder(btree\*root)

{

if(root!=NULL)

{

postorder(root->llink);

postorder(root->rlink);

cout<<root->data<<" ";

}

}

void tree::search(btree\*root)

{

int f=0,key;

if(root==NULL)

cout<<"Binary search tree is empty";

else

{

cout<<"Enter the key element:";

cin>>key;

cur=root;

while(cur!=NULL)

{

if(cur->data==key)

{

f=1;

break;

}

if(key>cur->data)

cur=cur->rlink;

else

cur=cur->llink;

}

if(f==1)

cout<<"Element is found";

else

cout<<"Element not found";

}

}

void tree::display(btree\*newnode,int level)

{

int i;

if(root==NULL)

cout<<"Binary search tree is empty"<<endl;

else

{

if(newnode!=NULL)

{

display(newnode->rlink,level+5);

for(i=0;i<level;i++)

cout<<" ";

cout<<newnode->data<<endl;

display(newnode->llink,level+5);

}

}

}

void main()

{

int ch,level=1;

tree obj1;

clrscr();

do

{

cout<<"\nMenu"<<endl;

cout<<"1.Insert\n2.Inorder\n3.Preorder\n4.Postorder\n"

"5.Search\n6.Display\n7.Exit"<<endl;

cout<<"Enter your choice"<<endl;

cin>>ch;

switch(ch)

{

case 1:obj1.insert();

break;

case 2:obj1.inorder(root);

break;

case 3:obj1.preorder(root);

break;

case 4:obj1.postorder(root);

break;

case 5:obj1.search(root);

break;

case 6:cout<<"------------------"<<endl;

obj1.display(root,level);

break;

case 7:exit(0);

}

}while(ch>=1&&ch<=7);

getch();

}

/\*

**Output:**

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

1

Enter item to insert

18

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

1

Enter item to insert

11

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

1

Enter item to insert

25

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

1

Enter item to insert

2

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

1

Enter item to insert

9

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

1

Enter item to insert

13

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

1

Enter item to insert

20

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

6

------------------

25

20

18

13

11

9

2

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

2

2 9 11 13 18 20 25

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

3

18 11 2 9 13 25 20

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

4

9 2 13 11 20 25 18

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

5

Enter the key element:11

Element is found

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

5

Enter the key element:10

Element not found

Menu

1.Insert

2.Inorder

3.Preorder

4.Postorder

5.Search

6.Display

7.Exit

Enter your choice

7

\*/

1. **Design, develop and implement a program for the following operations on Graph (G) of Cities (a) Create a Graph of N cities using Adjacency Matrix. (b) Print all the nodes reachable from a given starting node in a digraph using BFS method.**

#include<iostream.h>

#include<conio.h>

const int n=10;

class graph{

int v, a[n][n], visited[n\*n];

int q[n\*n],front,rear;

public:

graph();

void bfs(int);

};

graph::graph()

{

cout<<"Enter N: ";

cin>>v;

cout<<"Enter adjacency matrix\n";

for (int i=1;i<=v;i++)

{

for (int j=1;j<=v;j++)

cin>>a[i][j];

}

for (i=1;i<=v;i++)

visited[i]=0;

}

void graph::bfs(int s)

{

int i,t;

visited[s] =1;

front= rear = -1;

q[++rear] =s;

while(front!=rear)

{

t= q[++front];

for(i=1;i<=v;i++)

{

if((a[t][i]==1)&&(visited[i]==0))

{

q[++rear] =i;

visited[i] = 1;

}

}

}

if(rear)

{

for(i=1;i<=rear;i++)

cout<<q[i]<<" ";

}

else

cout<<"None";

}

void main()

{

clrscr();

int v,x;

graph g;

cout<<"Enter start node: ";

cin>>x;

cout<<"Nodes reachable from node-"<<x<<": ";

g.bfs(x);

getch();

}

/\*

**Output:**

Enter N: 4

Enter adjacency matrix

0 1 1 0

1 0 0 0

1 0 0 0

0 0 0 0

Enter start node: 1

Nodes reachable from node-1: 2 3

Enter N: 4

Enter adjacency matrix

0 1 1 0

1 0 0 0

1 0 0 0

0 0 0 0

Enter start node: 4

Nodes reachable from node-4: None

\*/