Part –B

5. Write a C++ program to convert an infix expression to postfix.

```
#include<iostream.h>
#include<ctype.h>
char st[100];
int top=-1;
void push(char x)
{
st[++top]=x;
}
char pop()
{
if(top==-1)
return '\0';
else
return st[top--];
}
int priority(char x)
{
if(x=='(')
return 0;
if(x=='+'||x=='-')
return 1;
if(x=='*'||x=='/')
return 2;
return -1;
}
int main()
{
char exp[100];
cout<<"Enter the infix expression:";
cin>>exp;
cout<<"\nConverted postfix expression:";
for(int i=0;exp[i]!='\0';i++)
{
char c=exp[i];
if(isalnum(c))
{
cout<,c;
}
else if(c=='(')
{
push(c);
}
else if(c==')')
{
while(top!=-1&&st[top]!='(')
{
cout<<pop();
}
pop();
}
else
```

```
{
while(top!=-1&&priority(st[top])>=priority(c))
{
cout<<pop();
}
push(c);
}
}
while(top!=-1)
{
cout<<pop();
}
cout<<"\n";
return 0;
}
```

Output:-

Enter the infix expression: A+B*C

Converted postfix expression: ABC*+

6. Write a C++ program to implement a simple queue.

```
#include<iostream.h>

#define MAX 10

int main()

{

int x,s[MAX],i,ch,item,front=0,rear=0;

do

{

cout<<”1.Insert at rear\n”;

cout<<”2.Delete at front\n”;

cout<<”3.Display\n”;

cout<<”4.Exit\n”;

cout<<”Enter your choice:”;

cin>>ch;

switch(ch)
```

```
{
Case 1:
if(rear==MAX)
{
cout<<”Queue is full”<<endl;
break;
}
cout<<”Enter the insert item:”;
cin>>item;
if((rear==0)&&(front==0))
{
front=1;
}
 rear++;
s[rear]=item;
break;
case 2:
if(front==0)
{
cout<<”Queue is empty”<<endl;
break;
}
X=s[front];
if(front==rear)
{
rear=0;
front=0;
```

```
}
else
{
front++;
}
cout<<”Deleted item is:”<<x<<endl;
break;
case 3:
if(front==0)
{
cout<<”Empty”<<endl;
break;
for(i=front;i<=rear;i++)
{
cout<<”s[“<<i<<”]=”<<s[i]<<” “;
}
cout<<endl;
break;
case 4:
return 0;
default:
cout<<”Invalid choice. Please try again.”<<endl;
}
}while(ch!=4);
return 0;
}
```

Output:-

1.Insert at rear

2.Delete at front

3.Display

4.Exit

Enter your choice:1

Enter the insert item:10

1.Insert at rear

2.Delete at front

3.Display

4.Exit

Enter your choice:1

Enter the insert item:20

1.Insert at rear

2.Delete at front

3.Display

4.Exit

Enter your choice:1

Enter the insert item:30

1.Insert at rear

2.Delete at front

3.Display

4.Exit

Enter your choice:1

Enter the insert item:40

1.Insert at rear

2.Delete at front

3.Display

4.Exit

Enter your choice:3

s[1]=10 s[2]=20 s[3]=30 s[4]=40

1.Insert at rear

2.Delete at front

3.Display

4.Exit

Enter your choice:2

Deleted item is=10

1.Insert at rear

2.Delete at front

3.Display

4.Exit

Enter your choice:3

s[2]=20 s[3]=30 s[4]=40

1.Insert at rear

2.Delete at front

3.Display

4.Exit

Enter your choice:4

7. Write a C++ program to implement linear linked list.

```
#include<iostream.h>
struct node
{
int data;
node *next;
};
node *head=NULL;
void createList(int n);
void traverseList();
int main()
```

```
{
int n;
cout<<"Enter the total number of nodes:";
cin>>n;
createList(n);
cout<<"\nData in the list:\n";
traverseList();
return 0;
}
void createList(int n)
{
node *newNode,*temp;
int data;
if(n<=0)
{
cout<<"Invalid number of nodes.";
return;
}
head=new node;
cout<<"Enter the data of node1:";
cin>>data;
head->data=data;
head->next=NULL;
temp=head;
for(int i=2;i<=n;i++)
{
newNode=new node;
cout<<"Enter the data of node "<<i<< ": ";
cin>>data;
newNode->data=data;
newNode->next=NULL;
temp->next=newNode;
temp=newNode;
}
}
void traverseList()
{
node *temp=head;
if(head==NULL)
{
cout<<"List is empty.";
return;
}
while(temp!=NULL)
{
cout<<"Data= "<<temp->data<<endl;
temp=temp->next;
}
}
```

Output:-

Enter the total number of nodes: 5

Enter the data of node 1: 10

Enter the data of node 2: 20

Enter the data of node 3: 30

Enter the data of node 4: 40

Enter the data of node 5: 50

Data in the list:

Data= 10

Data= 20

Data= 30

Data= 40

Data= 50

8. Write a C++ program to display traversal of tree.

```
#include<iostream.h>
struct Node
{
int data;
Node* left;
Node* right;
};
Node* newNode(int value)
{
Node* node=new Node();
node->data=value;
node->left=NULL;
node->right=NULL;
return node;
}
void inorder(Node* root)
{
if(root!=NULL)
{
inorder(root->left);
cout<<root->data<<" ";
inorder(root->right);
}
}
void preorder(Node* root)
{
if(root!=NULL)
{
cout<<root->data<<" ";
```

```
preorder(root->left);
preorder(root->right);
}
}
void postorder(Node* root)
{
if(root!=NULL)
{
postorder(root->left);
postorder(root->right);
cout<<root->data<<" ";
}
}
int main()
{
Node* root=newNode(1);
root->left=newNode(2);
root->right=newNode(3);
root->left->left=newNode(4);
root->left->right=newNode(5);
cout<<"Inorder traversal:";
inorder(root);
cout<<"\nPreorder traversal:";
preorder(root);
cout<<"\nPostorder traversal:";
postorder(root);
return 0;
}
```

Output:-

Inorder traversal: 4 2 5 1 3

Preorder traversal: 1 2 4 5 3

Postorder traversal: 4 5 2 3 1

9. Write a c++ program to search for an element in a binary search tree.

```
#include<iostream.h>

struct Node{

int data;

Node*left;

Node*right;

};

Node*createNode(int data)
```

```
{
Node*newNode=new Node();
if(!newNode){
cout<<"Memory error\n";
return NULL;
}
newNode->data=data;
newNode->left=newNode->right=NULL;
return newNode;
}
Node*insertNode(Node*root,int data)
{
if(root==NULL)
{
root=createNode(data);
return root;
}
if(data<root->data)
root->left=insertNode(root->left,data);
else if(data>root->data)
root->right=insertNode(root->right,data);
return root;
}
int searchNode(Node*root,int key)
{
if(root==NULL)
return 0;
```

```
if(root->data==key)
return 1;
if(key<root->data)
return searchNode(root->left,key);
else
return searchNode(root->right,key);
}
void inorderTraversal(Node*root)
{
if(root){
inorderTraversal(root->left);
cout<<root->data<<" ";
inorderTraversal(root->right);
}
}
int main()
{
Node*root=NULL;
root=insertNode(root,50);
root=insertNode(root,30);
root=insertNode(root,20);
root=insertNode(root,40);
root=insertNode(root,70);
root=insertNode(root,60);
root=insertNode(root,80);
cout<<"Inorder Traversal:";
inorderTraversal(root);
```

```
cout<<endl;

int key;

cout<<"Enter the element to search:";

cin>>key;

if(searchNode(root,key))

cout<<"Element "<<key<<" found in the binary search tree.\n";

else

cout<<"Element "<<key<<" not found in the binary search tree.\n";

return 0;

}
```

Output:-

```
Inorder Traversal: 20 30 40 50 60 70 80

Enter the element to search: 40

Element 40 found in the binary search tree.

Inorder Traversal: 20 30 40 50 60 70 80

Enter the element to search: 10

Element 10 not found in the binary search tree.
```