DS LAB MANUAL

****

Course Code	Course Name	Course Category	L-T-P	Credits
20CS1282	Data Structures	PCC	0-0-3	1.5
	Lab			

Course Objectives:

1. To develop skills to design and analyze simple linear and non-linear data structures
2. To strengthen the ability to identify and apply the suitable data structures for the given real-world problem
3. To gain knowledge in practical applications of data structures.

List of Experiments:

1.Write a C program that uses functions to perform the following:

a.Create a singly linked list of integers.

b.Delete a given integer from the above linked list.

c.Display the contents of the above list after deletion.

2. Write a C program that uses functions to perform the following:

a.Create a doubly linked list of integers.

b.Delete a given integer from the above doubly linked list.

c.Display the contents of the above list after deletion.

3.Write a C program implement the Stack ADT using Arrays and Linked List.

4.Write a C program that uses stack operations to convert a given infix expression into its postfix equivalent.

5.Write a C program that evaluates a postfix expression.

6. Write C program to implement queue ADT using array and doubly linked list.

7. a) Write C program to implement priority queue ADT using array.
b) Write C program to implement circular queue ADT using array.

8. Write C program for implementing the following sorting methods:
a) Insertion sort
b) Merge sort

9. Write C program for implementing the following sorting methods:
b) Quick sort
b) Selection sort
10.Write a C program for implementing Heap sort algorithm.
11.Write a C program that uses functions to perform the following:
a.Create a Binary Search Tree (BST).
b.Insert data in BST
b) Traverse the above BST recursively in Postorder.
12. Write a C program that uses functions to perform the following: a.Deletion an element BST
b.Traverse the above BST non recursively in Inorder.
13.Write a C program to implement all the functions of a dictionary (ADT) using hashing.
14.Write C program for implementing Depth first traversal and Breadth first traversal.

Course Outcomes:

At the end of this lab session, the student will

CO1	**Be able to design and analyze the time and space efficiency of the data structure**
CO2	**Be capable to identity the appropriate data structure for given problem**
CO3	**Have practical knowledge on the application of data structures**

Assessment Method

Assessment Tool	**EXPERIMENTS**	**Report/Viva-Voce/ Quiz/MCQ/Lab project**	**TOTAL**
Weightage (%)	25%	15%	40%

COURSE NATURE	**PRACTRICAL**

Assessment Method

Assessment Tool	**EXPERIMENTS**	**RECORD**	**Report/Viva-Voce/ Quiz/MCQ/Lab project**	**TOTAL**
Weightage (%)	25%	5%	10%	40%
End Semester Examination weightage (%)				60%

1. Write a C program that uses functions to perform the following:
a Create a singly linked list of integers.
b Delete a given integer from the above linked list.
c Display the contents of the above list after deletion.
PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node *newnode,*temp,*prevnode,*nextnode,*head=0;
void creation()
{
int choice;
while(choice)
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter the data:");
scanf("%d",&newnode->data);
newnode->next=0;
if (head==0)
temp=head=newnode;
else
{
temp->next=newnode;
temp=newnode;
}
printf("do you want to continue:(0,1)");
scanf("%d",&choice);
}
}
void display()
{
printf("elements of linked list are:\n");
temp=head;while(temp!=0)
{
printf("%d\n",temp->data);
temp=temp->next;
```

```
}
}
void count()
{
int count=0;
temp=head;
while(temp!=0)
{
count++;
temp=temp->next;
}
printf("no. of nodes:%d\n",count);
}
void insert_at_beg()
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter the data:");
scanf("%d",&newnode->data);
newnode->next=head;
head=newnode;
}
void insert_at_end()
{
temp=head;
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter the data:");
scanf("%d",&newnode->data);
while(temp->next!=0)
temp=temp->next;
temp->next=newnode;
}
void insert_at_pos()
{
int pos,i=1;
printf("enter the position at which the node is to be
inserted:");scanf("%d",&pos);
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter the data:");
scanf("%d",&newnode->data);
temp=head;
```

```
while(i<pos-1)
{
temp=temp->next;
i++;
}
newnode->next=temp->next;
temp->next=newnode;
}
void del_at_beg()
{
temp=head;
temp=temp->next;
free(head);
head=temp;
}
void del_at_end()
{
temp=head;
while(temp->next!=0)
{
prevnode=temp;
temp=temp->next;
}
prevnode->next=0;
free(temp);
}
void del_at_pos()
{
int pos,i=1;
printf("enter the position of node which is to be deleted:");
scanf("%d",&pos);
temp=head;
while(i<pos-1)
{temp=temp->next;
i++;
}
nextnode=temp->next;
temp->next=nextnode->next;
free(nextnode);
}
```

```
void del_num()
{
int num;
printf("enter the number:");
scanf("%d",&num);
temp=head;
while(temp->data!=num)
{
prevnode=temp;
temp=temp->next;
}
prevnode->next=temp->next;
free(temp);
}
void main()
{
int option;
printf("****MAIN MENU****\n");
printf("1.create the linked list\n");
printf("2.display the linked list\n");
printf("3.count the no. of nodes\n");
printf("4.insert a node at beginning\n");
printf("5.insert a node at ending\n");
printf("6.insering a node at given position\n");
printf("7.deleting a node at beginning\n");
printf("8.deleting a node at ending\n");
printf("9.deleting a node at given position\n");
printf("10.deleting given integer:\n");
do
{
printf("enter your option:");
scanf("%d",&option);switch(option)
{
case 1:creation();
break;
case 2:display();
break;
case 3:count();
break;
case 4:insert_at_beg();
```

```
break;
case 5:insert_at_end();
break;
case 6:insert_at_pos();
break;
case 7:del_at_beg();
break;
case 8:del_at_end();
break;
case 9:del_at_pos();
break;
case 10:del_num();
break;
default:
printf("invalid option");
}
}while(option!=11);
}
```

OUTPUT:

```
****MAIN MENU****
1.create the linked list
2.display the linked list
3.count the no. of nodes
4.insert a node at beginning
5.insert a node at ending
6.insering a node at given position
7.deleting a node at beginning
8.deleting a node at ending
9.deleting a node at given position
10.deleting given integer:enter your option:1
enter the data:10
do you want to continue:(0,1)1
enter the data:20
do you want to continue:(0,1)1
enter the data:30
do you want to continue:(0,1)1
enter the data:40
do you want to continue:(0,1)1
enter the data:50
do you want to continue:(0,1)0
```

```
enter your option:2
elements of linked list are:
10
20
30
40
50
enter your option:3
no. of nodes:5
enter your option:4
enter the data:5
enter your option:2
elements of linked list are:
5
10
20
30
40
50
enter your option:5
enter the data:60
enter your option:2
elements of linked list are:
5
10
20
30
4050
60
enter your option:6
enter the position at which the node is to be inserted:3
enter the data:15
enter your option:2
elements of linked list are:
5
10
15
20
30
40
```

```
50
60
enter your option:7
enter your option:2
elements of linked list are:
10
15
20
30
40
50
60
enter your option:8
enter your option:2
elements of linked list are:
10
15
20
30
40
50
enter your option:9
enter the position of node which is to be deleted:2
enter your option:2
elements of linked list are:
1020
30
40
50
enter your option:10
enter the number:40
enter your option:2
elements of linked list are:
10
20
30
50
enter your option:11
invalid option
```

2. Write a C program that uses functions to perform the following:
a Create a doubly linked list of integers.
b Delete a given integer from the above doubly linked list. c

Display the contents of the above list after deletion.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
struct node *prev;
};
struct node *head=0,*tail,*newnode,*temp,*nextnode;
void creation()
{
int choice;
while(choice)
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter the data:");scanf("%d",&newnode->data);
newnode->next=0;
newnode->prev=0;
if (head==0)
head=tail=newnode;
else
{
```

```
tail->next=newnode;
newnode->prev=tail;
tail=newnode;
}
printf("do you want to continue:(0,1)");
scanf("%d",&choice);
}
}
void display()
{
printf("the elements of list are:\n");
temp=head;
while(temp!=0)
{
printf("%d\n",temp->data);
temp=temp->next;
}
}
void count()
{
int count=0;
temp=head;
while(temp!=0)
{
count++;
temp=temp->next;
}
```

```
printf("the no. of elements are:%d\n",count);
}
void insert_at_beg()
{
newnode=(struct node*)malloc(sizeof(struct node));printf("enter
the data:");
scanf("%d",&newnode->data);
newnode->prev=0;
head->prev=newnode;
newnode->next=head;
head=newnode;
}
void insert_at_end()
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter the data:");
scanf("%d",&newnode->data);
newnode->next=0;
newnode->prev=tail;
tail->next=newnode;
tail=newnode;
}
void insert_at_pos()
{
int i=1,pos;
printf("enter the position at which the node is to be inserted:");
scanf("%d",&pos);
```

```
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter the data:");
scanf("%d",&newnode->data);
temp=head;
while(i<pos-1)
{
temp=temp->next;
i++;
}
newnode->prev=temp;
newnode->next=temp->next;
temp->next=newnode;
newnode->next->prev=newnode;
}
void del_at_beg()
{
temp=head;head=head->next;
head->prev=0;
free(temp);
}
void del_at_end()
{
temp=tail;
tail=tail->prev;
tail->next=0;
free(temp);
}
```

```
void del_at_pos()
{
int pos,i=1;
printf("enter the position at which the node is to be deleted:");
scanf("%d",&pos);
temp=head;
while(i<pos-1)
{
temp=temp->next;
i++;
}
nextnode=temp->next;
temp->next=nextnode->next;
nextnode->next->prev=temp;
free(nextnode);
}
void del_num()
{
int num;
printf("enter the number:");
scanf("%d",&num);
temp=head;
while(temp->data!=num)
{
temp=temp->next;
}
temp->prev->next=temp->next;
```

```
temp->next->prev=temp->prev;free(temp);
}
void main()
{
int option;
printf("***MAIN MENU***\n");
printf("1.create the list:\n");
printf("2.dispaly the list:\n");
printf("3.count the no. of node:\n");
printf("4.insert the node at beginning:\n");
printf("5.insert the node at ending:\n");
printf("6.insert the node at given position:\n");
printf("7.delete the node at beginnig:\n");
printf("8.delete the node at end:\n");
printf("9.delete the node at given position:\n");
printf("10.delete the given number:\n");
do
{
printf("enter your option:");
scanf("%d",&option);
switch(option)
{
case 1:creation();
break;
case 2:display();
break;
case 3:count();
```

```
break;
case 4:insert_at_beg();
break;
case 5:insert_at_end();
break;
case 6:insert_at_pos();
break;
case 7:del_at_beg();
break;
case 8:del_at_end();
break;
case 9:del_at_pos();break;
case 10:del_num();
break;
default:printf("invalid option\n");
}
}while(option<=11);
}
```

OUTPUT:

MAIN MENU

1.create the list:

2.dispaly the list:

3.count the no. of node:

4.insert the node at beginning:

5.insert the node at ending:

6.insert the node at given position:

7.delete the node at beginnig:

8.delete the node at end:

9.delete the node at given position:

10.delete the given number:

enter your option:1

enter the data:23

do you want to continue:(0,1)1

enter the data:89

do you want to continue:(0,1)1

enter the data:45

do you want to continue:(0,1)1

enter the data:83

do you want to continue:(0,1)1

enter the data:3

do you want to continue:(0,1)0

enter your option:2

the elements of list are:

23

89

45

83

3enter your option:3

the no. of elements are:5

enter your option:4

enter the data:11

enter your option:2

the elements of list are:

11

23
89
45
83
3
enter your option:5
enter the data:90
enter your option:2
the elements of list are:
11
23
89
45
83
3
90
enter your option:6
enter the position at which the node is to be inserted:4
enter the data:78
enter your option:2
the elements of list are:
11
23
89
78
45
83

```
3
90
enter your option:7
enter your option:2
the elements of list are:23
89
78
45
83
3
90
enter your option:8
enter your option:2
the elements of list are:
23
89
78
45
83
3
enter your option:9
enter the position at which the node is to be deleted:3
enter your option:2
the elements of list are:
23
89
45
```

83

3

enter your option:10

enter the number:83

enter your option:2

the elements of list are:

23

89

45

3

enter your option:12

invalid option

3. Write a C program that uses functions to perform the following:
a Create a circular linked list of integers.
b Delete a given integer from the above linked list.
c Display the contents of the above list after deletion.
PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node *newnode,*temp,*ptr,*nextnode,*p,*node,*head=0;
void creation()
{
int choice;
while(choice)
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter the data:");
scanf("%d",&newnode->data);
newnode->next=0;
if (head==0)
{
temp=head=newnode;
newnode->=head;
}
else
{
temp->next=newnode;
temp=newnode;
newnode->=head;
}
printf("do you want to continue:(0,1)");
scanf("%d",&choice);
}
}
void display()
{printf("elements of linked list are:\n");
temp=head;
```

```
while(temp->next!=head)
{
printf("%d\n",temp->data);
temp=temp->next;
}
}
void count()
{
int count=1;
temp=head;
while(temp->next!=head)
{
count++;
temp=temp->next;
}
printf("no. of nodes:%d\n",count);
}
void insert_at_beg()
{
newnode=(struct node*)malloc(sizeof(struct node*));
printf("enter data:");
scanf("%d",&newnode->data);
temp=head;
while(temp->next!=head)
{
temp=temp->next;
}
newnode->next=head;
temp->next=newnode;
head=newnode;
}
void insert_at_end()
{
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter data:");
scanf("%d",&newnode->data);newnode->next=NULL;
temp=head;
while(temp->next!=head)
{
temp=temp->next;
```

```
}
temp->next=newnode;
newnode->next=head;
}
void insert_at_pos()
{
int n=1,k=1,pos;
temp=head;
while(temp->next!=head)
{
temp=temp->next;
n++;
}
printf("enter the position at which the node is to be inserted:");
scanf("%d",&pos);
if(pos>n)
{
printf("invalid position.the length of the list is %d.",n);
}
else if(pos==1)
{insert_beg();}
else if(pos==n)
{insert_end();}
else
{
temp=head;
while (k<pos-1)
{
temp=temp->next;
k++;
}
newnode=(struct node*)malloc(sizeof(struct node*));
printf("enter data:");
scanf("%d",&newnode->data);newnode->next=temp->next;
temp->next=newnode;
}
}
void del_at_beg()
{
temp=head;
```

```
node=head;
while(temp->next!=head)
{
temp=temp->next;
}
head=head->next;
node->next=NULL;
node->data=NULL;
temp->next=head;
free(node);
}
void del_at_end()
{
int l=1;
p=head;
while (l<q-1)
{
p=p->next;
l++;
}
node=p->next;
p->next=head;
node->next=NULL;
node->data=NULL;
free(node);
}
void del_at_pos()
{
temp=head;
while (l<q-1){
temp=temp->next;
l++;
}
node=temp->next;
temp->next=temp->next->next;
node->next=NULL;
node->data=NULL;
free(node);
}
void del_num()
```

```
{
int num;
printf("enter the number:");
scanf("%d",&num);
temp=head;
while(temp->data!=num)
{
ptr=temp;
temp=temp->next;
}
ptr->next=temp->next;
free(temp);
}
void main()
{
int option;
printf("****MAIN MENU****\n");
printf("1.create the linked list\n");
printf("2.display the linked list\n");
printf("3.count the no. of nodes\n");
printf("4.insert a node at beginning\n");
printf("5.insert a node at ending\n");
printf("6.insering a node at given position\n");
printf("7.deleting a node at beginning\n");
printf("8.deleting a node at ending\n");
printf("9.deleting a node at given position\n");
printf("10.deleting given integer:\n");do
{
printf("enter your option:");
scanf("%d",&option);
switch(option)
{
case 1:creation();
break;
case 2:display();
break;
case 3:count();
break;
case 4:insert_at_beg();
break;
```

```
case 5:insert_at_end();
break;
case 6:insert_at_pos();
break;
case 7:del_at_beg();
break;
case 8:del_at_end();
break;
case 9:del_at_pos();
break;
case 10:del_num();
break;
default:
printf("invalid option");
}
}while(option!=11);
}
```

OUTPUT:

```
****MAIN MENU****
1.create the linked list
2.display the linked list
3.count the no. of nodes
4.insert a node at beginning5.insert a node at ending
6.insering a node at given position
7.deleting a node at beginning
8.deleting a node at ending
9.deleting a node at given position
10.deleting given integer:
enter your option:1
enter the data:10
do you want to continue:(0,1)1
enter the data:20
do you want to continue:(0,1)1
enter the data:30
do you want to continue:(0,1)1
enter the data:40
do you want to continue:(0,1)1
enter the data:50
do you want to continue:(0,1)0
enter your option:2
```

```
elements of linked list are:
10
20
30
40
50
enter your option:3
no. of nodes:5
enter your option:4
enter the data:5
enter your option:2
elements of linked list are:
5
10
20
30
40
50
enter your option:5
enter the data:60
enter your option:2elements of linked list are:
5
10
20
30
40
50
60
enter your option:6
enter the position at which the node is to be inserted:3
enter the data:15
enter your option:2
elements of linked list are:
5
10
15
20
30
40
50
```

```
60
enter your option:7
enter your option:2
elements of linked list are:
10
15
20
30
40
50
60
enter your option:8
enter your option:2
elements of linked list are:
10
15
20
30
4050
enter your option:9
enter the position of node which is to be deleted:2
enter your option:2
elements of linked list are:
10
20
30
40
50
enter your option:10
enter the number:40
enter your option:2
elements of linked list are:
10
20
30
50
enter your option:11
invalid option
```

4. Write a C program implement the Stack ADT using Arrays and Linked List.

PROGRAM:

```
//STACK USING ARRAYS
#include<stdio.h>
#define N 5
int stack[N];
int top=-1;
void push()
{
int n;
printf("enter the element:");
scanf("%d",&n);
if(top==N-1)
printf("overflow\n");
else{
top++;
stack[top]=n;
}
}
void pop()
{
if (top==-1)
printf("underflow\n");
else
{
printf("deleting element=%d\n",stack[top]);
top--;
}
}
void peek()
{
if(top==-1)
printf("underflow\n");
else
printf("%d\n",stack[top]);
}
```

```
void display()
{
if (top==-1)
printf("underflow\n");
else
{
printf("the elements of stack are:\n");
for (int i=top;i>=0;i--)
printf("%d\n",stack[i]);
}
}
void main()
{
int option;
printf("1.push() operation\n");
printf("2.pop() operation\n");
printf("3.peek() operation\n");printf("4.display() operation\n");
do
{
printf("enter your option:");
scanf("%d",&option);
switch(option)
{
case 1:push();
break;
case 2:pop();
break;
case 3:peek();
break;
case 4:display();
break;
default:printf("invalid option\n");
}
}while(option<=4);
}
```

OUTPUT:

1.push() operation

2.pop() operation
3.peek() operation
4.display() operation
enter your option:1
enter the element:10
enter your option:1
enter the element:20
enter your option:1
enter the element:30
enter your option:1
enter the element:40
enter your option:1
enter the element:50
enter your option:1
enter the element:60
overflowenter your option:3
50
enter your option:4
the elements of stack are:
50
40
30
20
10
enter your option:2
deleting element=50
enter your option:4
the elements of stack are:
40
30
20
10
enter your option:5
invalid option
PROGRAM:
//STACK USING LINKED LIST
#include<stdio.h>

```
#include<stdlib.h>
struct node
{
int data;
struct node *link;
};
struct node *newnode,*top=0,*temp;
void push()
{
int n;
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter the data:");
scanf("%d",&n);
newnode->data=n;
newnode->link=top;top=newnode;
}
void pop()
{
if (top==0)
printf("underflow\n");
else
{
temp=top;
printf("deleting element=%d\n",temp->data);
top=top->link;
free(temp);
}
}
void peek()
{
if (top==0)
printf("underflow\n");
else
printf("%d\n",top->data);
}
void display()
{
```

```
if (top==0)
printf("underflow\n");
else
{
printf("the elements of stack are:\n");
temp=top;
while(temp!=0)
{
printf("%d\n",temp->data);
temp=temp->link;
}
}
}
void main()
{
int option;printf("1.push() operation\n");
printf("2.pop() operation\n");
printf("3.peek() operation\n");
printf("4.display() operation\n");
do
{
printf("enter your option:");
scanf("%d",&option);
switch(option)
{
case 1:push();
break;
case 2:pop();
break;
case 3:peek();
break;
case 4:display();
break;
default:printf("invalid option\n");
}
}while(option<=4);
}
```

OUTPUT:

```
1.push() operation
2.pop() operation
3.peek() operation
4.display() operation
enter your option:1
enter the data:10
enter your option:1
enter the data:20
enter your option:1
enter the data:30
enter your option:1
enter the data:40
enter your option:1
enter the data:50
enter your option:4
the elements of stack are:50
40
30
20
10
enter your option:3
50
enter your option:2
deleting element=50
enter your option:4
the elements of stack are:
40
30
20
10
enter your option:5
invalid option
```

5. Write a C program that uses stack operations to convert a given infix expression into its postfix equivalent.

PROGRAM:

```
#include<stdio.h>
#include<ctype.h>
char stack[100];
int top=-1;
void push(char x)
{
top++;
stack[top]=x;
}
char pop()
{
if (top==-1)
return -1;else
return stack[top--];
}
int priority(char x)
{
if (x=='(')
return 0;
else if(x=='+' || x=='-')
return 1;
else if(x=='*' || x=='/')
return 2;
return 0;
}
int main()
```

```
{
char exp[100],*e,x;
printf("enter the expression:");
scanf("%s",exp);
e=exp;
while(*e!='\0')
{
if(isalnum(*e))
printf("%c",*e);
else if(*e=='(')
push(*e);
else if(*e==')')
{
while((x=pop())!='(')
printf("%c",x);
}
else
{
while(priority(stack[top])>=priority(*e))
printf("%c",pop());
push(*e);
}
e++;
}
while(top!=-1)
printf("%c",pop());
return 0;
}
```

OUTPUT:

enter the expression:a+b-c*d*(e+f-g)/h

ab+cd*ef+g-*h/

6. Write a C program that evaluates a postfix expression.

PROGRAM:

```
#include<stdio.h>
#include<ctype.h>
int stack[100];
int top=-1;
void push(int x)
{
top++;
stack[top]=x;
}
int pop()
{
return stack[top--];
}
int main()
{
char exp[100],*e,x;
printf("enter the expression:");
scanf("%s",exp);
e=exp;
int n1,n2,n3,num;
while(*e!='\0')
{
if (isdigit(*e))
{
num=*e-48;
push(num);
}else
{
n1=pop();
n2=pop();
switch(*e)
{
case '+':n3=n2+n1;
break;
case '-':n3=n2-n1;
break;
case '*':n3=n2*n1;
break;
case '/':n3=n2/n1;
break;
}
push(n3);
}
```

```
e++;
}
printf("the result of the expression %s is %d\n",exp,pop());
return 0;
}
```

OUTPUT:

```
enter the expression:245+*
the result of the expression 245+* is 18
```

7. Write C program to implement queue ADT using array and doubly linked list

```
//QUEUE USING ARRAYS
#include<stdio.h>
#define N 5
int queue[N];
int front=-1;
int rear=-1;
void enqueue(){
int n;
printf("enter the data:");
scanf("%d",&n);
if (front==-1 && rear==-1)
{
front=rear=0;
queue[rear]=n;
}
else if(rear==N-1)
printf("overflow\n");
else
{
rear++;
queue[rear]=n;
}
}
void dequeue()
{
if(front==-1 && rear==-1)
printf("underflow\n");
else
{
printf("deleting item=%d\n",queue[front]);
front++;
}
}
void peek()
{
if(front==-1 &&rear==-1)
printf("underflow\n");
```

```
else
printf("%d\n",queue[front]);
}
void display()
{
if (front==-1 && rear==-1)
printf("underflow\n");
else{
printf("the elements of queue are:\n");
for (int i=front;i<=rear;i++)
printf("%d\n",queue[i]);
}
}
void main()
{
int option;
printf("1.enqueue() operation\n");
printf("2.dequeue() operation\n");
printf("3.peek() operation\n");
printf("4.display() operation\n");
do
{
printf("enter your option:");
scanf("%d",&option);
switch(option)
{
case 1:enqueue();
break;
case 2:dequeue();
break;
case 3:peek();
break;
case 4:display();
break;
default:printf("invalid option\n");
}
}while(option<=4);
}
```

OUTPUT:

1.enqueue() operation

2.dequeue() operation
3.peek() operation
4.display() operation
enter your option:1
enter the data:5
enter your option:1enter the data:7
enter your option:1
enter the data:2
enter your option:1
enter the data:9
enter your option:1
enter the data:6
enter your option:4
the elements of queue are:
5
7
2
9
6
enter your option:3
5
enter your option:2
deleting item=5
enter your option:4
the elements of queue are:
7
2
9
6
enter your option:2
deleting item=7
enter your option:4
the elements of queue are:
2
9
6
enter your option:5
invalid option
PROGRAM:
//QUEUE USING DOUBLE LINKED LIST

```
#include<stdio.h>
#include<stdlib.h>struct node
{
int data;
struct node *next;
struct node *prev;
};
struct node *newnode,*temp,*front=0,*rear=0;
void enqueue()
{
int n;
newnode=(struct node*)malloc(sizeof(struct node));
printf("enter the value:");
scanf("%d",&n);
newnode->data=n;
newnode->next=0;
newnode->prev=0;
if(front==0 && rear==0)
{
front=newnode;
rear=newnode;
}
else
{
newnode->next=0;
newnode->prev=rear;
rear->next=newnode;
rear=newnode;
}
}
void dequeue()
{
if (front==0 && rear==0)
printf("underflow\n");
else if(front->next==0)
{
printf("deleting element=%d\n",front->data);
front=rear=0;
free(front);
}else
```

```
{
temp=front;
printf("deleting element=%d\n",temp->data);
front=front->next;
free(temp);
}
}
void peek()
{
if (front==0 && rear==0)
printf("underflow\n");
else
printf("top most element=%d\n",front->data);
}
void display()
{
if (front==0 && rear==0)
printf("underflow\n");
else
{
printf("the elements of queue are:\n");
temp=front;
while(temp!=0)
{
printf("%d\n",temp->data);
temp=temp->next;
}
}
}
void main()
{
int option;
printf("1.enqueue() operation\n");
printf("2.dequeue() operation\n");
printf("3.peek() operation\n");
printf("4.display() operation\n");
do
{printf("enter your option:");
scanf("%d",&option);
switch(option)
```

```
{
case 1:enqueue();
break;
case 2:dequeue();
break;
case 3:peek();
break;
case 4:display();
break;
default:printf("invalid option\n");
}
}while(option<=4);
}
```

OUTPUT:

```
1.enqueue() operation
2.dequeue() operation
3.peek() operation
4.display() operation
enter your option:1
enter the value:45
enter your option:1
enter the value:90
enter your option:1
enter the value:26
enter your option:1
enter the value:8
enter your option:1
enter the value:82
enter your option:4
the elements of queue are:
45
90
26
8
82
enter your option:3top most element=45
enter your option:2
deleting element=45
enter your option:4
the elements of queue are:
```

90
26
8
82
enter your option:5
invalid option

8. a) Write C program to implement priority queue ADT using array.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int pri_que[MAX];
int front=-1, rear=-1;
void check(int data)
{
int i,j;
for (i = 0; i <= rear; i++)
{
if (data >= pri_que[i])
{
for (j = rear + 1; j > i; j--)
{
pri_que[j] = pri_que[j - 1];
}
pri_que[i] = data;
return;
}
}
pri_que[i] = data;
}void insert_by_priority(int data)
{
if (rear >= MAX - 1)
{
printf("\nQueue overflow no more elements can be inserted");
return;
}
if ((front == -1) && (rear == -1))
{
front++;
rear++;
pri_que[rear] = data;
return;
}
else
check(data);
```

```
rear++;
}
void delete_by_priority(int data)
{
int i;
if ((front==-1) && (rear==-1))
{
printf("\nQueue is empty no elements to delete");
return;
}
for (i = 0; i <= rear; i++)
{
if (data == pri_que[i])
{
for (; i < rear; i++)
{
pri_que[i] = pri_que[i + 1];
}
pri_que[i] = -99;
rear--;
if (rear == -1)
front = -1;
return;}
}
printf("\n%d not found in queue to delete", data);
}
void display_pqueue()
{
if ((front == -1) && (rear == -1))
{
printf("\nQueue is empty");
return;
}
for (; front <= rear; front++)
{
printf(" %d ", pri_que[front]);
}
front = 0;
}
void main()
```

```
{
int n, ch;
printf("1 - Insert an element into queue\n");
printf("2 - Delete an element from queue\n");
printf("3 - Display queue elements\n");
printf("4 - Exit\n");
while (1)
{
printf("\nEnter your choice : ");
scanf("%d", &ch);
switch (ch)
{
case 1:
printf("\nEnter value to be inserted : ");
scanf("%d",&n);
insert_by_priority(n);
break;
case 2:
printf("\nEnter value to delete : ");
scanf("%d",&n);
delete_by_priority(n);break;
case 3:
display_pqueue();
break;
case 4:
exit(0);
default:
printf("\nChoice is incorrect, Enter a correct choice");
}
}
}
```

OUTPUT:

```
1 - Insert an element into queue
2 - Delete an element from queue
3 - Display queue elements
4 - Exit
Enter your choice : 1
Enter value to be inserted : 34
Enter your choice : 1
Enter value to be inserted : 56
```

```
Enter your choice : 1
Enter value to be inserted : 12
Enter your choice : 1
Enter value to be inserted : 90
Enter your choice : 1
Enter value to be inserted : 27
Enter your choice : 3
90 56 34 27 12
Enter your choice : 2
Enter value to delete : 56
Enter your choice : 3
90 34 27 12
Enter your choice : 4
```

b) Write C program to implement circular queue ADT using array.

```
PROGRAM:#include<stdio.h>
#define N 5
int queue[N];
int front=-1;
int rear=-1;
void enqueue()
{
int n;
printf("enter the value:");
scanf("%d",&n);
if (front ==-1 && rear==-1)
{
front=rear=0;
queue[rear]=n;
}
else if((rear+1)%N==front)
printf("queue is full\n");
else
{
rear++;
queue[rear]=n;
}
}
```

```
void dequeue()
{
if (front==-1 && rear==-1)
printf("underflow\n");
else if(front==rear)
front=rear=-1;
else
{
printf("deleting element=%d\n",queue[front]);
front++;
}
}
void peek()
{
if (front==-1 && rear==-1)printf("underflow\n");
else
printf("top most element=%d\n",queue[front]);
}
void display()
{
if (front==-1 && rear==-1)
printf("underflow\n");
else
{
printf("the elements of queue are:\n");
for (int i=front;i<=rear;i++)
printf("%d\n",queue[i]);
}
}
void main()
{
int option;
printf("1.enqueue() operation\n");
printf("2.dequeue() operation\n");
printf("3.peek() operation\n");
printf("4.display() operation\n");
do
{
printf("enter your option:");
scanf("%d",&option);
```

```
switch(option)
{
case 1:enqueue();
break;
case 2:dequeue();
break;
case 3:peek();
break;
case 4:display();
break;
default:printf("invalid option\n");
}
}while(option<=4);}
```

OUTPUT:

```
1.enqueue() operation
2.dequeue() operation
3.peek() operation
4.display() operation
enter your option:1
enter the value:23
enter your option:1
enter the value:94
enter your option:1
enter the value:28
enter your option:1
enter the value:63
enter your option:1
enter the value:56
enter your option:4
the elements of queue are:
23
94
28
63
56
enter your option:3
top most element=23
enter your option:2
deleting element=23
enter your option:4
```

the elements of queue are:
94
28
63
56
enter your option:5
invalid option

9.Write a C program for the following:
a) Linear search b)Binary search

PROGRAM:

```
//Linear search
#include<stdio.h>
int main()
{
int a[20],n,b,i;
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("Enter the element to be searched:");
scanf("%d",&b);
for(i=0;i<n;i++)
{
if(a[i]==b)
{
break;
}
}
if(i<n)
printf("Elements found at index %d",i);
else
printf("Elements not found");
return 0;
}
```

OUTPUT:

```
Enter the number of elements:4
Enter the elements:25
1
6
Enter the element to be searched:2
Elements found at index 0
Enter the number of elements:4
Enter the elements:2
```

5
1
6
Enter the element to be searched:4
Elements not found

//binary search

```
//binary search
#include <stdio.h>
void main()
{
int array[10];
int i, j, num, temp, keynum;
int low, mid, high;
printf("Enter number of elments in the array: \n");
scanf("%d", &num);
printf("Enter the elements one by one: \n");
for (i = 0; i < num; i++)
{
scanf("%d", &array[i]);
}
for (i = 0; i < num; i++)
{
for (j = 0; j < (num - i - 1); j++)
{
if (array[j] > array[j + 1])
{temp = array[j];
array[j] = array[j + 1];
array[j + 1] = temp;
}
}
}
printf("Enter the element to be searched \n");
scanf("%d", &keynum);
/* Binary searching begins */
low = 1;
high = num;
do
{
```

```
mid = (low + high) / 2;
if (keynum < array[mid])
high = mid - 1;
else if (keynum > array[mid])
low = mid + 1;
} while (keynum != array[mid] && low <= high);
if (keynum == array[mid])
{
printf("SEARCH SUCCESSFUL \n");
}
else
{
printf("SEARCH FAILED \n");
}
}
```

OUTPUT:

```
Enter number of elments in the array:
5
Enter the elements one by one:
2
7
3
9
6Enter the element to be searched
3
SEARCH SUCCESSFUL
```

10. Write C program for implementing the following sorting methods:
a **Insertion sort** b

Merge sort

PROGRAM:

```
//INSERTION SORT
#include<stdio.h>
void main()
{
int arr[50],n,temp;
printf("enter the no. of elements:");
scanf("%d",&n);
printf("enter the elements of array:");
for (int i=0;i<n;i++)
scanf("%d",&arr[i]);
for(int i=1;i<n;i++)
{
for (int j=i;j>0;j--)
{
if (arr[j-1]>arr[j])
{
temp=arr[j-1];
arr[j-1]=arr[j];
arr[j]=temp;
}
}
}
printf("the elements of sorted array:");
for (int i=0;i<n;i++)
printf("%d\t",arr[i]);
printf("\n");
}OUTPUT:
enter the no. of elements:10
enter the elements of array:
67
12
90
32
74
3
```

82
65
49
23
the elements of sorted array:3 12 23 32 49 65 67 74 82 90

PROGRAM:

```
//MERGE SORT
#include<stdio.h>
void merge(int arr[50],int lower,int mid,int upper);
void mergesort(int arr[50],int lower,int upper)
{
int mid;
if (lower<upper)
{
mid=(lower+upper)/2;
mergesort(arr,lower,mid);
mergesort(arr,mid+1,upper);
merge(arr,lower,mid,upper);
}
}
void merge(int arr[50],int lower,int mid,int upper)
{
int b[50];
int i=lower;
int j=mid+1;
int k=lower;
while(i<=mid && j<=upper){
if(arr[i]<=arr[j])
{
b[k]=arr[i];
i++;
}
else
{
b[k]=arr[j];
j++;
}
k++;
}
```

```
if (i>mid)
{
while(j<=upper)
{
b[k]=arr[j];
j++;
k++;
}
}
else
{
while(i<=mid)
{
b[k]=arr[i];
i++;
k++;
}
}
for (k=lower;k<=upper;k++)
arr[k]=b[k];
}
void main()
{
int arr[50],n;
printf("enter the numbers:");
scanf("%d",&n);printf("enter the elements of array:");
for (int i=0;i<n;i++)
scanf("%d",&arr[i]);
mergesort(arr,0,n-1);
printf("the elements of sorted array:");
for (int i=0;i<n;i++)
printf("%d\t",arr[i]);
printf("\n");
}
```

OUTPUT:

```
enter the numbers:10
enter the elements of array:
89
12
3
```

73
9
19
89
94
65
39
the elements of sorted array:3 9 12 19 39 65 73 89 89 94

11. Write C program for implementing the following sorting methods:

a **Quick sort**

b **Selection sort**

PROGRAM:

PROGRAM:

```
//QUICK SORT
#include<stdio.h>
void quicksort(int arr[50],int lower,int upper)
{
int temp,pivot,start,end;
if(lower<upper)
{
pivot=lower;
start=lower;end=upper;
while(start<end)
{
while(arr[start]<=arr[pivot])
start++;
while(arr[end]>arr[pivot])
end--;
if (start<end)
{
temp=arr[start];
arr[start]=arr[end];
arr[end]=temp;
}
}
temp=arr[end];
arr[end]=arr[lower];
arr[lower]=temp;
quicksort(arr,lower,end-1);
quicksort(arr,end+1,upper);
}
}
void main()
{
int arr[50],n;
printf("enter the no. of elements:");
scanf("%d",&n);
```

```
printf("enter the elements of array:");
for(int i=0;i<n;i++)
scanf("%d",&arr[i]);
quicksort(arr,0,n-1);
printf("the elements of sorted array:");
for (int i=0;i<n;i++)
printf("%d\t",arr[i]);
printf("\n");
}
```

OUTPUT:

```
enter the no. of elements:10
enter the elements of array:
2378
12
6
90
65
73
85
39
1
the elements of sorted array:1 6 12 23 39 65 73 78
85 90
```

PROGRAM:

```
//SELECTION SORT
#include<stdio.h>
void main()
{
int arr[50],n,temp;
printf("enter the no. of elements of array:");
scanf("%d",&n);
printf("enter the elements of array:");
for (int i=0;i<n;i++)
scanf("%d",&arr[i]);
for (int i=0;i<n;i++)
{
for (int j=i+1;j<n;j++)
{
if (arr[i]>arr[j])
{
temp=arr[i];
```

```
arr[i]=arr[j];
arr[j]=temp;
}
}
}
printf("the elements of sorted array:");
for (int i=0;i<n;i++)printf("%d\t",arr[i]);
printf("\n");
}
```

OUTPUT:

```
enter the no. of elements of array:10
enter the elements of array:
3
98
2
67
45
92
12
88
57
56
the elements of sorted array:2 3 12 45 56 57 67 88 92 98
```

12. Write a C program for implementing Heap sort algorithm.

PROGRAM:

```
#include <stdio.h>
#define MAX 50
void RestoreHeapUp(int *,int);
void RestoreHeapDown(int*,int,int);
int main()
{
int Heap[MAX],n,i,j;
printf("Enter the number of elements : ");
scanf("%d",&n);
printf("Enter the elements : ");
for(i=1;i<=n;i++)
{
scanf("%d",&Heap[i]);
RestoreHeapUp(Heap,i);
}
j=n;
for(i=1;i<=j;i++){
int temp;
temp=Heap[1];
Heap[1]= Heap[n];
```

```
Heap[n]=temp;
n = n-1;
RestoreHeapDown(Heap,1,n);
}
n=j;
printf("\n The sorted elements are: ");
for(i=1;i<=n;i++)
printf("%4d",Heap[i]);
return 0;
}
void RestoreHeapUp(int *Heap,int index)
{
int val=Heap[index];
while((index>1)&&(Heap[index/2]<val))
{
Heap[index]=Heap[index/2];
index/=2;
}
Heap[index]=val;
}
void RestoreHeapDown(int *Heap,int index,int n)
{
```

```
int val=Heap[index];
int j=index*2;
while(j<=n)
{
if((j<n)&&(Heap[j]<Heap[j+1]))
j++;
if(Heap[j]<Heap[j/2])
break;
Heap[j/2]=Heap[j];
j=j*2;
}
Heap[j/2]=val;
}OUTPUT:
Enter the number of elements : 5
Enter the elements : 76
23
89
12
8
The sorted elements are: 8 12 23 76 89
```

13.Write a C program for the following:

a **bubble sort** b **shell sort**

// Bubble sort

```
#include<stdio.h>
void main()
{
int arr[20],n,i,j,flag,temp;
printf("Enter the no.of elements of the array:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("the elements of array are:");
scanf("%d",&arr[i]);
}
for(i=0;i<n-1;i++)
{
flag=0;
for(j=0;j<n-1-i;j++)
{
if(arr[j]>arr[j+1])
{
temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
flag=1;
}
}if(flag==0)
break;
}
printf("the elements of sorted array is:");
for(i=0;i<n;i++)
{
printf("%d\t",arr[i]);
}
printf("\n");
}
```

OUTPUT:

```
Enter the no.of elements of the array:4
the elements of array are:2
the elements of array are:6
the elements of array are:1
the elements of array are:9
```

the elements of sorted array is:1 2 6 9
//shell sort
PROGRAM:

```
#include<stdio.h>
void main()
{
int a[50],n,i,j,k,gap,temp;
printf("enter no.of elements:");
scanf("%d",&n);
printf("the elements of array are:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(gap=n/2;gap>=1;gap=gap/2)
{
for(j=gap;j<n;j++)
{
temp=a[j];
for(k=j-gap;a[k]>=temp && k>=0; k=k-gap)
a[k+gap]=a[k];
a[k+gap]=temp;
}
}
printf("the elements of sorted array:");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
printf("\n");
}
```

OUTPUT:

```
enter no.of elements:4
the elements of array are:3
1
6
8
the elements of sorted array:1
3
6
8
```

14.Write a C program for the following:
a **countable sort** b **radix sort**

//countable sort
PROGRAM:

```
//count sort
#include<stdio.h>
6
8int getMax(int a[],int n)
{
int max=a[0];
for(int i=1;i<n;i++)
{
if(a[i]>max)
{
max=a[i];
}
}
return max;
}
void countsort(int a[],int n)
{ int i;
int b[n+1];
int max=getMax(a,n);
int count[max+1];
for(int i=0;i<=max;++i)
{
count[i]=0;
}
for(int i=0;i<n;i++)
{
count[a[i]]++;
}
for(i=1;i<=max;i++)
{
count[i]=count[i]+count[i-1];
}
for(i=n-1;i>=0;i--)
{
b[--count[a[i]]]=a[i];
```

```
// count[a[i]]--;
}
for(i=0;i<n;i++)
{
a[i]=b[i];
}
}void printArr(int a[],int n)
{
int i;
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
}
int main()
{
int n;
int a[50];
printf("enter the number of elements:");
scanf("%d",&n);
printf("enter the elements:");
for(int i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("elements before count sorting are:");
printArr(a,n);
countsort(a,n);
printf("\nelements after count sort are:\n");
printArr(a,n);
return 0;
}
```

OUTPUT:

```
enter the number of elements:5
enter the elements:2
9
5
8
3
elements before count sorting are:29
elements after count sort are:
2
3
5
```

8
9
5
8
3//radix sort
PROGRAM:

```
#include<stdio.h>
int getMax(int a[],int n)
{ int i;
int max=a[0];
for(i=0;i<n;i++)
{
if(a[i]>max)
max=a[i];
}
return max;
}
int countsort(int a[],int n,int place)
{ int i;
int b[n+1];
int count[10]={0};
for(int i=0;i<n;i++)
{
count[(a[i]/place)%10]++;
}
for(i=1;i<10;i++)
count[i]=count[i]+count[i-1];
for(i=n-1;i>=0;i--)
{
b[--count[(a[i]/place)%10]]=a[i];
}
for(i=0;i<n;i++)
a[i]=b[i];
}
void radixsort(int a[],int n)
{
int max=getMax(a,n);
for(int place=1;max/place>0;place=place*10)
{
countsort(a,n,place);}
}
void printArray(int a[],int n)
{
int i;
```

```
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
}
int main()
{
int a[50],n,i;
printf("enter the number of elements:");
scanf("%d",&n);
printf("enter the elements:\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("entered elememts before radixsort:\n");
printArray(a,n);
radixsort(a,n);
printf("\nentered elements after radix sorting are:\n");
printArray(a,n);
}
```

OUTPUT:

```
enter the number of elements:4
enter the elements:
7
3
1
5
entered elememts before radixsort:
7
3
1
5
entered elements after radix sorting are:
1
3
5
7
```

15. Write a C program to implement all the functions of a dictionary (ADT) using hashing.

PROGRAM:

```
#include <stdio.h>
int ht[10], i, found = 0,
key;
void insert_val()
{
int val,f=0;
printf( "\nEnter the
element to be inserted :
" );
scanf( "%d", &val );
key = ( val % 10 ) - 1;
if ( ht[key] == -1 )
{
ht[key] = val;
}
else
{
if ( key < 9 )
{
for ( i = key + 1;i < 10;i+
+ )
{
if ( ht[i] == -1 )
{
ht[i] = val;
f=1;
break;
}
}
}
if(f==0)
```

```
{
for ( i = 0;i < key;i++ )
{
if ( ht[i] == -1 )
{
ht[i] = val;break;
}
}
}
}
f=0;
}
void search_val()
{
int val, flag = 0;
printf( "\nEnter the
element to be searched ::
" );
scanf( "%d", &val );
key = ( val % 10 ) - 1;
if ( ht[ key ] == val )
flag = 1;
else
{
for (i = key + 1;i < 10;i+
+)
{
if(ht[i] == val)
{
flag = 1;
key = i;
break;
}
}
```

```
}
if (flag == 0)
{
for (i = 0;i < key;i++)
{
if (ht[ i ] == val)
{
flag = 1;
key = i;
break;
}
}
}if (flag == 1)
{
found=1;
printf("\n The item
searched was found at
position %d !", key + 1 );
}
else
{
key = -1;
printf( "\nThe item
searched was not found
in the hash table" );
}
}
void display()
{
for (i = 0;i < 10;i++)
printf( "\t%d", ht[ i ] );
}
void delete_val()
{
```

```
search_val();
if (found==1)
{
if ( key != -1 )
{
printf( "\nThe element
deleted is %d ",
ht[ key ] );
ht[ key ] = -1;
}
}
}
int main()
{
int option;
for ( i = 0;i < 10;i++ )
ht[i] = -1;
printf( "\n MENU \
n1.Insert \n2.Search \
n3.Delete \n4.Display \
n5.Exit");
do
{
printf( "\n Enter your
option.");
scanf( "%d", &option);
switch (option){
case 1:insert_val();
break;
case 2:search_val();
break;
case 3:delete_val();
break;
case 4:display();
```

```
break;
default:printf( "\nInvalid
choice entry!!!\n" );
break;
}
}while (option!=5);
return 0;
}
```

OUTPUT:

```
MENU
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your option.1
Enter the element to be
inserted : 2
Enter your option.1
Enter the element to be
inserted : 6
Enter your option.1
Enter the element to be
inserted : 7
Enter your option.1
Enter the element to be
inserted : 5
Enter your option.1
Enter the element to be
inserted : 4
Enter your option.1
Enter the element to be
inserted : 2
Enter your option.4
```

-1 2 2 4 5 6 7 -1 -1 -1
Enter your option.1
Enter the element to be
inserted : 4
Enter your option.4-1 2 2
4 5 6 7 4 -1 -1
Enter your option.2
Enter the element to be
searched :: 2
The item searched was
found at position 2 !
Enter your option.3
Enter the element to be
searched :: 4
The item searched was
found at position 4 !
The element deleted is 4
Enter your option.4
-1 2 2 -1 5 6 7 4 -1 -1
Enter your option.5
Invalid choice entry!!!

16. Write a C program that uses functions to perform the following:

a **Create a Binary Search Tree (BST).** b
Insert data in BST
c **Traverse the above BST recursively in Postorder.**

AND

17. Write a C program that uses functions to perform the following:

a **Deletion an element in BST**
b **Traverse the above BST non recursively in Inorder.**

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *left;
struct node *right;
};
struct node *newnode,*root,*temp,*ptr;
struct stack
{
struct node *tnode;struct node *next;
};
void creation(struct node *root)
{
int val;
printf("enter the root value:");
scanf("%d",&val);
root->data=val;
}
void insertion(struct node *root)
{
int val;
char flag;
printf("enter the data:");
scanf("%d",&val);
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=val;
```

```
temp=root;
while(temp!=0)
{
ptr=temp;
if (newnode->data<temp->data)
{
temp=temp->left;
flag='l';
}
else
{
temp=temp->right;
flag='r';
}
}
temp=newnode;
if(flag=='l')
ptr->left=newnode;
if(flag=='r')
ptr->right=newnode;
}
struct node *minvalue(struct node *node){
struct node *current=node;
while(current->left!=0)
{
current=current->left;
}
return current;
}
struct node *deleteNode(struct node *root, int key)
{
if (root == NULL)
return root;
if (key < root->data)
root->left = deleteNode(root->left, key);
else if (key > root->data)
root->right = deleteNode(root->right, key);
else
{
if (root->left == NULL)
{
struct node *temp = root->right;
free(root);
return temp;
```

```
}
else if (root->right == NULL)
{
struct node *temp = root->left;
free(root);
return temp;
}
struct node *temp = minvalue(root->right);
root->data = temp->data;
root->right = deleteNode(root->right, temp->data);
}
return root;
}
int isEmpty(struct stack *top)
{
return (top==0)?1:0;}
void push(struct stack **top,struct node *ptr)
{
struct stack *new=(struct stack*)malloc(sizeof(struct stack));
if (new==0)
{
printf("stack overflow\n");
exit(0);
}
new->tnode=ptr;
new->next=*top;
*top=new;
}
struct node *pop(struct stack **top_ref)
{
struct node *res;
struct stack *top;
if(isEmpty(*top_ref))
{
printf("stack underflow\n");
exit(0);
}
else
{
top=*top_ref;
res=top->tnode;
*top_ref=top->next;
free(top);
return res;
```

```
}
}
void inorder(struct node *root)
{
temp=root;
struct stack *s=0;
int done=0;
while(!done)
{
if(temp!=0){
push(&s,temp);
temp=temp->left;
}
else
{
if(!isEmpty(s))
{
temp=pop(&s);
printf("%d\t",temp->data);
temp=temp->right;
}
else
done=1;
}
}
}
void postorder(struct node *root)
{
if(root!=0)
{
postorder(root->left);
postorder(root->right);
printf("%d\t",root->data);
}
}
void main()
{
printf("1.creation\n");
printf("2.inserion\n");
printf("3.inorder\n");
printf("4.deletion\n");
printf("5.postorder\n");
int option,val;
do
```

```
{
printf("enter your option:");
scanf("%d",&option);
switch(option){
case 1:root=(struct node*)malloc(sizeof(struct node));
creation(root);
break;
case 2:insertion(root);
break;
case 3:inorder(root);
break;
case 4:printf("enter the value u want to delete:");
scanf("%d",&val);
deleteNode(root,val);
break;
case 5:postorder(root);
break;
default:printf("invalid option\n");
}
}while(option<=5);
}
```

OUTPUT:

```
1.creation
2.inserion
3.inorder
4.deletion
5.postorder
enter your option:1
enter the root value:25
enter your option:2
enter the data:30
enter your option:2
enter the data:28
enter your option:2
enter the data:36
enter your option:2
enter the data:38
enter your option:2
enter the data:50
enter your option:2
enter the data:48
enter your option:2enter the data:45
enter your option:2
enter the data:12
```

```
enter your option:2
enter the data:5
enter your option:2
enter the data:20
enter your option:2
enter the data:10
enter your option:2
enter the data:1
enter your option:2
enter the data:8
enter your option:2
enter the data:22
enter your option:2
enter the data:15
enter your option:2
enter the data:40
enter your option:3
1 5 8 10 12 15 20 22 25 28 30 36 38 40
45 48 50
enter your option:5
1 8 10 5 15 22 20 12 28 40 45 48 50 38
36 30 25
enter your option:4
enter the value u want to delete:36
enter your option:3
1 5 8 10 12 15 20 22 25 28 30 38 40 45
48 50
enter your option:6
invalid option
```

18. Write C program for implementing Depth first traversal and Breadth first traversal.

PROGRAM:

//DEPTH FIRST TRAVERSAL

```
#include <stdio.h>
#define MAX 5
void depth_first_search(int adj[][MAX],int visited[],int start)
{
int stack[MAX];
int top =-1, i;
printf("%c-",start+65);
visited[start] = 1;
stack[++top] = start;
while(top!=-1)
{
start = stack[top];
for(i = 0; i < MAX; i++)
{
if(adj[start][i] && visited[i] == 0)
{
stack[++top] = i;
printf("%c-", i + 65);
visited[i] = 1;
break;
}
}
if(i == MAX)
top--;
}
}
int main()
{
int adj[MAX][MAX];
int visited[MAX] = {0}, i, j;
printf("\n Enter the adjacency matrix: ");
for(i = 0; i < MAX; i++)
{
for(j = 0; j < MAX; j++)
scanf("%d", &adj[i][j]);
}
printf("DFS Traversal: ");depth_first_search(adj,visited,0);
printf("\n");
return 0;
```

```
}
```
OUTPUT:
Enter the adjacency matrix:
01010
10110
01001
11001
00110
DFS Traversal: A-B-C-E-D-

//BREADTH FIRST TRAVERSAL

```
#include<stdio.h>
#define MAX 5
void breadth_first_search(int adj[][MAX],int visited[],int start)
{
int queue[MAX],rear=-1,front=-1,i;
queue[++rear]=start;
visited[start]=1;
while(rear!=front)
{
start=queue[++front];
printf("%c\t",start+65);
for (i=0;i<MAX;i++)
{
if (adj[start][i]==1 && visited[i]==0)
{
queue[++rear]=i;
visited[i]=1;
}
}
}
}
int main()
{
int visited[MAX]={0};int adj[MAX][MAX],i,j;
printf("enter the adjacency matrix:\n");
for (i=0;i<MAX;i++)
{
for (j=0;j<MAX;j++)
scanf("%d",&adj[i][j]);
}
breadth_first_search(adj,visited,0);
return 0;
}
```

OUTPUT:
enter the adjacency matrix:
01010
10110
01001
11001
00110
ABDCE