PROGRAM 7

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void create();

void insert_front();

void insert_rear();

void display();

void delete_front();

void delete_rear();

int count=0;

struct node{

char usn[20],name[50],branch[10];

intsem;

unsigned long long int phno;

structnode*link;

};

struct node *first=NULL,*last=NULL,*temp=NULL,*p;

void main()   {

int ch,n,i;

while(1)    {

printf("1.create SLL 2.insert at front 3.insert at rear 4.display
5.delete at front 6.delete at rear 7.exit\n");

printf("enter choice\n");
scanf("%d",&ch);
switch(ch)   {
case 1:printf("enter the no.of students\n");
scanf("%d",&n);
for(i=1;i<=n;i++)
insert_front();  break;
case 2:insert_front();   break;
case 3:insert_rear();break;
case 4:display();break;
case 5:delete_front();break;
case 6:delete_rear();break;
case 7:exit(0);
default:printf("invalid choice\n");break;   }}}
void create()    {
char usn[20],name[50],branch[10];
intsem;
unsigned long long int phno;
temp=(struct node*)malloc(sizeof(struct node));
printf("enter usn,name,branch,sem,phno\n");
scanf("%s%s%s%d%llu",usn,name,branch,&sem,&phno);
strcpy(temp->usn,usn);
strcpy(temp->name,name);
strcpy(temp->branch,branch);
temp->sem=sem;
```

```
temp->phno=phno;
count++;     }
void insert_front()     {
if(first==NULL)   {
create();
temp->link=NULL;
first=temp;
last=temp;   }
else   {
create();
temp->link=first;
first=temp;    }}
void insert_rear()  {
if(first==NULL)  {
create();
temp->link=NULL;
first=temp;
last=temp;  }
else  {
create();
temp->link=NULL;
last->link=temp;
last=temp;   }}
void display()   {
```

```
if(first==NULL)  {
printf("list is empty\n");     }
else     {
p=first;
printf("content of list is\n");
while(p!=NULL)     {
printf("%s\t%s\t%s\t%d\t%llu\n",p->usn,p->name,p->branch,p-
>sem,p->phno);
p=p->link;    }
printf("total no.of students %d\n",count);      }}
void delete_front()      {
p=first;
if(first==NULL)       {
printf("list is empty\n");       }
else if(p->link==NULL)       {
printf("deleted node is %s\t%s\t%s\t%d\t%llu\n",p->usn,p-
>name,p->branch,p->sem,p->phno);
free(p);
first=NULL;
count--;       }
else       {
first=p->link;
printf("deleted node is %s\t%s\t%s\t%d\t%llu\n",p->usn,p-
>name,p->branch,p->sem,p->phno);
free(p);
count--;    }}
void delete_rear()          {
p=first;
if(first==NULL)             {
printf("list is empty\n");            }
else if(p->link==NULL)          {
printf("deleted node is %s\t%s\t%s\t%d\t%llu\n",p->usn,p-
>name,p->branch,p->sem,p->phno);
free(p);
first=NULL;
count--;            }
else          {
while(p->link!=last)
p=p->link;
printf("deleted node is %s\t%s\t%s\t%d\t%llu\n",last->usn,last-
>name,last->branch,last-
>sem,last->phno);
free(last);
p->link=NULL;
last=p;
count--;      }  }
```

PROGRAM 8

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void create(); void

insert_front(); void

insert_rear(); void

display();

void delete_front();

void delete_rear();

int count=0;

struct node           {

intssn;

char name[50],dept[20],desg[20];

floatsal;

unsigned long long int phno;

struct node *llink,*rlink;           };

struct node *first=NULL,*last=NULL,*temp;

main()          {

int ch,n,i;

while(1)         {

printf("1.create\n 2.insert_front\n 3.insert_rear\n 4.display\n
5.delete_front\n 6.delete_rear\n

7.exit\n");
printf("enter choice\n");
scanf("%d",&ch);
switch(ch)          {
case 1:printf("enter the number of employee\n");
scanf("%d",&n);
for(i=0;i<n;i++)
insert_rear();
break;
case 2:insert_front();break;
case 3:insert_rear();break;
case 4:display();break;
case 5:delete_front();break;
case 6:delete_rear();break;
case 7:exit(0);
default:printf("invalid choice\n");break;          }}}
void create()             {
intssn;
char name[50],dept[20],desg[20];
floatsal;
unsigned long long int phno;
temp=(struct node*)malloc(sizeof(struct node));
temp->llink=temp->rlink=NULL;
printf("enter ssn,name,dept,desg,salaryand phno\n");
scanf("%d%s%s%s%f%llu",&ssn,name,dept,desg,&sal,&phno);
```

```
temp->ssn=ssn;
strcpy(temp->name,name);
strcpy(temp->dept,dept);
strcpy(temp->desg,desg);
temp->sal=sal;
temp->phno=phno;
count++;                }
void insert_front()             {
if(first==NULL)               {
}       else
{           create();
first=temp;
last=temp;
create();
temp->rlink=first;
first->llink=temp;
first=temp;              }}
void insert_rear()         {
if(first==NULL)           {
create();
first=temp;
else            {  }}
last=temp;            }
create();
```

```
last->rlink=temp;

temp->llink=last;

temp->rlink=NULL;

last=temp;

void display()              {

struct node *p;

if(first==NULL)              {

printf("listisempty\n");

return;              }

p=first;

printf("contentsoflist\n");

while(p!=NULL)              {

printf("%d\t%s\t%s\t%s\t%f\t%llu\n",p->ssn,p->name,p->dept,p-
>desg,p->sal,p->phno);

p=p->rlink;              }

printf("total no. of employee %d\n",count);              }

void delete_front()              {

struct node *p;

if(first==NULL)              {

printf("list is empty,cannot delete\n");              }

else if(first->rlink==NULL)              {

printf("deleted data is %d\t%s\t%s\t%s\t%f\t%llu\n",first-
>ssn,first->name,first->dept,first->desg,first-

>sal,first->phno);
```

```
first=NULL;

free(first);

count--;        }

else        {

p=first;

first=p->rlink;

printf("deleted data is %d\t%s\t%s\t%s\t%f\t%llu\n",p->ssn,p-
>name,p->dept,p->desg,p->sal,p-

>phno);

free(p);

count--;                }}

void delete_rear()              {

struct node*p;

if(first==NULL)             {

printf("list is empty,cannot delete\n");                }

else if(first->rlink==NULL)             {

printf("deleted data is %d\t%s\t%s\t%s\t%f\t%llu\n",first-
>ssn,first->name,first-

>dept,first->desg,first->sal,first->phno);

first=NULL;

free(first);

count--;            }

else            {

p=last;
```

```
last=p->llink;
printf("deleted data is %d\t%s\t%s\t%s\t%f\t%llu\n",p->ssn,p->name,p->dept,p->desg,p-
>sal,p->phno);
free(p);
last->rlink=NULL;
count--;          } }
```

Program 10(c0ntinuation)

```
case 3: search(root);
break;
case 4: exit(0);         }}}
```

program 10:

```
#include<stdio.h>

#include<stdlib.h>

struct BST          {

int data;

struct BST *lchild;

struct BST *rchild;            };

typedef struct BST * NODE;

NODE create()                 {

NODE temp;

temp = (NODE) malloc(sizeof(struct BST));

printf("\nEnter The value: ");

scanf("%d", &temp->data);

temp->lchild = NULL;

temp->rchild = NULL;

return temp;                 }

void insert(NODE root, NODE newnode);

void inorder(NODE root);

void preorder(NODE root);

void postorder(NODE root);

void search(NODE root);

void insert(NODE root, NODE newnode)              {

if (newnode->data < root->data)            {

if (root->lchild == NULL)
```

```
root->lchild = newnode;

else

insert(root->lchild, newnode);          }

if (newnode->data > root->data)          {

if (root->rchild == NULL)

root->rchild = newnode;

else

insert(root->rchild, newnode);        } }

void search(NODE root)          {

int key;

NODE cur;

if(root == NULL)           {

printf("\nBST is empty.");

return;              }

printf("\nEnter Element to be searched: ");

scanf("%d", &key);

cur = root;

while (cur != NULL)              {

if (cur->data == key)              {

printf("\nKey element is present in BST");

return;                    }

if (key < cur->data)

cur = cur->lchild;

else
```

```
cur = cur->rchild;              }
printf("\nKey element is not found in the BST");                }
void inorder(NODE root)                {
if(root != NULL)                  {
inorder(root->lchild);
printf("%d ", root->data);
inorder(root->rchild);             }}
void preorder(NODE root)             {
if (root != NULL)                   {
printf("%d ", root->data);
preorder(root->lchild);
preorder(root->rchild);               }}
void postorder(NODE root)                  {
if (root != NULL)                 {
postorder(root->lchild);
postorder(root->rchild);
printf("%d ", root->data);             } }
int main()              {
int ch, key, val, i, n;
NODE root = NULL, newnode;
while(1)           {
printf("\n~BST MENU~");
printf("\n1.Create a BST");
printf("\n2.Search");
printf("\n3.BST Traversals: ");
printf("\n4.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)                    {
case 1: printf("\nEnter the number of elements: ");
scanf("%d", &n);
for(i=1;i<=n;i++)                  {
newnode = create();
if (root == NULL)
root = newnode;
else
insert(root, newnode);                  }
break;
case 2: if (root == NULL)
printf("\nTree Is Not Created");
else                    {
printf("\nThe Preorder display : ");
preorder(root);
printf("\nThe Inorder display : ");
inorder(root);
printf("\nThe Postorder display : ");
postorder(root);                }
break;
```

PROGRAM 11

```
#include<stdio.h>
#include<stdlib.h>
int n,a[10][10],i,j,source,s[10],choice,count;
void bfs(int n,int a[10][10],int source,int s[])          {
int q[10],u;
intfront=1,rear=1;
s[source]=1;
q[rear]=source;
while(front<=rear)              {
u=q[front];
front=front+1;
for(i=1;i<=n;i++)
if(a[u][i]==1&&s[i]==0)              {
rear=rear+1;
q[rear]=i;
s[i]=1;           } } }
void dfs(int n,int a[10][10],int source,int s[])                 {
s[source]=1;
for(i=1;i<=n;i++)
if(a[source][i]==1 &&s[i]==0)
dfs(n,a,i,s);             }
int main()              {
printf("Enter the number of nodes : \n");
```

```
scanf("%d",&n);
printf("\n Enter the adjacency matrix\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
while(1)            {
printf("\n\n1.BFS\n 2.DFS\n 3.Exit\n");
printf("\nenter your choice\n");
scanf("%d",&choice);
switch(choice)          {
case 1: printf("\n Enter the source :\n");
scanf("%d",&source);
for(i=1;i<=n;i++)
s[i]=0;
bfs(n,a,source,s);
for(i=1;i<=n;i++)           {
if(s[i]==0)
printf("\n The node %d is not reachable\n",i);
else
printf("\n The node %d is reachable\n",i);            }
break;
case 2:printf("\nEnter the source vertex :\n");
scanf("%d",&source);
count=0;
for(i=1;i<=n;i++)
s[i]=0;
dfs(n,a,source,s);
for(i=1;i<=n;i++)
if(s[i])
count=count+1;
if(count==n)
printf("\nThe graph is connected.");
else
printf("\nThe graph is not connected.");
break;
case 3: exit(0);               }}}
```

PROGRAM 12

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
void display(int a[MAX]);
int create(int num);
void linearprob(int a [MAX],int key,int num);
void main()         {
int a[MAX],i,num,key,ans=1;
printf("collission handling by linear probing\n");
for(i=0;i<MAX;i++)
```

```
a[i]= -1;
do                {
printf("enter the data\n");
scanf("%4d",&num);
key=create(num);
linearprob(a,key,num);
printf("do yuou want to continue[1/0]\n");
scanf("%d",&ans);
}while(ans);
display(a);              }
int create(int num)              {
int key;
key=num%100;
return key;              }
void linearprob(int a[MAX],int key, int num)            {
intflag=0,count=0,i;
if(a[key]==-1)
a[key]=num;
else                     {
printf("\n collision deleted\n");
i=0;
while((i<key)&&(flag==0))           {
if(a[i]==-1)              {
a[i]=num;
flag=1;
break;                  }
i++;                    }}}
void display(int a[MAX])           {
int ch,i;
printf("\n 1.display all 2.filtered display\n");
printf("enter choice\n");
scanf("%d",&ch);
if(ch==1)              {
for(i=0;i<MAX;i++)
printf("%d\t%d\n",i,a[i]);                  }
else                       {
for(i=0;i<MAX;i++)            {
if(a[i]!=-1)            {
printf("%d\t%d\n",i,a[i]);
continue;             } }}}
```