Experiment 1a

```
#include<stdio.h> #include<conio.h> #include<stdlib.h>

#define MAX 20 void main()

{

int choice,temp,len,pos,temp2,l,r,flag=0; char stri[MAX],stro[MAX];

printf("\n Enter a string with max character %d ",MAX); scanf("%s",stri); for(temp=0,len=0;stri[temp]!='\0';temp++,len++);

printf("\n Length of the given String is %d",len);

while(1)

{

printf("\n ****** \n 1.Print Substring\n 2.Copy string into other array\n 3.Reverse the given string\n 4.Check whether the string is Palindrome\n 5.Compare string\n 6.Exit \n");

scanf("%d",&choice); switch(choice)

{

case 1 printf("\n Enter the position from where the sub string must get displayed "); scanf("%d",&pos);

for(temp=0;temp<len-pos;temp++)

{

stro[temp]=stri[pos+temp];

}

stro[temp]='\0';

printf("\nSubString is %s",stro); break;

case 2 for(temp=0;stri[temp]!='\0';temp++)

{

stro[temp]=stri[temp];

}

stro[temp]='\0'; printf("\n Entered string is %s",stri); printf("\n String after copying %s",stro); break;

case 3

printf("\n Entered string is %s",stri); for(temp=len-1,temp2=0;temp>=0;temp--,temp2++)

{

stro[temp2]=stri[temp];

}

stro[temp2]='\0';

printf("\n String after reversing %s",stro); break;

case 4

printf("\n Entered string is %s",stri); l=0,r=len-1; while(stri[l++]!=stri[r--])

{

flag=1; break;

}

if(flag==1)

printf("\n Entered string is not a plaindrome."); else printf("\n Entered string is a palindrome."); break;
```

```
case 5
printf("\n Enter another string to perform comparision "); scanf("%s",stro); temp=0;
while(stri[temp]==stro[temp] && stri[temp]!='\0')
{
if(stri[temp]>stro[temp])
printf("\n Orginal string is greater than Another string"); else if(stri[temp]<stro[temp])
printf("\n Orginal string is smaller than Another string"); else
printf("Both strings are Equal"); temp++;
break;
case 6 exit(1);
}
}
}
}
```

Enter a string with max character 20 HelloWelcome

Length of the given String is 12

******

1. Print Substring

Copy string into other array 3.Reverse the given string

1. Check whether the string is Palindrome

Compare string 6.Exit

1

Enter the position from where the sub string must get displayed 2

SubString is lloWelcome

******

1. Print Substring

Copy string into other array 3.Reverse the given string

4.Check whether the string is Palindrome 5.Compare string

6.Exit 3

Entered string is HelloWelcome String after reversing emocleWolleH

******

1. Print Substring

Copy string into other array 3.Reverse the given string

4.Check whether the string is Palindrome 5.Compare string

6.Exit 4

Entered string is HelloWelcome Entered string is not a plaindrome.

******

1. Print Substring

Copy string into other array 3.Reverse the given string

4.Check whether the string is Palindrome 5.Compare string

6.Exit 5

Enter another string to perform comparision HelloWorld Both strings are not Equal

******

1. Print Substring

Copy string into other array 3.Reverse the given string

4.Check whether the string is Palindrome 5.Compare string

6.Exit 6

Experiment 1b

#include<stdio.h> #include<conio.h> #include<stdlib.h> #define MAX 20

void main()

{

int choice,temp,len,pos,temp2,l,r,flag=0; char stri[MAX],stro[MAX]; printf(" Enter a string with max character ",MAX); scanf("%s",stri);

for(temp=0,len=0;stri[temp]!='\0';temp++,len++); printf("\nLength of the given String is %d",len);

while(1){

printf("\n MENU \n 1.Print substring\n 2.Copy string into other array\n 3.Reverse the given string\n 4.Check string as palindrome\n 5.Compare string

\n 6.Exit\n"); scanf("%d",&choice); switch(choice){

case 1 printf("\n Enter the position from where the sub string must get displayed ");

scanf("%d",&pos); for(temp=0;temp<len-pos;temp++){ stro[temp]=stri[pos+temp];

}

stro[temp]='\0';

printf("\n SubString is ",stro); break; case 2 for(temp=0;stri[temp]!='\0';temp++){ stro[temp]=stri[temp];

}

stro[temp]='\0'; printf("\nEntered string is %s",stri); printf("\n String after copying %s",stro);

break; case 3

printf("\n Entered string is %s",stri); for(temp=len-1,temp2=0;temp>=0;temp--,temp2++){ stro[temp2]=stri[temp];

```
}
stro[temp2]='\0';
printf("\n String after reversing %s",stro); break; case 4
printf("\n Entered string is %s",stri); l=0,r=len-1; while(stri[l++]!=stri[r--]){
flag=1; break;
}
if(flag==1)
printf("\n Entered string is not a plaindrome"); else printf("\n Entered string is a palindrome"); break; case 5
printf("\nEnter another string to perform comparision "); scanf("%s",stro);
temp=0;
while(stri[temp]==stro[temp] && stri[temp]!='\0')
{
if(stri[temp]>stro[temp]) printf("\n Orginal string is greter than Another string");
else if(stri[temp]<stro[temp])
printf("\n Orginal string is smaller than Another string"); else
printf("\n Orginal string is equal than Another string"); temp++;} break; case 6 exit(1);
}//switch_ends
}//while_ends
}//main_ends
```

Length of the given String is 9

MENU

Print substring

1. Copy string into other array 3.Reverse the given string 4.Check string as palindrome 5.Compare string

6.Exit 1

Enter the position from where the sub string must get displayed 2

SubString is MENU

1. Print substring

Copy string into other array 3.Reverse the given string 4.Check string as palindrome 5.Compare string

6.Exit 2

Entered string is HiWelcome String after copying HiWelcome MENU

1. Print substring

Copy string into other array 3.Reverse the given string 4.Check string as palindrome 5.Compare string

6.Exit 3

Entered string is HiWelcome String after reversing emocleWiH MENU

1. Print substring

Copy string into other array 3.Reverse the given string 4.Check string as palindrome 5.Compare string

6.Exit 4

Entered string is HiWelcome Entered string is not a plaindrome MENU

1. Print substring

Copy string into other array 3.Reverse the given string 4.Check string as palindrome 5.Compare string

6.Exit 5

Enter another string to perform comparision Hi

Orginal string is not than Another string MENU

Print substring

1. Copy string into other array 3.Reverse the given string 4.Check string as palindrome

5.Compare string 6.Exit

Experiment 2 a

#include <stdio.h> #include <string.h> #include<stdlib.h> typedef struct Employee

{

int code;

char name[20]; int salary;

}Employee;

void read(Employee *st,int n);

void insert(Employee *st,int position,int n); void Delete(Employee *st,int position,int n); Employee * search(Employee *st,int code,int n); void print(Employee *st,int n);

void sort(Employee *st,int n); void modify(Employee *st,int n); int main()

{

Employee st[30],*pos; int n,i,ch,code,position;

do

{

printf("\n 1.Create \n 2.Insert \n 3.Delete \n 4.Search "); printf("\n 5.Print \n 6.Sort \n 7.Modify\n 8.Quit"); printf("\n Enter Your Choice");

scanf("%d",&ch); switch(ch)

{

case 1 printf("\n Enter Number of Employees "); scanf("%d",&n);

```
read(st,n); break;
case 2 printf("\n Enter the position(no of %d) ",n); scanf("%d",&position);
if(position<=n+1)
{
insert(st,position,n); n++;
print(st,n);
}
else
printf("\n Can not insert");
break;
case 3printf("\n Enter the code "); scanf("%d",&code); pos=search(st,code,n);
if(pos != NULL )
{
Delete(st,pos-st,n); n--;
print(st,n);
}
else
printf("\n Can not delete "); break;
case 4 printf("\n Enter code "); scanf("%d",&code); pos=search(st,code,n); if(pos==NULL)
printf("\nnot found"); else
{ printf("\n Found at location=%ld",pos-st+1);
printf("\n %s\t %d\t %d",pos->name,pos->code,pos->salary);
}
break;
case 5 print(st,n); break;
case 6 sort(st,n);print(st,n); break;
case 7 modify(st,n);
print(st,n); break;
case 8
exit(0); break;
defaultprintf("Invalid choice XX"); break;
}
}while(ch!=8);
}
void insert( Employee *st,int position,int n)
{ int i;
printf("\n Enter data (Name -- Code -- Salary) "); for(i=n-1;i>=position-1;i--)
*(st+i+1)=*(st+i);
scanf("%s %d %d",(st+position-1)->name,&(st+position-1)-
```

```
>code,&(st+position-1)->salary);
}
void Delete(Employee st[],int position,int n)
{ int i;
for(i=position+1;i<n;i++)
*(st+i-1)=*(st+i);
}
Employee * search(Employee *st,int code,int n)
{ Employee *p;
for(p=st;p<st+n;p++) if(code==p->code) return(p);
return(NULL);
}
void print(Employee *st,int n)
{ Employee *p;
for(p=st;p<st+n;p++)
printf("\n %20s %5d %5d",p->name,p->code,p->salary);
}
void read(Employee *st,int n)
{ Employee *p;
printf("\n Enter data (Name -- Code -- Salary) "); for(p=st;p<st+n;p++)
scanf("%s %d %d",p->name,&p->code,&p->salary);
}
void modify(Employee *st, int n)
{
int code; Employee *pos;
printf("\n Enter the code "); scanf("%d",&code); pos=search(st,code,n); if(pos==NULL)
printf("\n No such Number "); else
{
printf("\n Enter data (Name -- Code -- Salary) "); scanf("%s %d %d",pos->name,&pos->code,&pos->salary);
}
}
void sort(Employee *st,int n)
{ int i,j;
Employee temp,*p; for(i=1;i<n;i++)
for(p=st;p<st+n-i;p++) if(p->code > (p+1)->code)
{
temp=*p;
*p=*(p+1);
*(p+1)=temp;
```

}

}

1.Create 2.Insert 3.Delete 4.Search 5.Print 6.Sort 7.Modify 8.Quit

Enter Your Choice1

Enter Number of Employees 3

Enter data (Name -- Code -- Salary) Anil 1001 7500 Krushna 1002 7200

Trupti 1003 3444

1. Create 2.Insert 3.Delete 4.Search 5.Print 6.Sort 7.Modify 8.Quit

Enter Your Choice2

Enter the position(no of 3) 1002

Can not insert 1.Create

1. Insert 3.Delete 4.Search 5.Print 6.Sort 7.Modify 8.Quit

Enter Your Choice2

Enter the position(no of 3) 1

Enter data (Name -- Code -- Salary) Soham 1007 8000

Soham 1007 8000

Anil 1001 7500

Krushna 1002 7200

Trupti 1003 3444 1.Create

2.Insert 3.Delete 4.Search 5.Print 6.Sort 7.Modify 8.Quit

Enter Your Choice5

Soham 1007 8000

Anil 1001 7500

Krushna 1002 7200

Trupti 1003 3444

1. Create 2.Insert 3.Delete 4.Search 5.Print 6.Sort 7.Modify 8.Quit

Enter Your Choice3

Enter the code 1007

Anil 1001 7500

Krushna 1002 7200

Trupti 1003 3444 1.Create

1. Insert 3.Delete 4.Search 5.Print 6.Sort 7.Modify 8.Quit

Enter Your Choice5

Anil 1001 7500

Krushna 1002 7200

Trupti 1003 3444 1.Create

2.Insert 3.Delete

4.Search 5.Print 6.Sort 7.Modify 8.Quit

Enter Your Choice6

Anil 1001 7500

Krushna 1002 7200

Trupti 1003 3444 1.Create

2.Insert 3.Delete 4.Search 5.Print 6.Sort 7.Modify 8.Quit

Enter Your Choice7

Enter the code 1003

Enter data (Name -- Code -- Salary) trupti 1009 6700

Anil 1001 7500

Krushna 1002 7200

trupti 1009 6700 1.Create

2.Insert 3.Delete 4.Search

5.Print 6.Sort 7.Modify 8.Quit

Enter Your Choice5

Anil 1001 7500

Krushna 1002 7200

trupti 1009 6700 1.Create

2.Insert 3.Delete 4.Search 5.Print 6.Sort 7.Modify 8.Quit

Enter Your Choice8

Experiment 2b

```
#include<stdio.h> #include<conio.h>

//DECLARATION OF STRUCTURE

struct Employee{ int id;

char name[20]; int age;

long int salary;

}e[15];

void create(int p); void display(int y);

void modify(int m,int c); void append(int x);

int search(int key,int a); void sort(int b);

void main()

{ int choice,no,i,n,num; char op;

printf("\n Enter how many records you want to Add? \n"); scanf("%d",&n);

create(n); do

{

printf("menu\n 1.Display\n 2.Modify\n 3.Mppend\n 4.Search\n 5.Sort\n"); printf("\n Enter your choice \n");

scanf("%d",&choice); switch(choice) //switch structure

{

case 1display(n); break;

case 2printf("\n Enter the Employee id to be modified \n"); scanf("%d",&num);

modify(num,n); //calling modify() function break;

case 3append(n); n++;

break; case 4

printf("\n Enter the Employee number to be searched \n"); scanf("%d",&num);

i=search(num,n);

//calling search() function which returns int value if(i==-1)

printf("\n Employee Not Found XX\n");

else

printf("\n Employee found at %d location\n",i); break;

case 5sort(n); //calling sort function break;

defaultprintf("\n Invalid choice.\n"); break;

}

printf("\n Do you want to continue? \n"); op=getch(); putch(op);

}while(op=='Y'||op=='y'); getch();

}

void create(int p)

{ int i; for(i=0;i<p;i++)
```

```
{ printf("\n Enter information for [%d] Employee -->\n",i+1); printf("\n Enter the Employee id ");

scanf("%d",&e[i].id); printf("\n Enter name \n"); scanf("%s",e[i].name); printf("\n Enter age \n"); scanf("%d",&e[i].age); printf("\n Enter salary \n");
scanf("%ld",&e[i].salary);

}

}

void display(int y)

{ int i;

printf("\n **The information for the Employees**\n"); printf("\n id \t Name \t Age \t Salary\n"); for(i=0;i<y;i++)

printf("\n %d \t %s\t %d\t %d",e[i].id,e[i].name,e[i].age,e[i].salary);

}

void modify(int m,int c)

{ int pos; pos=search(m,c); if(pos==-1)

printf("\n Employee id does not Exist.\n"); else {

printf("\n Enter the information for [%d] Employee again \n",pos); printf("\n Enter the new Employee id \n"); scanf("%d",&e[pos].id);

printf("\n Enter new name \n"); scanf("%s",e[pos].name); printf("\n Enter new age \n"); scanf("%d",&e[pos].age); printf("\n Enter new salary \n");

scanf("%ld",&e[pos].salary);

}

}

void append(int x)

{ printf("\n Enter the new record for [%d] Employee \n",x+1); printf("\n Enter the new Employee id \n"); scanf("%d",&e[x].id);

printf("\n Enter new name \n"); scanf("%s",e[x].name); printf("\n Enter new age \n"); scanf("%d",&e[x].age); printf("\n Enter new salary \n");
scanf("%ld",&e[x].salary);

}

int search(int key,int a)

{

int i;

for (i=0;i<a;i++) if(key==e[i].id) return(i+1); return -1;

}

void sort(int b)

{

int i,j;

struct Employee temp; for(i=0; i<b-1 ;i++) for(j=0;j<b-1-i;j++)

if(e[j].id>e[j+1].id)

{ temp=e[j];

e[j]=e[j+1]; e[j+1]=temp;

}

printf("\n Records sorted in ascending order of ids. \n");

}
```

Enter how many records you want to Add? 2

Enter information for [1] Employee -->

Enter the Employee id 101

Enter name

Amit

Enter age 23

Enter salary 10000

Enter information for [2] Employee -->

Enter the Employee id 109

Enter name Trupti

Enter age 23

Enter salary 9000

menu 1.Display 2.Modify 3.Mppend 4.Search 5.Sort

Enter your choice 1

The information for the Employees

id Name Age Salary

101 Amit 23 10000
109 Trupti 23 9000

Do you want to continue? ymenu

1.Display 2.Modify 3.Mppend 4.Search 5.Sort

Enter your choice

2

Enter the Employee id to be modified 109

Enter the information for [2] Employee again

Enter the new Employee id 187

Enter new name Trupti

Enter new age 24

Enter new salary 10000

Do you want to continue? ymenu

1.Display 2.Modify 3.Mppend 4.Search 5.Sort

Enter your choice

3

Enter the new record for [3] Employee

Enter the new Employee id 105

Enter new name umesh

Enter new age 34

Enter new salary 89000

Do you want to continue? ymenu

1.Display 2.Modify 3.Mppend 4.Search 5.Sort

Enter your choice 4

Enter the Employee number to be searched 109

Employee found at 2 location

Do you want to continue? ymenu

1. Display 2.Modify 3.Mppend 4.Search 5.Sort

Enter your choice

5

Records sorted in ascending order of ids.

Do you want to continue? n

Experiment 3a

#include <stdio.h> #include <stdlib.h>

int stack[100],choice,n,top,x,i; void push(void);

void pop(void); void display(void); int main()

{

top=-1;

printf("\n Enter the size of STACK [MAX=100]"); scanf("%d",&n);

printf("\n\t OPERARIONS");

printf("\n\t "); printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT"); do

{

printf("\n Enter your Choice"); scanf("%d",&choice); switch(choice)

{

case 1

```
{
push(); break;
}
case 2
{
pop(); break;
}
case 3
{
display(); break;
}
case 4
{
printf("\n\t EXIT POINT "); break;
}
default
{
printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
}
}
}
while(choice!=4); return 0;
}
void push()
{
if(top>=n-1)
{
printf("\n\tSTACK is over flow. ");
}
else
{
printf(" Enter a value to be pushed"); scanf("%d",&x);
top++; stack[top]=x;
}
}
void pop()
{
if(top<=-1)
{
```

```
printf("\n\t Stack is under flow");
}
else
{
printf("\n\t The popped elements is %d ",stack[top]); top--;
}
}
void display()
{
if(top>=0)
{
printf("\n The elements in the Stack are as follows \n"); for(i=top; i>=0; i--)
printf("\n%d",stack[i]); printf("\n Press Next Choice ");
}
else
{
printf("\n The Stack is Empty");
}
}
```

Enter the size of STACK [MAX=100]3

OPERATIONS

PUSH

- 1. POP

DISPLAY

- 1. EXIT

Enter your Choice1

Enter a value to be pushed12

Enter your Choice1

Enter a value to be pushed15

Enter your Choice3

The elements in the Stack are as follows

15

12

Press Next Choice Enter your Choice2

The popped elements is 15 Enter your Choice2

The popped elements is 12 Enter your Choice2

Stack is under flow Enter your Choice3

The Stack is Empty Enter your Choice4

EXIT POINT

```
#include <stdio.h> #include <stdlib.h> #define n 5

int main()

{

int queue[n],ch=1,front=0,rear=0,i,j=1,x=n; printf("QUEUE OPERATIONS USING ARRAY");

printf("\n1.Insertion \n2.Deletion \n3.Display \n4.Exit"); while(ch)

{

printf("\nEnter the Choice "); scanf("%d",&ch);

switch(ch)

{

case 1

if(rear==x)

printf("\n Queue is Full "); else

{

printf("\n Enter no %d",j++); scanf("%d",&queue[rear++]);

}

break; case 2

if(front==rear)

{

printf("\n Queue is Empty");

}

else

{

printf("\n Deleted Element is %d",queue[front++]); x++;

}

break; case 3

printf("\nQueue Elements are \n "); if(front==rear)

printf("\n Queue is Empty ");

else

{

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

```
{
printf("%d",queue[i]); printf("\n");
}
break; case 4
exit(0); default
printf("Wrong Choice Please see the options ");
}
}
}
return 0;
}
```

QUEUE OPERATIONS USING ARRAY

1.Insertion 2.Deletion 3.Display 4.Exit

Enter the Choice 1 Enter no 112

Enter the Choice 1 Enter no 254

Enter the Choice 2

Deleted Element is 12 Enter the Choice 3

Queue Elements are 54

Enter the Choice 1 Enter no 354

Enter the Choice 3

Queue Elements are 54

54

Enter the Choice 4

Experiment 4

```
#include<stdio.h> #include<stdlib.h>
/*—-Function Prototypes—-*/ void create();
void display();
void insert_begin(); void insert_end(); void insert_pos(); void delete_begin(); void delete_end(); void delete_pos();
struct node
{
int info;
struct node *next;
};
struct node *start=NULL; struct node *last=NULL; int main() //main() starts
{
int choice; while(1){
printf("\n SINGLE LINKED LIST OPERATIONS \n");
printf("\n menu \n"); printf("\n 1.Create \n"); printf("\n 2.Display \n");
printf("\n 3.Insert at the Beginning \n"); printf("\n 4.Insert at the End \n");
```

```
printf("\n 5.Insert at specified position or mid \n"); printf("\n 6.Delete from beginning \n");
printf("\n 7.Delete from the end \n");
printf("\n 8.Delete from specified position or mid \n"); printf("\n 9.Exit \n");
printf("Enter your choice\t"); scanf("%d",&choice); switch(choice)
{
case 1 create(); break; case 2 display();
break; case 3
insert_begin(); break;
case 4 insert_end(); break;
case 5 insert_pos(); break;
case 6 delete_begin(); break;
case 7 delete_end(); break;
case 8 delete_pos(); break;
case 9 exit(0); break; default
printf("\n Wrong Choice\n"); break;
}//end of switch()
}
return 0;
}//end of main() void create()
{
struct node *temp,*ptr;
temp=(struct node *)malloc(sizeof(struct node)); if(temp==NULL)
{
printf("\n Out of Memory Space\n"); exit(0);
}
printf("\n Enter the data value for the node \t"); scanf("%d",&temp->info);
temp->next=NULL; if(start==NULL)
{
start=temp;
}
else
{
ptr=start;
while(ptr->next!=NULL)
{
ptr=ptr->next;
}
ptr->next=temp;
}
```

```
}//end of create() void display()
{
struct node *ptr; if(start==NULL)
{
printf("\n List is empty \n"); return;
}
else
{
ptr=start;
printf("\n The List elements are \n"); while(ptr!=NULL)
{
printf("%d\t",ptr->info ); ptr=ptr->next ;
}//end of while
}//end of else
}//end of display()
void insert_begin()
{
struct node *temp;
temp=(struct node *)malloc(sizeof(struct node)); if(temp==NULL)
{
printf("\n Out of Memory Space \n"); return;
}
printf("\n Enter the data value for the node \t" ); scanf("%d",&temp->info);
temp->next =NULL; if(start==NULL)
{
start=temp;
}
else
{
temp->next=start; start=temp;
}
}//end of insert_begin() void insert_end()
{
struct node *temp,*ptr;
temp=(struct node *)malloc(sizeof(struct node)); if(temp==NULL)
{
printf("\nOut of Memory Space\n"); return;
}
printf("\n Enter the data value for the node \t" ); scanf("%d",&temp->info );
```

```
temp->next =NULL; if(start==NULL)
{
start=temp;
}
else
{
ptr=start;
while(ptr->next !=NULL)
{
ptr=ptr->next ;
}
ptr->next =temp;
}
}//end of insert_end void insert_pos()
{
struct node *ptr,*temp; int i,pos;
temp=(struct node *)malloc(sizeof(struct node)); if(temp==NULL)
{
printf("\nOut of Memory Space\n"); return;
}
printf("\nEnter the position for the new node to be inserted\t"); scanf("%d",&pos);
printf("\n Enter the data value of the node \t"); scanf("%d",&temp->info) ;
temp->next=NULL; if(pos==0)
{
temp->next=start; start=temp;
}
else
{
for(i=0,ptr=start;i<pos-1;i++)
{
ptr=ptr->next; if(ptr==NULL)
{
printf("\n Position not found \n"); return;
}
}
temp->next =ptr->next ; ptr->next=temp;
}//end of else
}//end of insert_pos void delete_begin()
{
```

```
struct node *ptr; if(ptr==NULL)
{
printf("\n List is Empty \n"); return;
}
else
{
ptr=start; start=start->next ;
printf("\n The deleted element is %d\t",ptr->info); free(ptr);
}
}//end of delete_begin() void delete_end()
{
struct node *temp,*ptr; if(start==NULL)
{
printf("\n List is Empty "); exit(0);
}
else if(start->next ==NULL)
{
ptr=start; start=NULL;
printf("\n The deleted element is %d\t",ptr->info); free(ptr);
}
else
{
ptr=start;
while(ptr->next!=NULL)
{
temp=ptr; ptr=ptr->next;
}
temp->next=NULL;
printf("\n The deleted element is %d\t",ptr->info); free(ptr);
}
}//end of delete_begin() void delete_pos()
{
int i,pos;
struct node *temp,*ptr; if(start==NULL)
{
printf("\n The List is Empty \n"); exit(0);
}
else
{
```

```
printf("\n Enter the position of the node to be deleted \t"); scanf("%d",&pos);

if(pos==0)

{

ptr=start; start=start->next ;

printf("\nThe deleted element is%d\t",ptr->info ); free(ptr);

}

else

{

ptr=start; for(i=0;i<pos;i++)

{

temp=ptr; ptr=ptr->next ; if(ptr==NULL)

{

printf("\n Position not Found \n"); return;

}

}

temp->next =ptr->next ;

printf("\n The deleted element is %d\t",ptr->info ); free(ptr);

}

}//end of else

}
```

SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid 9.Exit

Enter your choice 1

Enter the data value for the node 13

SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid 9.Exit

Enter your choice 1

Enter the data value for the node 16

SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid 9.Exit

Enter your choice 1

Enter the data value for the node 99

SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid
3. Exit

Enter your choice 2

The List elements are 13 16 99

SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid 9.Exit

Enter your choice 3

Enter the data value for the node 43

SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid
3. Exit

Enter your choice 4

Enter the data value for the node 56

SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid 9.Exit

Enter your choice 5

Enter the position for the new node to be inserted 6

Enter the data value of the node 500

Position not found

SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid 9.Exit

Enter your choice 2

The List elements are

43 13 16 99 56

SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid 9.Exit

Enter your choice 6

The deleted element is 43 SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

1. Insert at the Beginning
2. Insert at the End
3. Insert at specified position or mid 6.Delete from beginning
4. Delete from the end
5. Delete from specified position or mid 9.Exit

Enter your choice 7

The deleted element is 56 SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid 9.Exit

Enter your choice 8

Enter the position of the node to be deleted 2

The deleted element is 99 SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid 9.Exit

Enter your choice 2

The List elements are 13 16

SINGLE LINKED LIST OPERATIONS

menu 1.Create 2.Display

3.Insert at the Beginning 4.Insert at the End

5.Insert at specified position or mid 6.Delete from beginning

1. Delete from the end
2. Delete from specified position or mid 9.Exit

Enter your choice

Experiment 5

#include <stdio.h> #include <stdlib.h>

struct BST

{

int data;

struct BST *left; struct BST *right;

};

typedef struct BST NODE; NODE *node;

NODE* createtree(NODE *node, int data)

{

if (node == NULL)

{

NODE *temp;

temp= (NODE*)malloc(sizeof(NODE)); temp->data = data;

temp->left = temp->right = NULL; return temp;

}

if (data < (node->data))

{

node->left = createtree(node->left, data);

}

else if (data > node->data)

{

node -> right = createtree(node->right, data);

}

return node;

}

void inorder(NODE *node)

{

if(node != NULL)

{

inorder(node->left); printf("%d\t", node->data); inorder(node->right);

}

}

void preorder(NODE *node)

{

if(node != NULL)

```
{
printf("%d\t", node->data); preorder(node->left); preorder(node->right);
}
}
void postorder(NODE *node)
{
if(node != NULL)
{
postorder(node->left); postorder(node->right); printf("%d\t", node->data);
}
}
int main()
{
int data, ch, i, n; NODE *root=NULL; while (ch!=5)
{
printf("\nbinary search tree ");
printf("\n 1.Insertion in Binary Search Tree "); printf("\n 2.Inorder ");
printf("\n 3.Preorder "); printf("\n 4.Postorder "); printf("\n 5.Exit \n"); printf("\n Enter your Choice ");
scanf("%d", &ch); switch (ch)
{
case 1 printf("\n Enter size of tree " );
scanf("%d", &n);
printf("\n Enter the elements of tree \n");
for(i=0; i<n; i++)
{
scanf("%d", &data); root=createtree(root, data);
}
break;
case 2 printf("\n Inorder Traversal \n"); inorder(root);
break;
case 3 printf("\n Preorder Traversal \n"); preorder(root);
break;
case 4 printf("\n Postorder Traversal \n"); postorder(root);
break; case 5 exit(0);
default printf("\n Enter valid choice \n"); break;
}
}
}
```

BINARY SEARCH TREE OPERATIONS

1.Insertion in Binary Search Tree 2.Inorder

3.Preorder 4.Postorder 5.Exit

Enter your Choice 1

Enter size of tree 3

Enter the elements of tree 99

98

97

binary search tree

1.Insertion in Binary Search Tree 2.Inorder

3.Preorder 4.Postorder 5.Exit

Enter your Choice 2

Inorder Traversal 97 98 99

binary search tree

1.Insertion in Binary Search Tree 2.Inorder

3.Preorder 4.Postorder 5.Exit

Enter your Choice 3

Preorder Traversal 99 98 97

binary search tree

1.Insertion in Binary Search Tree 2.Inorder

Preorder

1. Postorder 5.Exit

Enter your Choice 4

Postorder Traversal 97 98 99

binary search tree

1.Insertion in Binary Search Tree 2.Inorder

3.Preorder 4.Postorder 5.Exit

Enter your Choice

Experiment 6

```
#include<stdio.h> #include<stdlib.h> #include<conio.h>

int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20]; int delete();

void add(int item); void bfs(int s,int n); void dfs(int s,int n); void push(int item); int pop();

void main()

{

int n,i,s,ch,j; char c,dummy;

printf("Enter the number of Vertices"); scanf("%d",&n);

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

{

for(j=1;j<=n;j++)

{

printf("Enter 1 IF %d HAS A NODE WITH %d ELSE 0",i,j); scanf("%d",&a[i][j]);

}

}

printf("THE ADJACENCY MATRIX IS\n");

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

{

for(j=1;j<=n;j++)

{

printf(" %d",a[i][j]);

}

printf("\n");

}

do

{

for(i=1;i<=n;i++) vis[i]=0; printf("\n menu ");

printf("\n 1.B.F.S");

printf("\n 2.D.F.S"); printf("\n Enter your choice"); scanf("%d",&ch);

printf("Enter the Source Vertex"); scanf("%d",&s);

switch(ch)

{

case 1bfs(s,n); break;

case 2 dfs(s,n); break;

}

printf("DO U WANT TO CONTINUE (Y/N) ? ");

scanf("%c",&dummy);

scanf("%c",&c);
```

```
}
while((c=='y')||(c=='Y'));
}//main exit
void bfs(int s,int n)
{
int p,i; add(s);
vis[s]=1; p=delete(); if(p!=0) printf(" %d",p); while(p!=0)
{
for(i=1;i<=n;i++) if((a[p][i]!=0)&&(vis[i]==0))
{
add(i);
vis[i]=1;
}
p=delete(); if(p!=0)
printf(" %d ",p);
}
for(i=1;i<=n;i++) if(vis[i]==0)
bfs(i,n);
}
void add(int item)
{
if(rear==19) printf("QUEUE FULL"); else
{
if(rear==-1)
{
q[++rear]=item; front++;
}
else q[++rear]=item;
}
}
int delete()
{
int k; if((front>rear)||(front==-1)) return(0);
else
{
k=q[front++]; return(k);
}
}
void dfs(int s,int n)
```

```
{
int i,k; push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf(" %d ",k); while(k!=0)
{
for(i=1;i<=n;i++) if((a[k][i]!=0)&&(vis[i]==0))
{
push(i);
vis[i]=1;
}
k=pop();
if(k!=0)
printf(" %d ",k);
}
for(i=1;i<=n;i++) if(vis[i]==0)
dfs(i,n);
}
void push(int item)
{
if(top==19)
printf("Stack overflow "); else
stack[++top]=item;
}
int pop()
{
int k; if(top==-1) return(0); else
{
k=stack[top-1]; return(k);
}
}
#include<stdio.h> int no_vertices;
void printGraph(int adj[][no_vertices])
{
for(int i=0;i<no_vertices;i++)
{
for(int j =0;j<no_vertices;j++)
{
```

```
printf(" %d ",adj[i][j]);
}
printf("\n");
}
}
void dfs(int adj[][no_vertices],int visited[], int start)
{
printf("%d\t",start); visited[start] =1;
for(int i=0;i<no_vertices;i++)
{
if(visited[i]!=1 && adj[start][i]==1)
{
dfs(adj,visited,i);
}
}
}
void bfs(int adj[][no_vertices], int start)
{
int visited[no_vertices],queue[no_vertices],front=-1,rear=-1;
for(int i=0;i<no_vertices;i++) visited[i] =0;
front++;
queue[++rear] = start; visited[start] = 1;
while(front<=rear)
{
start = queue[front++]; printf("%d\t",start);
for(int i=0;i<no_vertices;i++)
{
if(adj[start][i]==1 && visited[i] != 1)
{
queue[++rear] = i; visited[i] =1;
}
}
}
}
int main()
{
int s,d,ch,start;
printf("\nEnter the number of vertices "); scanf("%d",&no_vertices);
int adj[no_vertices][no_vertices],visited[no_vertices];
```

```
for(int i =0;i<no_vertices;i++) for(int j=0;j<no_vertices;j++)
adj[i][j] =0;
while(s!=-1 && d!=-1)
{
printf("Enter an Edge fromnode(0 to %d) to node(0 to %d) ",no_vertices,no_vertices);
scanf("%d%d",&s,&d);
adj[s][d] = 1;
adj[d][s] = 1;
}
do
{
printf("\n 1. BFS \n 2.DFS \n 3.Print adjency Matrix \n 4.Exit "); scanf("%d",&ch);
switch(ch)
{
case 1
printf("Enter the Vertex fron which do you wanted to start "); scanf("%d",&start);
bfs(adj,start);break; case 2
printf("Enter the Vertex fron which do you wanted to start "); scanf("%d",&start);
for(int i =0;i<no_vertices;i++) visited[i] = 0;
dfs(adj,visited, start);break;
break;
case 3printGraph(adj);break; case 4 break;
}
}while(ch != 4);
return 0;
}
```

Enter the number of vertices 5

Enter an Edge fromnode(0 to 5) to node(0 to 5) 1 2

Enter an Edge fromnode(0 to 5) to node(0 to 5) 2 3

Enter an Edge fromnode(0 to 5) to node(0 to 5) 0 4

Enter an Edge fromnode(0 to 5) to node(0 to 5) 1 0

Enter an Edge fromnode(0 to 5) to node(0 to 5) 2 0

Enter an Edge fromnode(0 to 5) to node(0 to 5) -1 -1

1. BFS 2.DFS

3.Print adjency Matrix 4.Exit 1

Enter the Vertex fron which do you wanted to start 1 1 0 2 4 3

1. BFS 2.DFS

3.Print adjency Matrix 4.Exit 2

Enter the Vertex fron which do you wanted to start 0 0 1 2 3 4

1. BFS 2.DFS

3.Print adjency Matrix 4.Exit 4

Experiment 7

#include <stdio.h> #include <stdlib.h>

void push(); void pop(); void display(); struct node

{

int val;

struct node *next;

};

struct node *head;

void main ()

{

int choice=0;

printf("\n OPERATIONS\n"); while(choice != 4)

{

printf("\n menu\n");

printf("\n 1.Push\n 2.Pop\n 3.Show\n 4.Exit"); printf("\n Enter your choice \n"); scanf("%d",&choice);

switch(choice)

{

case 1

{

push(); break;

}

case 2

{

pop(); break;

}

case 3

{

display(); break;

}

case 4

{

```
printf("Exiting"); break;
}
default
{
printf("Please Enter valid choice ");
}
};
}
}
void push ()
{
int val;
struct node *ptr = (struct node*)malloc(sizeof(struct node)); if(ptr == NULL)
{
printf("Not able to push the element");
}
else
{
printf("Enter the value "); scanf("%d",&val); if(head==NULL)
{
ptr->val = val; ptr -> next = NULL; head=ptr;
}
else
{
ptr->val = val; ptr->next = head; head=ptr;
}
printf("Item pushed");
}
}
void pop()
{
int item;
struct node *ptr; if (head == NULL)
{
printf("Underflow");
}
else
{
item = head->val; ptr = head;
```

```
head = head->next;
free(ptr); printf("Item popped");
}
}
void display()
{
int i;
struct node *ptr; ptr=head;
if(ptr == NULL)
{
printf("Stack is empty\n");
}
else
{
printf("Printing Stack elements \n"); while(ptr!=NULL)
{
printf("%d\n",ptr->val); ptr = ptr->next;
}
}
}
```

STACK OPERATIONS USING LINKED LIST

menu

1.Push 2.Pop 3.Show 4.Exit

Enter your choice 1

Enter the value 12 Item pushed menu

1.Push 2.Pop 3.Show 4.Exit

Enter your choice 1

Enter the value 66 Item pushed menu

1.Push 2.Pop 3.Show 4.Exit

Enter your choice 3

Printing Stack elements 66

12

menu

1.Push 2.Pop 3.Show 4.Exit

Enter your choice 2

Item popped menu

1.Push 2.Pop 3.Show 4.Exit

Enter your choice 4

Exiting

```
#include<stdio.h> #include<stdlib.h>

struct node

{

int data;

struct node *next;

};

struct node *front; struct node *rear; void insert(); void delete(); void display(); void main ()

{

int choice; while(choice != 4)

{

printf("\n menu\n");

printf("\n 1.Insertmenu \n 2.Deletemenu \n 3.Displaymenu \n 4.Exitmenu\n");

printf("\nEnter your choice "); scanf("%d",& choice); switch(choice)

{

case 1 insert(); break; case 2 delete(); break; case 3 display(); break; case 4 exit(0);

break; default

printf("\n Enter valid choicemenu \n");

}

}

}

void insert()

{

struct node *ptr; int item;

ptr = (struct node *) malloc (sizeof(struct node)); if(ptr == NULL)

{

printf("\n Overflowmenu \n"); return;

}

else

{

printf("\n Enter value \n"); scanf("%d",&item);

ptr -> data = item; if(front == NULL)
```

```
{
front = ptr; rear = ptr;
front -> next = NULL; rear -> next = NULL;
}
else
{
rear -> next = ptr; rear = ptr;
rear->next = NULL;
}
}
}
void delete ()
{
struct node *ptr; if(front == NULL)
{
printf("\n Underflowmenu \n"); return;
}
else
{
ptr = front;
front = front -> next; free(ptr);
}
}
void display()
{
struct node *ptr; ptr = front; if(front == NULL)
{
printf("\n Empty queuemenu\n");
}
else
{ printf("\n Printing values menumenu. \n"); while(ptr != NULL)
{
printf("\n%d\n",ptr -> data); ptr = ptr -> next;
}
#include<stdio.h> #include<stdlib.h>
struct node
{
int data;
struct node *next;
```

```
};
struct node *front; struct node *rear; void insert(); void delete(); void display(); void main ()
{
int choice; while(choice != 4)
{
printf("\n menu\n");
printf("\n 1.Insert \n 2.Delete \n 3.Display \n 4.Exit\n"); printf("\nEnter your choice ");
scanf("%d",& choice); switch(choice)
{
case 1 insert();
      break;
      case 2
      delete();
      break;
      case 3
      display();
      break;
      case 4
      exit(0);
      break;
      default
      printf("\n Enter valid choicemenu \n");
   }
 }
}
void insert()
{
struct node *ptr; int item;
ptr = (struct node *) malloc (sizeof(struct node)); if(ptr == NULL)
{
printf("\n Overflow \n"); return;
}
else
{
printf("\n Enter value \n"); scanf("%d",&item);
ptr -> data = item; if(front == NULL)
{
front = ptr; rear = ptr;
front -> next = NULL; rear -> next = NULL;
}
else
{
rear -> next = ptr; rear = ptr;
```

```
rear->next = NULL;
}
}
}
void delete ()
{
struct node *ptr; if(front == NULL)
{
printf("\n Underflow \n"); return;
}
else
{
ptr = front;
front = front -> next; free(ptr);
}
}
void display()
{
struct node *ptr; ptr = front; if(front == NULL)
{
printf("\n Empty queue\n");
}
else
{ printf("\n Printing values . \n"); while(ptr != NULL)
{
printf("\n%d\n",ptr -> data); ptr = ptr -> next;
}
}
}
```

menu

1.Insert 2.Delete 3.Display 4.Exit

Enter your choice 1 Enter value

65

menu

1.Insert 2.Delete 3.Display 4.Exit

Enter your choice 1

Enter value 33

menu

1.Insert 2.Delete 3.Display 4.Exit

Enter your choice 1 Enter value

699

menu

1.Insert 2.Delete 3.Display 4.Exit

Enter your choice 3 Printing values . 65

33

699

menu

1.Insert 2.Delete 3.Display 4.Exit

Enter your choice 2 menu

1.Insert 2.Delete

3.Display 4.Exit

Enter your choice 3 Printing values . 33

699

menu

1.Insert 2.Delete 3.Display 4.Exit

Enter your choice 4

Experimeny 8

```
#include <stdio.h> #include <stdlib.h>

struct node

{

int co, exp; struct node* next;

};

struct node* create (struct node* head, int co, int exp)

{

struct node *temp, *flag;

if (head==NULL)

{

temp=(struct node* )malloc (sizeof(struct node)); temp->co = co;

temp->exp = exp; temp->next = NULL; head=temp;

}

else{

temp=head;

while (temp->next!=NULL) temp=temp->next;

flag = (struct node*)malloc(sizeof(struct node)); flag ->co = co;

flag ->exp = exp; flag ->next = NULL; temp ->next = flag;

}

return head;

}
```

```
struct node *polyAdd(struct node*p1, struct node *p2, struct node *sum)
{
struct node *poly1=p1, *poly2=p2, *res;
if(poly1!=NULL && poly2==NULL)
{
sum=poly1;
return sum;
}
else if(poly1==NULL && poly2!=NULL)
{
sum=poly2; return sum;
}
while(poly1 != NULL && poly2 !=NULL)
{
if(sum==NULL)
{
sum=(struct node*)malloc(sizeof(struct node)); res=sum;
}
else{
res->next=(struct node*)malloc(sizeof(struct node)); res=res->next;
}
if(poly1->exp > poly2->exp)
{
res->co=poly1->co; res->exp=poly1->exp; poly1=poly1->next;
}
else if(poly1->exp > poly2->exp)
{
res->co=poly2->co; res->exp=poly2->exp; poly2=poly2->next;
}
else if(poly1->exp==poly2->exp)
{
res->co=poly1->co + poly2->co; res->exp=poly1->exp; poly1=poly1->next; poly2=poly2->next;
}
}
while(poly1!=NULL)
{
res->next=(struct node*)malloc(sizeof(struct node)); res=res->next;
res->co=poly1->co; res->exp=poly1->exp; poly1=poly1->next;
}
```

```
while(poly2!=NULL)
{
res->next=(struct node*)malloc(sizeof(struct node)); res=res->next;
res->co=poly2->co; res->exp=poly2->exp; poly2=poly2->next;
}
res->next = NULL; return sum;
}
void display (struct node* head)
{
struct node *temp=head; while(temp!=NULL)
{
printf("%d^%d+", temp->co, temp-> exp); temp=temp->next;
}
printf("\n");
}
void main()
{
struct node*p1=NULL, *p2=NULL, *sum=NULL; int ch, co, exp;
int loop=1; while(loop)
{
printf("\n 1.Enter Polynoimal 1 "); printf("\n 2.Enter Polynomial 2 "); printf("\n 3. Addition");
printf("\n 4.Exit");
scanf("%d",&ch);
switch(ch)
{
case 1 printf("Enter coefficient \n"); scanf("%d", &co); printf("Enter exponent\n"); scanf("%d", &exp); p1=create(p1,co,exp);
break;
case 2 printf("Enter coefficient \n"); scanf("%d", &co); printf("Enter exponent\n"); scanf("%d", &exp); p2=create(p2,co,exp);
break;
case 3 sum=polyAdd(p1,p2,sum); printf("\nPolynomial 1\n"); display(p1); printf("\nPolynomial 2\n"); display(p2);
printf("\nSum \n"); display(sum); break;
case 4 loop=0;
break;
default printf("Wrong choice."); break;
}
}
}
```

1. Enter Polynoimal 1
2. Enter Polynomial 2
3. Addition 4.Exit1

Enter coefficient 4

Enter exponent 54

1. Enter Polynoimal 1
2. Enter Polynomial 2
3. Addition 4.Exit1

Enter coefficient 56

Enter exponent 3

1. Enter Polynoimal 1
2. Enter Polynomial 2
3. Addition 4.Exit2

Enter coefficient 66

Enter exponent 1

1. Enter Polynoimal 1
2. Enter Polynomial 2
3. Addition 4.Exit3

Polynomial 1 4^54+56^3+

Polynomial 2 66^1+

Sum 4^54+56^3+66^1+

1. Enter Polynoimal 1
2. Enter Polynomial 2
3. Addition

Experiment 10

#include<stdio.h> #include<stdlib.h> struct node

{

int data;

struct node *next;

};

struct node *head;

void beginsert (); void lastinsert (); void randominsert(); void begin_delete(); void last_delete(); void random_delete(); void display();

void search(); void main ()

{

int choice =0;

while(choice != 7)

{

printf("\n ,menu* \n");

printf("\n 1.Insert in begining\n 2.Insert at last\n 3.Delete from Beginning\n 4.Delete from last\n 5.Search for an element\n 6.Show\n 7.Exit\n");

printf("\nEnter your choice?\n"); scanf("\n%d",&choice); switch(choice)

{

case 1: beginsert(); break;

case 2: lastinsert(); break;

case 3: begin_delete(); break;

case 4: last_delete(); break;

```
case 5: search(); break; case 6: display();
break; case 7:
exit(0); break; default:
printf("Please enter valid choice");
}
}
}
void beginsert()
{
struct node *ptr,*temp; int item;
ptr = (struct node *)malloc(sizeof(struct node)); if(ptr == NULL)
{
printf("\n Overflow");
}
else
{
printf("\n Enter the node data: "); scanf("%d",&item);
ptr -> data = item; if(head == NULL)
{
head = ptr;
ptr -> next = head;
}
else
{
temp = head;
while(temp->next != head) temp = temp->next;
ptr->next = head; temp -> next = ptr; head = ptr;
}
printf("\n Node inserted\n");
}
}
void lastinsert()
{
struct node *ptr,*temp; int item;
ptr = (struct node *)malloc(sizeof(struct node)); if(ptr == NULL)
{
printf("\n Overflow\n");
}
else
```

```
{
printf("\n Enter Data: "); scanf("%d",&item);
ptr->data = item; if(head == NULL)
{
head = ptr;
ptr -> next = head;
}
else
{
temp = head;
while(temp -> next != head)
{
temp = temp -> next;
}
temp -> next = ptr; ptr -> next = head;
}
printf("\n Node inserted\n");
}
}
void begin_delete()
{
struct node *ptr; if(head == NULL)
{
printf("\n Underflow");
}
else if(head->next == head)
{
head = NULL; free(head);
printf("\n Node deleted\n");
}
else
{ ptr = head;
while(ptr -> next != head) ptr = ptr -> next;
ptr->next = head->next; free(head);
head = ptr->next; printf("\n Node deleted\n");
}
}
void last_delete()
{
```

```
struct node *ptr, *preptr; if(head==NULL)
{
printf("\n Underflow");
}
else if (head ->next == head)
{
head = NULL; free(head);
printf("\n Node deleted\n");
}
else
{
ptr = head;
while(ptr ->next != head)
{
preptr=ptr;
ptr = ptr->next;
}
preptr->next = ptr -> next; free(ptr);
printf("\n Node deleted\n");
}
}
void search()
{
struct node *ptr; int item,i=0,flag=1; ptr = head;
if(ptr == NULL)
{
printf("\n Empty List\n");
}
else
{
printf("\n Enter item which you want to search: \n"); scanf("%d",&item);
if(head ->data == item)
{
printf("Item found at location: %d",i+1); flag=0;
}
else
{
while (ptr->next != head)
{
```

```
if(ptr->data == item)
{
printf("item found at location: %d ",i+1); flag=0;
break;
}
else
{
flag=1;
}
i++;
ptr = ptr -> next;
}
}
if(flag != 0)
{
printf("Item not found\n");
}
}
}
void display()
{
struct node *ptr; ptr=head; if(head == NULL)
{
printf("\n Nothing to print");
}
else
{
printf("\n Values . \n");
while(ptr -> next != head)
{
printf("%d\n", ptr -> data); ptr = ptr -> next;
}
printf("%d\n", ptr -> data);
}
}
```

,menu*

1.Insert in begining 2.Insert at last

3.Delete from Beginning 4.Delete from last 5.Search for an element

6.Show

7.Exit

Enter your choice? 1

Enter the node data: 43

Node inserted

,menu*

1.Insert in begining 2.Insert at last

3.Delete from Beginning 4.Delete from last 5.Search for an element 6.Show

7.Exit

Enter your choice? 2

Enter Data: 66

Node inserted

,menu*

1.Insert in begining 2.Insert at last

3.Delete from Beginning 4.Delete from last 5.Search for an element 6.Show

7.Exit

Enter your choice? 5

Enter item which you want to search: 2

Item not found

,menu*

1.Insert in begining 2.Insert at last

3.Delete from Beginning 4.Delete from last 5.Search for an element

6.Show 7.Exit

Enter your choice? 5

Enter item which you want to search: 32

Item not found

,menu*

1.Insert in begining 2.Insert at last

3.Delete from Beginning 4.Delete from last 5.Search for an element 6.Show

7.Exit

Enter your choice? 3

Node deleted

,menu*

1.Insert in begining 2.Insert at last

3.Delete from Beginning 4.Delete from last

5.Search for an element 6.Show

7.Exit

Enter your choice? 4

Node deleted

,menu*

1.Insert in begining 2.Insert at last

3.Delete from Beginning 4.Delete from last 5.Search for an element 6.Show

7.Exit

Enter your choice? 6

Nothing to print

,menu*

1.Insert in begining 2.Insert at last

3.Delete from Beginning 4.Delete from last 5.Search for an element 6.Show

7.Exit

Enter your choice? 7

Experiment 9