Exp. No: Date:

LINEAR SEARCH

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[50],i,n,x,flag=0;
clrscr();
printf("Enter the value of n:");
scanf("%d",&n);
printf("Enter the numbers:");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
printf("Enter the number to be searched:");
scanf("%d",&x);
for(i=1;i<=n;i++)
{
if(x==a[i])
{
printf("The number %d is found in %d position",x,i);
flag=1;
break;
}
}
if(flag==0)
printf("The number is not found");
getch();
}
```

Exp. No: Date:

OUTPUT:

```
Enter the value of n:4
Enter the numbers:22
24
14
25
Enter the number to be searched:24
The number 24 is found in 2 position
```

```
Enter the number to be searched:46
The number is not found
```

Exp. No: Date:

BINARY SEARCH

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],m,n,i,j,temp;
void binary(int a[100],int m,int n);
clrscr();
printf("Enter the size of array");
scanf("%d",&m);
printf("Enter array element");
for(i=1;i<=m;i++)
{
scanf("%d",&a[i]);
}
for(i=1;i<=m;i++)
{
for(j=i+1;j<=m;j++)
{
if(a[i]>a[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
}
printf("Sorted array");
for(i=1;i<=m;i++)
{
printf("\n%d",a[i]);
}
printf("\nEnter the search element");
scanf("%d",&n);
binary(a,m,n);
getch();
}
void binary(int a[100],int m,int n)
{
int low=1,high=m,mid,flag=0;
while(low<=high&&flag==0)
```

```
{
mid=(low+high)/2;
if(n==a[mid])
{
flag=1;
printf("The number %d is found in:%d",n,mid);
}
else if(n>a[mid])
{
low=mid+1;
}
else
{
high=mid-1;
}
}
if(flag==0)
{
printf("The number is not found");
}
}
```

Exp. No: Date:

OUTPUT:

```
Enter the size of array4
Enter array element21
12
23
14
Sorted array
12
14
21
23
Enter the search element23
The number 23 is found in:4

Enter the number to be searched:25
The number is not found
```

Exp. No: Date:

FIBONACCI SEARCH

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n,a[50],i,key,loc,p,q,r,m,fk;
  clrscr();
  printf("\nenter the value of n");
  scanf("%d",&n);
  printf("enter the number");
  for(i=1;i<=n;i++)
   scanf("%d",&a[i]);
  printf("enter the number to be searched");
  scanf("%d",&key);
  fk=fib(n+1);
  p=fib(fk);
  q=fib(p);
  r=fib(q) ;
  m=(n+1)-(p+q);
  if(key>a[p])
   p=p+m;
  loc=rfibsearch(a,n,p,q,r,key);
  if(loc==0)
   printf("key is not found");
  else
   printf("The number is found in %d position",loc);
  getch();
}
int fib(int m)
{
  int a,b,c;
  a=0;
  b=1;
  c=a+b;
  while(c<m)
  {
   a=b;
   b=c;
   c=a+b;
  }
  return b;
}
int rfibsearch(int a[],int n,int p,int q,int r,int key)
```

```
{
  int t;
  if(p<1||p>n)
   return 0;
  else if(key==a[p])
   return p;
  else if(key<a[p])
  {
   if(r==0)
    return 0;
   else
   {
    p=p-r;
    t=q;
    q=r;
    r=t-r;
    return rfibsearch(a,n,p,q,r,key);
   }
  }
  else
  {
   if(q==1)
     return 0;
   else
   {
    p=p+r;
    q=q-r;
    r=r-q;
    return rfibsearch(a,n,p,q,r,key);
   }
  }
}
```

Exp. No: Date:

OUTPUT:

```
enter the value of n4
enter the number22
25
24
14
enter the number to be searched25
The number is found in 2 position
```

```
Enter the number to be searched:46
The number is not found
```

Exp. No: Date:

INSERTION SORT

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
int n;
void main()
{
int i,a[10];
void insertion(int a[10]);

clrscr();
printf("Enter the number of terms:");
scanf("%d",&n);
printf("Enter the number:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
insertion(a);
getch();
}
void insertion(int a[10])
{
int i,j,temp;
for(i=0;i<=n-1;i++)
{
temp=a[i];
j=i-1;
while ((j>=0)&&(a[j]>temp))
{
a[j+1]=a[j];
j=j-1;
}
a[j+1]=temp;
}
printf("The sorted elements are:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
}
```

Exp. No: Date:

OUTPUT:

```
Enter the number of terms:4
Enter the number:24
14
25
22
The sorted elements are:
14
22
24
25
```

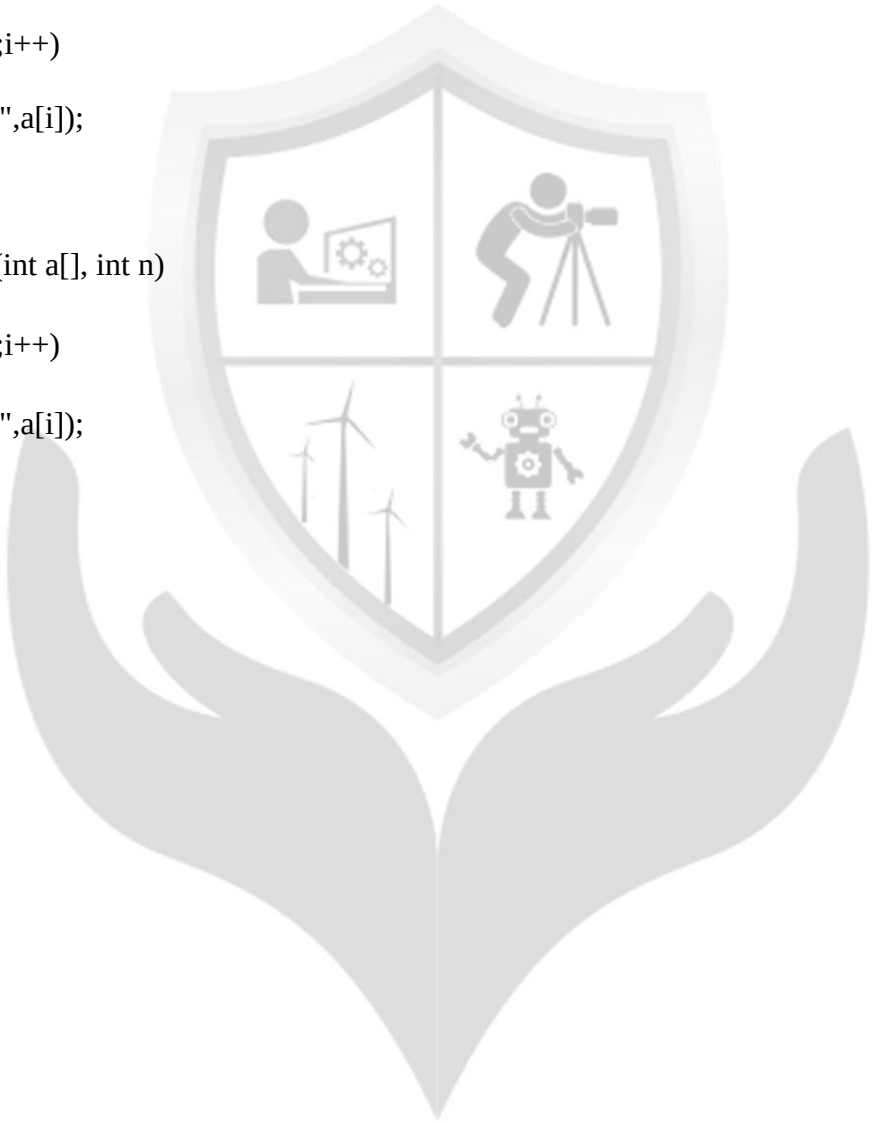
Exp. No: Date:

SELECTION SORT

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
int i,j,t,n,k,a[20],min;
void get();
void selsort();
void display(int[],int);

void main()
{
clrscr();
get();
selsort();
getch();
}
void get()
{
printf("\nEnter the number of elements:");
scanf("%d",&n);
printf("\nEnter the elements:\n");
for(i=1;i<=n;i++)
{
scanf("%d",&a[i]);
}
printf("\nThe unsorted elements are:\n");
display(a,n);
}
void selsort()
{
for(k=1;k<n;k++)
{
min=k;
for(j=k+1;j<=n;j++)
{
if(a[j]<a[min])
{
min=j;
}
}
if(min!=k)
{
t=a[k];
a[k]=a[min];
```

```
a[min]=t;
}
printf("\n PASS: %d", k);
display(a,n);
}
printf("\n\nSorted elements are:\n");
display(a,n);
}
void display(int a[], int n)
{
for(i=1;i<=n;i++)
{
printf("\t\%d",a[i]);
}

}
void display(int a[], int n)
{
for(i=1;i<=n;i++)
{
printf("\t\%d",a[i]);
}
}
```



Exp. No: Date:

OUTPUT:

```
Enter the number of elements:5

Enter the elements:
12
32
1
3
45

The unsorted elements are:
        12      32      1       3       45
 PASS: 1        1       32      12      3       45
 PASS: 2        1       3       12      32      45
 PASS: 3        1       3       12      32      45
 PASS: 4        1       3       12      32      45

Sorted elements are:
        1       3       12      32      45
```

Exp. No: Date:

BUBBLE SORT

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
int n;
void main()
{
int i,a[10];
void bubble(int a[10]);
clrscr();
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
bubble(a);
getch();
}
void bubble(int a[10])
{
int i,j,temp;
for(i=0;i<=n-2;i++)
{
for(j=0;j<=n-2-i;j++)
{
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
printf("The sorted array:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
}
```

Exp. No: Date:

OUTPUT:

```
Enter the number of elements:4
Enter the elements:43
21
23
12
The sorted array:
12
21
23
43
```

Exp. No: Date:

SHELL SORT

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
int n;
void shell(int a[10],int n);
void main()
{
int i,a[10];
clrscr();
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
shell(a,n);
printf("\n The sorted elements are:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
getch();
}
void shell(int a[10],int n)
{
int k,i,j,temp;
for(k=n/2;k>0;k=k/2)
for(i=k;i<n;i++)
{
temp=a[i];
for(j=i;(j>=k)&&(a[j-k]>temp);j=j-k)
{
a[j]=a[j-k];
}
a[j]=temp;
}
}
```

Exp. No: Date:

OUTPUT:

```
Enter the number of elements:5
Enter the elements:21
34
75
12
32

 The sorted elements are:
12
21
32
34
75
```

Exp. No: Date:

QUICK SORT

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int n;
void quick(int a[10],int low,int high);
int partition(int a[10],int low,int high);
void swap(int a[10],int*low,int*high);
void main()
{
int i,a[10];
clrscr();
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
quick(a,0,n-1);
printf("\n The sorted elements are:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
getch();

}
void quick(int a[10],int low,int high)
{
int m,i;
if(low<high)
{
m=partition(a,low,high);
quick(a,low,m-1);
quick(a,m+1,high);
}
}
int partition(int a[10],int low,int high)
{
int pivot=a[low],i=low,j=high;
while(i<=j)
{
while(a[i]<=pivot)
i++;
while(a[j]>pivot)
{
j--;
```

```
}
if(i<=j)
swap(a,&i,&j);
}
swap(a,&low,&j);
return j;
}
void swap(int a[10],int*i,int*j)
{
int temp;
temp=a[*i];
a[*i]=a[*j];
a[*j]=temp;
}
```

Exp. No: Date:

OUTPUT :

```
Enter the number of elements:4
Enter the elements:34
12
56
1

 The sorted elements are:
1
12
34
56
```

Exp. No: Date:

HEAP SORT

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
void heapsort(int[],int);
void heapify(int[],int);
void adjust(int[],int);
void main()
{
int a[20],i,n=1;
clrscr();
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0,n=1;i<n;i++,n++)
scanf("%d",&a[i]);
heapsort(a,n);
printf("\nThe sorted elements are:");
for(i=0,n=1;i<n;i++,n++)
printf("\n%d",a[i]);
getch();
}
void heapsort(int a[],int n)
{
int i,t;
heapify(a,n);
for(i=n-1;i>0;i--)
{
t=a[0];
a[0]=a[i];
a[i]=t;
adjust(a,i);
}
}
void heapify(int a[],int n)
{
int i;
for(i=n/2;i>1;i--)
{
adjust(a,n);
}
}
void adjust(int a[],int n)
{
int i,j,item;
j=0;
item=a[j];
```

```
i=2*j+1;
while(i<=n-1)
{
if((i+1<=n-1)&&(a[i]<a[i+1]))
{
i++;
}
if(item<a[i])
{
a[j]=a[i];
j=1;
i=2*j+1;
}
else
{
break;
}
}
a[j]=item;
}
```

Exp. No: Date:

OUTPUT:

```
Enter the number of elements:5
Enter the elements:56
78
21
12
34

The sorted elements are:
12
21
34
56
78
```

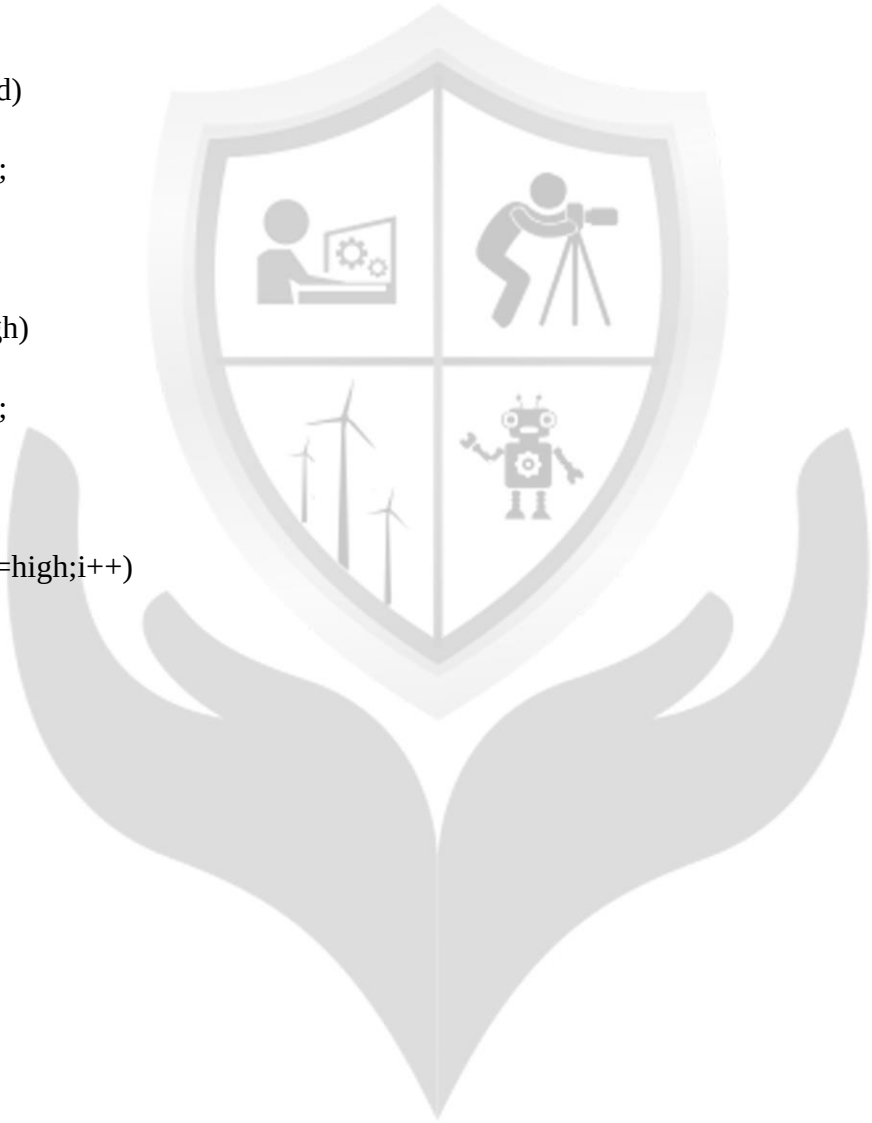
Exp. No: Date:

MERGE SORT

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int n;
void merge(int a[30],int low,int high);
int combine(int a[30],int low,int mid,int high);
void main()
{
int i,a[30];
clrscr();
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
merge(a,0,n-1);
printf("The sorted elements are:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
getch();
}
void merge(int a[30],int low,int high)
{
int mid;
if(low<high)
{
mid=(low+high)/2;
merge(a,low,mid);
merge(a,mid+1,high);
combine(a,low,mid,high);
}
}
int combine(int a[30],int low,int mid,int high)
{
int k=low,i=low,j=mid+1;
int temp[30];
while((i<=mid) && (j<=high))
{
if(a[i]<=a[j])
{
```

```
temp[k]=a[i];
i++;
k++;
}
else
{
temp[k]=a[j];
j++;
k++;
}
}
while(i<=mid)
{
temp[k]=a[i];
i++;
k++;
}
while(j<=high)
{
temp[k]=a[j];
j++;
k++;
}
for(i=low;i<=high;i++)
a[i]=temp[i];
}
```



Exp. No: Date:

OUTPUT:

```
Enter the number of elements:4
Enter the elements:23
42
43
1
The sorted elements are:
1
23
42
43_
```

Exp. No: Date:

RADIX SORT

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
int min=0,count=0,array[100]={0},array1[100]={0};
void main()
{
int k,i,j,temp,t,n;
printf("Enter the number of elements:");
scanf("%d",&count);
printf("Enter the elements:");
for(i=0;i<count;i++)
{
scanf("%d",&array[i]);
array1[i]=array[i];
}
for(k=0;k<3;k++)
{
for(i=0;i<count;i++)
{
min=array[i]%10;
t=i;
for(j=i+1;j<count;j++)
{
if(min>(array[j]%10))
{
min=array[j]%10;
t=j;
}
}
temp=array1[t];
array1[t]=array1[i];
array1[i]=temp;
temp=array[t];
array[t]=array[i];
array[i]=temp;
}
for(j=0;j<count;j++)
array[j]=array[j]/10;
}
printf("\n The sorted elements are\n:");
for(i=0;i<count;i++)
printf("\n%d",array1[i]);
getch();
}
```

Exp. No: Date:

OUTPUT :

```
Enter the number of elements:5
Enter the elements:23
65
45
10
40

 The sorted elements are
:
10
23
40
45
65
```

Exp. No: Date:

SPARSE MATRIX AND ITS TRANSPOSE

SOURCE CODE:

```
#include<stdio.h>
#define MAX 20
void printsparse(int[][3]);
void readsparse(int[][3]);
void transpose(int[][3],int[][3]);
int main()
{
int b1[MAX][3],b2[MAX][3],m,n;
printf("Enter the size of matrix (rows,columns):");
scanf("%d%d",&m,&n);
b1[0][0]=m;
b1[0][1]=n;
readsparse(b1);
transpose(b1,b2);
printsparse(b2);
}
void readsparse(int b[MAX][3])
{
int i,t;
printf("\nEnter no. of non-zero elements:");
scanf("%d",&t);
b[0][2]=t;
for(i=1;i<=t;i++)
{
printf("\nEnter the next triple(row,column,value):");
scanf("%d%d%d",&b[i][0],&b[i][1],&b[i][2]);
}
}
void printsparse(int b[MAX][3])
{
int i,n;
n=b[0][2]; //no of 3-triples
printf("\nAfter Transpose:\n");
printf("\nrow\t\tcolumn\t\tvalue\n");
for(i=0;i<=n;i++)
printf("%d\t\t%d\t\t%d\n",b[i][0],b[i][1],b[i][2]);
}
void transpose(int b1[][3],int b2[][3])
{
int i,j,k,n;
b2[0][0]=b1[0][1];
```

```
b2[0][1]=b1[0][0];
b2[0][2]=b1[0][2];
k=1;
n=b1[0][2];
for(i=0;i<b1[0][1];i++)
for(j=1;j<=n;j++)
//if a column number of current triple==i then insert the current triple in b2
if(i==b1[j][1])
{
b2[k][0]=i;
b2[k][1]=b1[j][0];
b2[k][2]=b1[j][2];
k++;
}
}
```

Exp. No: **Date:**

OUTPUT:

```
Enter the size of matrix (rows,columns):3
4

Enter no. of non-zero elements:4

Enter the next triple(row,column,value):1
0
5

Enter the next triple(row,column,value):1
2
3

Enter the next triple(row,column,value):2
1
1

Enter the next triple(row,column,value):2
3
2
```

```
After Transpose:

row             column          value
4               3               4
0               1               5
1               2               1
2               1               3
3               2               2
```

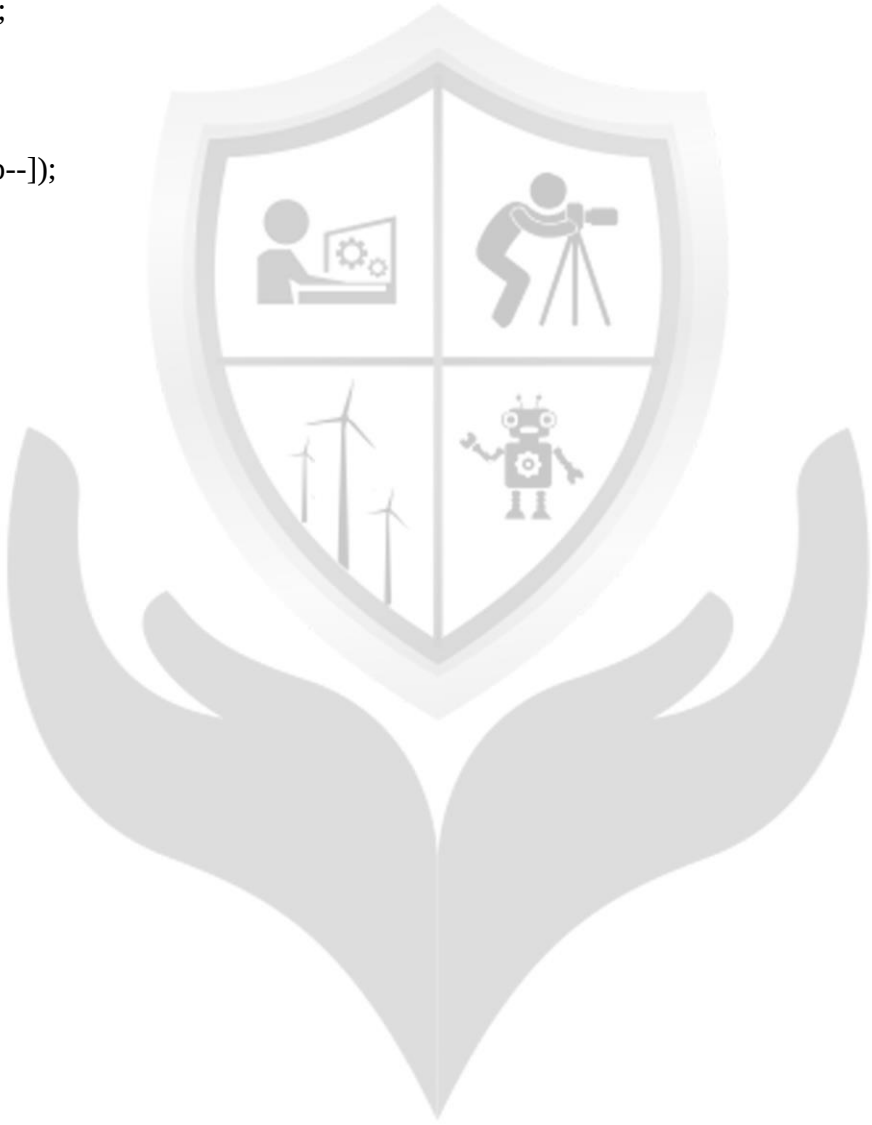
Exp. No: Date:

ARITHMETIC TO POSTFIX EXPRESSION

SOURCE CODE:

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
static char str[20];
int top=-1;
main()
{
char in[20],post[20],ch;
int i,j,l;
clrscr();
printf("enter the string");
gets(in);
l=strlen(in);
for(i=0,j=0;i<l;i++)
if(isalpha(in[i]))
post[j++]=in[i];
else
{
if(in[i]=='(')
push(in[i]);
else if(in[i]==')')
while((ch=pop())!='(')
post[j++]=ch;
else
{
while(priority(in[i])<=priority(str[top]))
post[j++]=pop();
push(in[i]);
}
}
while(top!=-1)
post[j++]=pop();
post[j]='\0';
printf("\n equivalent infix to postfix is:%s",post);
getch();
}
priority (char c)
{
switch(c)
{
case'+':
case'-': return 1;
```

```
case'*':
case'/':
return 2;
case'$':  return 3;
}
return 0;
}
push(char c)
{
str[++top]=c;
}
pop()
{
return(str[top--]);
}
```



Exp. No: **Date:**

OUTPUT:

```
enter the string(a+b)-(c-d)*e/f

 equivalent infix to postfix is:ab+cd-e*f/-
```

Exp. No: Date:

QUEUE USING ARRAY

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#define size 5
int ch=0,front=-1,rear=-1,q[size];
void insert();
void del();
void display();
void main()
{
clrscr();
while(1)
{
printf("\n1.INSERTION");
printf("\n2.DELETION");
printf("\n3.DISPLAY");
printf("\n4.EXIT");
printf("\nENTER THE CHOICE...");
scanf("%d",&ch);
switch(ch)
{
case 1:
insert();
break;
case 2:
del();
break;
case 3:
display();
break;
case 4:
exit(0);
break;
default:
printf("\nTHE INVALID CHOICE");
}
}
getch();
}
void insert()
{
int data;
```

```
if(rear==(size-1))
{
printf("\nQUEUE IS OVERFLOW");
}
else
{
if(front==-1)
{
front=0;
}
printf("\nENTER THE QUEUE ELEMENT...");
scanf("%d",&data);
rear++;
q[rear]=data;
}
}
void del()
{
if(front>rear)
{
printf("\nTHE QUEUE IS UNDERFLOW");
}
else
{
printf("\nTHE ELEMENT %d IS DELETED",q[front]);
front++;
}
}
void display()
{
if(front>rear)
{
printf("\nTHE QUEUE IS UNDERFLOW");
}
else
{
int i;
for(i=front;i<=rear;i++)
{
printf("\nTHE QUEUE ELEMENT IS ...%d",q[i]);
}
}
}
```

Exp. No: Date:

OUTPUT:

```
1.INSERTION
2.DELETION
3.DISPLAY
4.EXIT
ENTER THE CHOICE...1

ENTER THE QUEUE ELEMENT...12

1.INSERTION
2.DELETION
3.DISPLAY
4.EXIT
ENTER THE CHOICE...2

THE ELEMENT 12 IS DELETED
1.INSERTION
2.DELETION
3.DISPLAY
4.EXIT
ENTER THE CHOICE...1

ENTER THE QUEUE ELEMENT...35

1.INSERTION
2.DELETION
3.DISPLAY
4.EXIT
ENTER THE CHOICE...3

THE QUEUE ELEMENT IS ...35
1.INSERTION
2.DELETION
3.DISPLAY
4.EXIT
ENTER THE CHOICE..._
```

Exp. No: Date:

CIRCULAR QUEUE

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#define size 5
int ch=0,front=-1,rear=-1,q[size];
void insert();
void del();
void display();
void main()
{
clrscr();
while(1)
{
printf("\n1.INSERTION");
printf("\n2.DELETION");
printf("\n3.DISPLAY");
printf("\n4.EXIT");
printf("\nENTER THE CHOICE...");
scanf("%d",&ch);
switch(ch)
{
case 1:
insert();
break;
case 2:
del();
break;
case 3:
display();
break;
case 4:
exit(0);
break;
default:
printf("\nTHE INVALID CHOICE");
}
}
getch();
}
void insert()
{
int data;
```

```
if(((front==0)&&(rear==size-1))||(front==(rear+1)))
{
printf("\nQUEUE IS OVERFLOW");
}
if(front==-1)
{
front=0;
rear=0;
}
else
{
if(rear==size-1)
{
rear=0;
}
else
{
rear++;
}
}
printf("\nENTER THE QUEUE ELEMENT...");
scanf("%d",&data);
q[rear]=data;
}
void del()
{
if(front==-1)
{
printf("\nQUEUE UNDERFLOW");

}
printf("\nTHE QUEUE ELEMENT %d DELETED",q[front]);
q[front]=0;
if(front==rear)
{
front=-1;
rear=-1;
}
else
{
if(front==size-1)
{
front=0;
}
else
{
```

```
front++;
}
}
}
void display()

{
int fp=front,rp=rear;
if(fp==-1)
{
printf("\nQUEUE UNDERFLOW");

}
if(fp<=rp)
{
while(fp<=rp)
{
printf("\nTHE QUEUE ELEMENT ARE...%d",q[fp]);
fp++;
}
}
else
{
while(fp<=(size-1))
{
printf("\nTHE QUEUE ELEMENT ARE...%d",q[fp]);
fp++;
}
fp=0;
while(fp<=rp)
{
printf("\nTHE QUEUE ELEMENT IS...%d",q[fp]);
fp++;
}
}
}
```

Exp. No: Date:

OUTPUT:

```
1.INSERTION
2.DELETION
3.DISPLAY
4.EXIT
ENTER THE CHOICE...1

ENTER THE QUEUE ELEMENT...12

1.INSERTION
2.DELETION
3.DISPLAY
4.EXIT
ENTER THE CHOICE...2


THE ELEMENT 12 IS DELETED
1.INSERTION
2.DELETION
3.DISPLAY
4.EXIT
ENTER THE CHOICE...1

ENTER THE QUEUE ELEMENT...35

1.INSERTION
2.DELETION
3.DISPLAY
4.EXIT
ENTER THE CHOICE...3

THE QUEUE ELEMENT IS ...35
1.INSERTION
2.DELETION
3.DISPLAY
4.EXIT
ENTER THE CHOICE..._
```

Exp. No: Date:

PRIORITY QUEUE

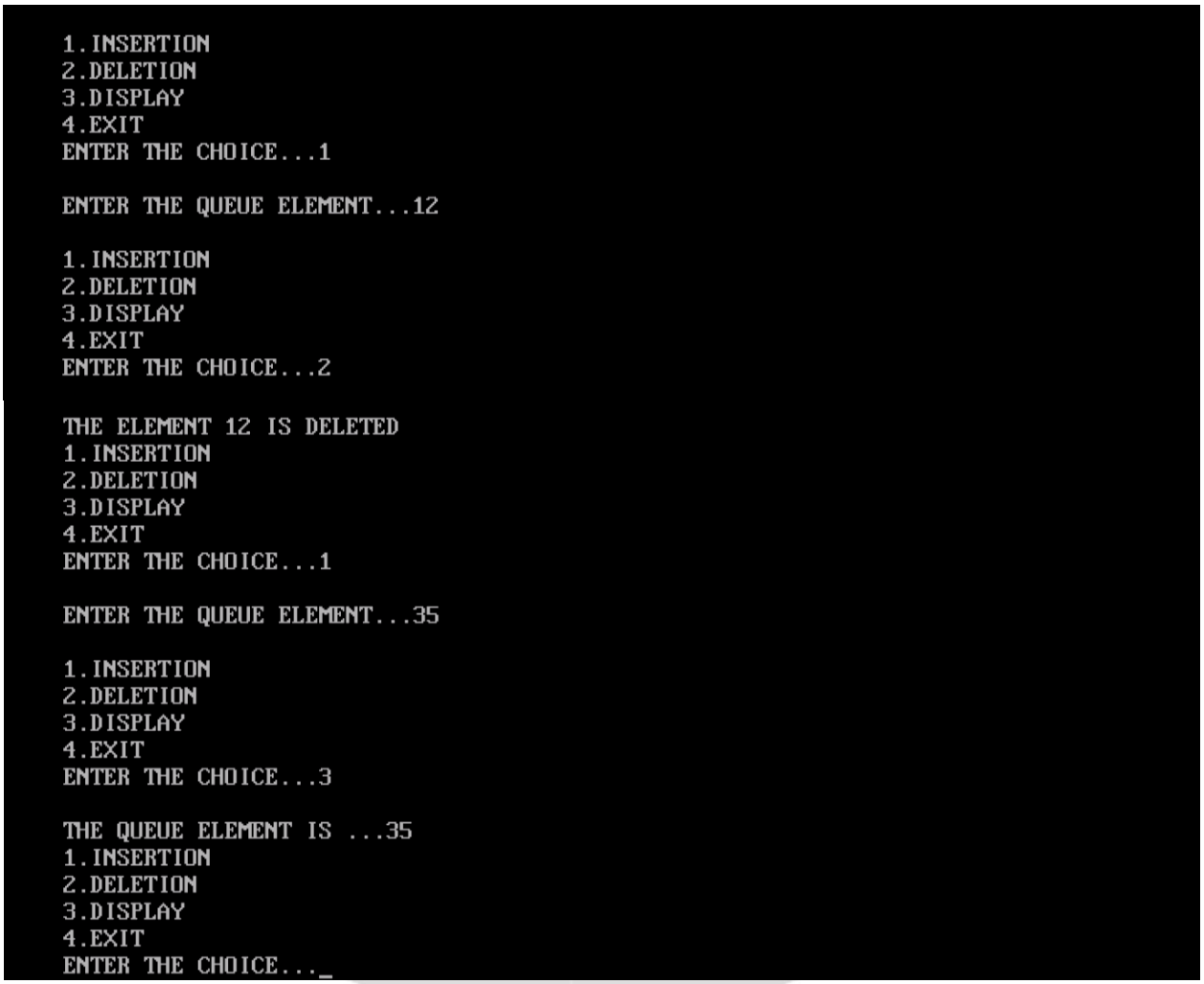
SOURCE CODE:

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
void enqueue();
void dequeue();
void display();
struct node
{
int info;
struct node *link;
}*front,*rear;
void main()
{
int n;
clrscr();
while(1)
{
printf("\n1.INSERTION");
printf("\n2.DELETION");
printf("\n3.DISPLAY");
printf("\n4.EXIT");
printf("\nENTER YOUR CHOICE...");
scanf("%d",&n);
switch(n)
{
case 1:
enqueue();
break;
case 2:
dequeue();
break;
case 3:
display();
break;
case 4:
exit(0);
break;
default:
printf("\nINVALID CHOICE");
}
getch();
```

```
}}
void enqueue()
{
struct node *tmp;
int added_item;
tmp=(struct node *)malloc(sizeof(struct node));
printf("\nENTER THE QUEUE ELEMENT...");
scanf("%d",&added_item);
tmp->info=added_item;
tmp->link=NULL;
if(front==NULL)
{
front=tmp;
}
else
{
rear->link=tmp;
}
rear=tmp;
}
void dequeue()
{
struct node *tmp;
if(front==NULL)
{
printf("\nTHE QUEUE IS EMPTY");
}
else
{
tmp=front;
printf("\nTHE ELEMENT %d IS DELETED",tmp->info);
front=front->link;
free(tmp);
}}
void display()
{
struct node *ptr;
ptr=front;
if(front==NULL)
{
printf("\nQUEUE IS EMPTY");
}
else
{
while(ptr!=NULL)
{
```

```
printf("\nTHE QUEUE ELEMENT IS...%d",ptr->info);
ptr=ptr->link;
}}}
```

OUTPUT:



Exp. No: Date:

DEQUEUE

SOURCE CODE:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define size 5

int main()
{
   int arr[size],R=-1,F=0,te=0,ch,n,i,x;

   for(;;)                    // An infinite loop
   {
      system("cls");              // for clearing the screen
      printf("F=%d  R=%d\n\n",F,R);
      printf("1. Add Rear\n");
      printf("2. Delete Rear\n");
      printf("3. Add Front\n");
      printf("4. Delete Front\n");
      printf("5. Display\n");
      printf("6. Exit\n");
      printf("Enter Choice: ");
      scanf("%d",&ch);

      switch(ch)
      {
         case 1:
            if(te==size)
            {
               printf("Queue is full");
               getch();    // pause the loop to see the message
            }
            else
            {
               printf("Enter a number ");
               scanf("%d",&n);
               R=(R+1)%size;
               arr[R]=n;
               te=te+1;
            }
            break;

         case 2:
            if(te==0)
```

```
    {
       printf("Queue is empty");
       getch();    // pause the loop to see the message
    }
    else
    {
       if(R==-1)
       {
          R=size-1;
       }
       printf("Number Deleted From Rear End = %d",arr[R]);
       R=R-1;
       te=te-1;
       getch();    // pause the loop to see the number
    }
    break;

case 3:
    if(te==size)
    {
       printf("Queue is full");
       getch();    // pause the loop to see the message
    }
    else
    {
       printf("Enter a number ");
       scanf("%d",&n);
       if(F==0)
       {
          F=size-1;
       }
       else
       {
          F=F-1;
       }
       arr[F]=n;
       te=te+1;
    }
    break;

case 4:
    if(te==0)
    {
       printf("Queue is empty");
       getch();    // pause the loop to see the message
    }
```

```
            else
            {
               printf("Number Deleted From Front End = %d",arr[F]);
               F=(F+1)%size;
               te=te-1;
               getch();    // pause the loop to see the number
            }
            break;

         case 5:
            if(te==0)
            {
               printf("Queue is empty");
               getch();    // pause the loop to see the message
            }
            else
            {
               x=F;
               for(i=1; i<=te; i++)
               {
                  printf("%d ",arr[x]);
                  x=(x+1)%size;
               }
               getch();    // pause the loop to see the numbers
            }
            break;

         case 6:
            exit(0);
            break;

         default:
            printf("Wrong Choice");
            getch();       // pause the loop to see the message
      }
   }
   return 0;
}
```

Exp. No: Date:

OUTPUT:

```
F=0  R=-1

1. Add Rear
2. Delete Rear
3. Add Front
4. Delete Front
5. Display
6. Exit
Enter Choice: 1
Enter a number 24_
```

```
F=0  R=0

1. Add Rear
2. Delete Rear
3. Add Front
4. Delete Front
5. Display
6. Exit
Enter Choice: 2
Number Deleted From Rear End = 24
```

```
F=0  R=0

1. Add Rear
2. Delete Rear
3. Add Front
4. Delete Front
5. Display
6. Exit
Enter Choice: 4
Number Deleted From Front End = 22_
```

```
F=1  R=0

1. Add Rear
2. Delete Rear
3. Add Front
4. Delete Front
5. Display
6. Exit
Enter Choice: 5
22 56 67 56 77
```

Exp. No: Date:

SINGLY LINKED LIST

SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void create_list(int);
void addatbeg(int);
void addafter(int,int);
void del(int);
void display();
void rev();
void search(int);
void count();
struct node
{
int info;
struct node *link;
}*start;
void main()
{
int choice=0,n,m,position,i;
start=NULL;
clrscr();
while(1)
{
printf("\n1.INSERTION");
printf("\n2.ADD AT BEGINNING");
printf("\n3.ADD AFTER");
printf("\n4.DELETION");
printf("\n5.DISPLAY");
printf("\n6.COUNT");
printf("\n7.REVERE");
printf("\n8.SEARCH");
printf("\n9.QUIT");
printf("\nENTER THE CHOICE...");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("\nNODES YOU NEED...");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nENTER THE ELEMENT...");
```

```
scanf("%d",&m);
create_list(m);
}
break;
case 2:
printf("\nENTER THE ELEMENT...");
scanf("%d",&m);
addatbeg(m);
break;
case 3:
printf("\nENTER THE ELEMENT...");
scanf("%d",&m);
printf("\nENTER POSTION TO INSERT...");
scanf("%d",&position);
addafter(m,position);
break;
case 4:
if(start==NULL)
{
printf("\nLIST IS EMPTY");
continue;
}
printf("\nENTER THE ELEMENT FOR DELETION...");
scanf("%d",&m);
del(m);
break;
case 5:
display();
break;
case 6:
count();
break;
case 7:
rev();
break;
case 8:
printf("\nENTER THE ELEMENT TO BE SEARCHED...");
scanf("%d",&m);
search(m);
break;
case 9:
exit(0);
default:
printf("\nTHE INVALID CHOICE");
}
}
```

```
}
void create_list(int data)
{
struct node *q,*temp;
temp=malloc(sizeof(struct node));
temp->info=data;
temp->link=NULL;
if(start==NULL)
{
start=temp;
}
else
{
q=start;
while(q->link!=NULL)
{
q=q->link;
}
q->link=temp;
}
}
void addatbeg(int data)
{
struct node *q,*temp;
temp=malloc(sizeof(struct node));
temp->info=data;
temp->link=start;
start=temp;
}
void addafter(int data,int pos)
{
struct node *q,*temp;
int i;
q=start;
for(i=0;i<(pos-1);i++)
{
q=q->link;
if(q==NULL)
{
printf("\nTHERE ARE LESS THAN %d ELEMENT.",pos);
return;
}
}
temp=malloc(sizeof(struct node));
temp->link=q->link;
temp->info=data;
```

```
q->link=temp;
}
void del(int data)
{
struct node *temp,*q;
if(start->info==data)
{
temp=start;
start=start->link;
free(temp);
return;
}
q=start;
while(q->link->link!='\0')
{
if(q->link->info==data)
{
temp=q->link;
q->link=temp->link;
free(temp);
return;
}
q=q->link;
}
if(q->link->info==data)
{
temp=q->link;
free(temp);
q->link='\0';
return;
}
printf("\nELEMENT %d NOT FOUND.",data);
}
void display()
{
struct node *q;
if(start=='\0')
{
printf("\nTHE LIST IS EMPTY");
return;
}
q=start;
while(q!='\0')
{
printf("\nTHE ELEMENT IS...%d",q->info);
q=q->link;
```

```
}
printf("\n");
}
void count()
{
struct node *q=start;
int cnt=0;
while(q!='\0')
{
q=q->link;
cnt++;
}
printf("\nNUMBER OF ELEMENT ARE %d",cnt);
}
void rev()
{
struct node *p1,*p2,*p3;
if(start->link=='\0')
{
return;
}
p1=start;
p2=p1->link;
p3=p2->link;
p1->link='\0';
p2->link=p1;
while(p3!='\0')
{
p1=p2;
p2=p3;
p3=p3->link;
p2->link=p1;
}
start=p2;
printf("\nTO CHECK REVERSED LIST PLEASE CHOOSE OPTION 5");
}
void search(int data)
{
struct node *ptr=start;
int pos=1;
while(ptr!='\0')
{
if(ptr->info==data)
{
printf("\nITEM %d FOUND AT POSITON...%d",data,pos);
return;
```

```
}
ptr=ptr->link;
pos++;
}
if(ptr=='\0')
{
printf("\nITEM %d NOT FOUND IN LIST",data);
}
}
```

Exp. No: Date:

OUTPUT:

```
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...1

NODES YOU NEED...3

ENTER THE ELEMENT...32

ENTER THE ELEMENT...5

ENTER THE ELEMENT...7
```

```
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...2

ENTER THE ELEMENT...3

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
```

```
ENTER THE CHOICE...3

ENTER THE ELEMENT...78

ENTER POSTION TO INSERT...2

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
```

Exp. No: **Date:**

```
ENTER THE CHOICE...5

THE ELEMENT IS...3
THE ELEMENT IS...32
THE ELEMENT IS...78
THE ELEMENT IS...5
THE ELEMENT IS...7

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...4
```

```
ENTER THE ELEMENT FOR DELETION...32

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...6

NUMBER OF ELEMENT ARE 4
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
```

Exp. No: Date:

```
ENTER THE CHOICE...6

NUMBER OF ELEMENT ARE 4
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...7

TO CHECK REVERSED LIST PLEASE CHOOSE OPTION 5
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...8

ENTER THE ELEMENT TO BE SEARCHED...78

ITEM 78 FOUND AT POSITON...3
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...
```

Exp. No: Date:

DOUBLY LINKED LIST

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void create_list(int);
void addatbeg(int);
void addafter(int,int);
void del(int);
void display();
void rev();
void count();
struct node
{
struct node *prev;
int info;
struct node *next;
}*start;
void main()
{
int choice,n,m,po,i;
clrscr();
start=NULL;
while(1)
{
printf("\n 1.CREATE A LIST");
printf("\n 2.ADD AT BEGINNING");
printf("\n 3.ADD AFTER");
printf("\n 4.DELETE");
printf("\n 5.DISPLAY");
printf("\n 6.COUNT");
printf("\n 7.REVERSE");
printf("\n 8.QUIT");
printf("\nENTER YOUR CHOICE...");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("\nENTER NODES YOU NEED...");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nENTER THE ELEMENT...");
scanf("%d",&m);
```

```
create_list(m);
}
break;
case 2:
printf("\nENTER THE ELEMENT...");
scanf("%d",&m);
addatbeg(m);
break;
case 3:

printf("\nENTER THE ELEMENT...");
scanf("%d",&m);
printf(" ENTER THE POSTION...");
scanf("%d",&po);
addafter(m,po);
break;
case 4:
if(start==NULL)
{
printf("\nTHE LIST IS EMPTY");
continue;
}
printf("\nENTER THE DELETION ELEMENT...");
scanf("%d",&m);
del(m);
break;
case 5:
if(start==NULL)
{
printf("\n List is empty!");
continue;
}
display();
break;
case 6:
count();
break;
case 7:
if(start==NULL)
{
printf("\nLIST IS EMPTY!");
continue;
}
rev();
break;
case 8:
```

```
exit(0);
default:
printf("\nWRONG CHOICE");
}
getch();
}
}
void create_list(int num)
{
struct node *q,*tmp;
tmp=malloc(sizeof(struct node));
tmp->info=num;
tmp->next=NULL;
if(start==NULL)
{
tmp->prev=NULL;
start->prev=tmp;
start=tmp;
}
else
{
q=start;
while(q->next!=NULL)
{
q=q->next;
}
q->next=tmp;
tmp->prev=q;
}
}
void addatbeg(int num)
{
struct node *tmp;
tmp=malloc(sizeof(struct node));
tmp->prev=NULL;
tmp->info=num;
tmp->next=start;
start->prev=tmp;
start=tmp;
}
void addafter(int num,int c)
{
struct node *tmp,*q;
int i;
q=start;
for(i=0;i<(c-1);i++)
```

```
{
q=q->next;
if(q==NULL)
{
printf("\nTHERE IS LESS THAN %d ELEMENT",c);
return;
}
}
tmp=malloc(sizeof(struct node));
tmp->info=num;
q->next->prev=tmp;
tmp->next=q->next;
tmp->prev=q;
q->next=tmp;
}
void del(int num)
{
struct node *tmp,*q;
if(start->info==num)
{
tmp=start;
start=start->next;
start->prev=NULL;
free(tmp);
return;
}
q=start;
while(q->next->info!=num)
{
q=q->next;
}
if((q->next->info==num)&&(q->next->next==NULL))
{
tmp=q->next;
free(tmp);
q->next=NULL;
return;
}
if(q->next->info==num)
{
tmp=q->next;
q->next=tmp->next;
tmp->next->prev=q;
free(tmp);
return;
}
```

```
printf("\nTHE ELEMENT %d NOT FOUND",num);
}
void display()
{
struct node *q;
if(start==NULL)
{
printf("\nLIST IS EMPTY");
return;
}
q=start;
printf("\nLIST IS");
while(q!=NULL)
{
printf(" %d ",q->info);
q=q->next;
}
printf("\n");
}
void count()
{
struct node *q=start;
int cnt=0;
while(q!=NULL)
{
q=q->next;
cnt++;
}
printf("\nNUMBER OF ELEMENT IS %d",cnt);
}
void rev()
{
struct node *p1,*p2;
p1=start;
p2=p1->next;
p1->next=NULL;
p1->prev=p2;
while(p2!=NULL)
{
p2->prev=p2->next;
p2->next=p1;
p1=p2;
p2=p2->prev;
}
start=p1;
}
```

Exp. No: Date:

OUTPUT:

```
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...1

NODES YOU NEED...3

ENTER THE ELEMENT...32

ENTER THE ELEMENT...5

ENTER THE ELEMENT...7
```

```
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...2

ENTER THE ELEMENT...3

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
```

```
ENTER THE CHOICE...3

ENTER THE ELEMENT...78

ENTER POSTION TO INSERT...2

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
```

Exp. No: **Date:**

```
ENTER THE CHOICE...5

THE ELEMENT IS...3
THE ELEMENT IS...32
THE ELEMENT IS...78
THE ELEMENT IS...5
THE ELEMENT IS...7

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...4
```

```
ENTER THE ELEMENT FOR DELETION...32

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...6

NUMBER OF ELEMENT ARE 4
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
```

Exp. No: Date:

```
ENTER THE CHOICE...6

NUMBER OF ELEMENT ARE 4
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...7

TO CHECK REVERSED LIST PLEASE CHOOSE OPTION 5
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...8

ENTER THE ELEMENT TO BE SEARCHED...78

ITEM 78 FOUND AT POSITON...3
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...
```

Exp. No: Date:

CIRCULAR LINKED LIST

SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void create_list(int);
void addatbeg(int);
void addafter(int,int);
void del(int);
void display();
struct node
{
int info;
struct node *link;
}*last;
void main()
{
int choice,n,m,po,i;
clrscr();
last=NULL;
while(1)
{
printf("\n 1.CREATE A LIST");
printf("\n 2.ADD AT BEGINNING");
printf("\n 3.ADD AFTER");
printf("\n 4.DELETE");
printf("\n 5.DISPLAY");
printf("\n 6.QUIT");
printf("\nENTER YOUR CHOICE...");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("\nENTER NODES YOU NEED...");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nENTER THE ELEMENT...");
scanf("%d",&m);
create_list(m);
}
break;
case 2:
printf("\nENTER THE ELEMENT...");
```

```
scanf("%d",&m);
addatbeg(m);
break;
case 3:

printf("\nENTER THE ELEMENT...");
scanf("%d",&m);
printf(" ENTER THE POSITION...");
scanf("%d",&po);
addafter(m,po);
break;
case 4:
if(last==NULL)
{
printf("\nLIST UNDERFLOW");
continue;
}
printf("\nENTER THE DELETION ELEMENT...");
scanf("%d",&m);
del(m);
printf("\n THE ELEMENT %d IS DELETED",m);
break;
case 5:
display();
break;
case 6:
exit(0);
default:
printf("\nWRONG CHOICE");
}
getch();
}
}
void create_list(int num)
{
struct node *q,*tmp;
tmp=malloc(sizeof(struct node));
tmp->info=num;
if(last==NULL)
{
last=tmp;
tmp->link=last;
}
else
{
tmp->link=last->link;
```

```
last->link=tmp;
last=tmp;
}
}
void addatbeg(int num)
{
struct node *tmp;
tmp=malloc(sizeof(struct node));
tmp->info=num;
tmp->link=last->link;
last->link=tmp;
}
void addafter(int num,int pos)
{
struct node *tmp,*q;
int i;
q=last->link;
for(i=0;i<(pos-1);i++)
{
q=q->link;
if(q==last->link)
{
printf("\nTHERE ARE LESS THAN %d ELENEMT",pos);
return;
}
}
tmp=malloc(sizeof(struct node));
tmp->link=q->link;
tmp->info=num;
q->link=tmp;
if(q==last)
{
last=tmp;
}
}
void del(int num)
{
struct node *tmp,*q;
if(last->link==last && last->info==num)
{
tmp=last;
last=NULL;
free(tmp);
return;
}
q=last->link;
```

```
if(q->info==num)
{
tmp=q;
last->link=q->link;
free(tmp);
return;
}
while(q->link!=last)
{
if(q->link->info == num)  // Element deleted in between
{
tmp=q->link;
q->link=tmp->link;
free(tmp);
printf("\nTHE ELEMENT %d IS DELETED",num);
return;
}
q=q->link;
}
if(q->link->info==num)
{
tmp=q->link;
q->link=last->link;
free(tmp);
last=q;
return;
}
printf("\nTHE ELEMENT %d NOT FOUND",num);
}
void display()
{
struct node *q;
if(last==NULL)
{
printf("\nLIST IS EMPTY");
return;
}
q=last->link;
printf("\nLIST IS");
while(q!=last)
{
printf(" %d ",q->info);
q=q->link;
}
printf(" %d ",last->info);
}
```

Exp. No: Date:

OUTPUT:

```
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...1

NODES YOU NEED...3

ENTER THE ELEMENT...32

ENTER THE ELEMENT...5

ENTER THE ELEMENT...7
```

```
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...2

ENTER THE ELEMENT...3

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
```

```
ENTER THE CHOICE...3

ENTER THE ELEMENT...78

ENTER POSTION TO INSERT...2

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
```

Exp. No: **Date:**

```
ENTER THE CHOICE...5

THE ELEMENT IS...3
THE ELEMENT IS...32
THE ELEMENT IS...78
THE ELEMENT IS...5
THE ELEMENT IS...7

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...4
```

```
ENTER THE ELEMENT FOR DELETION...32

1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...6

NUMBER OF ELEMENT ARE 4
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
```

Exp. No: Date:

```
ENTER THE CHOICE...6

NUMBER OF ELEMENT ARE 4
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...7

TO CHECK REVERSED LIST PLEASE CHOOSE OPTION 5
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...8

ENTER THE ELEMENT TO BE SEARCHED...78

ITEM 78 FOUND AT POSITON...3
1.INSERTION
2.ADD AT BEGINNING
3.ADD AFTER
4.DELETION
5.DISPLAY
6.COUNT
7.REVERE
8.SEARCH
9.QUIT
ENTER THE CHOICE...
```

Exp. No: Date:

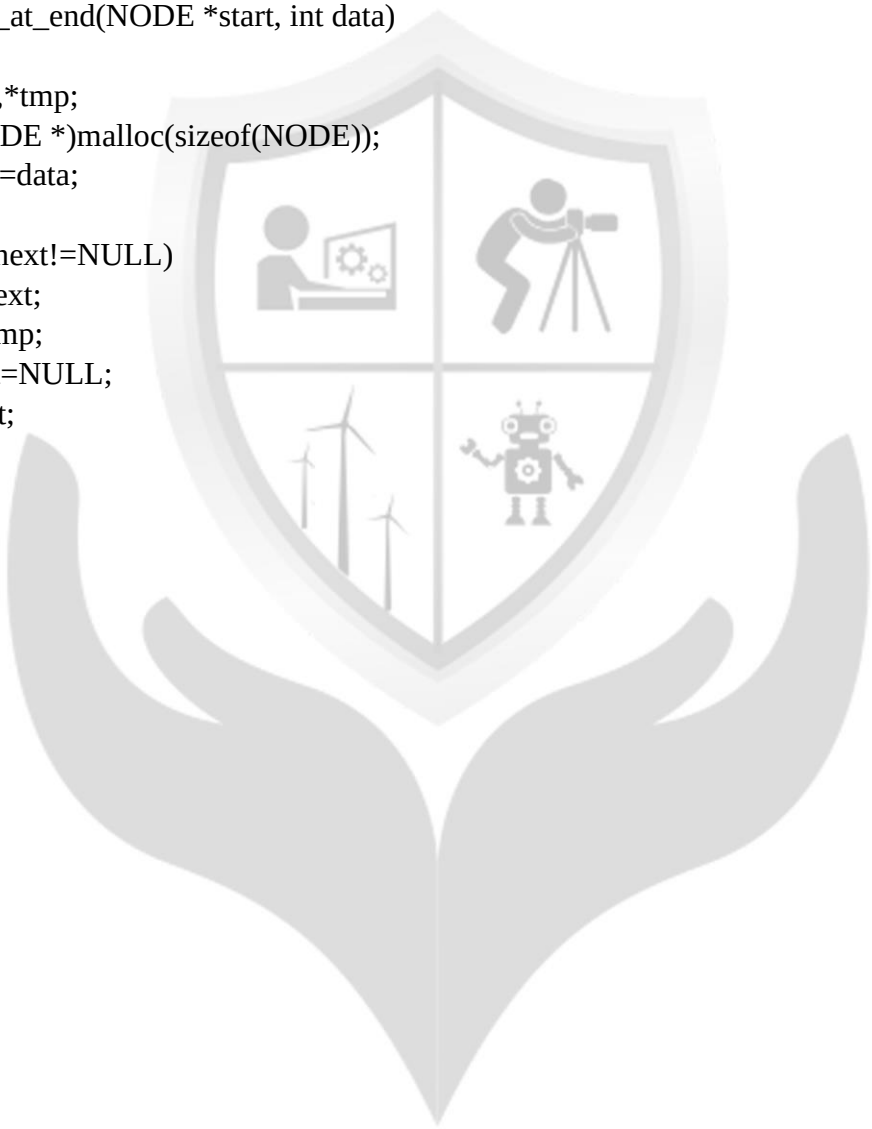
CONCATENATION OF LINKED LIST

SOURCE CODE:

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

typedef struct node
{
   int data;
   struct node *next;
}NODE;
NODE *create_list(NODE *);
NODE *concat( NODE *start1,NODE *start2);
NODE *add_at_beg(NODE *start, int data);
NODE *add_at_end(NODE *start,int data);
void display(NODE *start);

int main()
{
   NODE *start1=NULL,*start2=NULL;
   printf("\nsingle linked list-1\n");
   start1=create_list(start1);
   printf("\nsingle linked list-2\n");
   start2=create_list(start2);
   printf("\nFirst list is  : ");
   display(start1);
   printf("\nSecond list is  : ");
   display(start2);
   start1=concat(start1, start2);
   printf("\nConcatenated list is  : ");
   display(start1);
   getch();
   return 0;
}

NODE *concat( NODE *start1,NODE *start2)
{
   NODE *ptr;
   if(start1==NULL)
   {
      start1=start2;
      return start1;
   }
   if(start2==NULL)
      return start1;
```

```
    ptr=start1;
    while(ptr->next!=NULL)
        ptr=ptr->next;
    ptr->next=start2;
    return start1;
}
NODE *create_list(NODE *start)
{
    int i,n,data;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&n);
    start=NULL;
    if(n==0)
        return start;

    printf("Enter the element to be inserted : ");
    scanf("%d",&data);
    start=add_at_beg(start,data);

    for(i=2;i<=n;i++)
    {
        printf("Enter the element to be inserted : ");
        scanf("%d",&data);
        start=add_at_end(start,data);
    }
    return start;
}

void display(NODE *start)
{
    NODE *p;
    if(start==NULL)
    {
        printf("\nList is empty\n");
        return;
    }
    p=start;
    while(p!=NULL)
    {
        printf("%d ", p->data);
        p=p->next;
    }
    printf("\n");
}

NODE *add_at_beg(NODE *start,int data)
```

```
{
   NODE *tmp;
   tmp=(NODE *)malloc(sizeof(NODE));
   tmp->data=data;
   tmp->next=start;
   start=tmp;
   return start;
}

NODE *add_at_end(NODE *start, int data)
{
   NODE *p,*tmp;
   tmp= (NODE *)malloc(sizeof(NODE));
   tmp->data=data;
   p=start;
   while(p->next!=NULL)
      p=p->next;
   p->next=tmp;
   tmp->next=NULL;
   return start;
}
```



Exp. No: Date:

OUTPUT:

```
single linked list-1

Enter the number of nodes : 2
Enter the element to be inserted : 22
Enter the element to be inserted : 24

single linked list-2

Enter the number of nodes : 2
Enter the element to be inserted : 14
Enter the element to be inserted : 25

First list is  : 22 24

Second list is  : 14 25

Concatenated list is  : 22 24 14 25
```

Exp. No: Date:

TREE TRAVERSAL TECHNIQUES

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
struct node
{
        int data;
        struct node *right,*left;
}*root,*p,*q;
struct node *make(int y)
{
        struct node *newnode;
        newnode=(struct node *)malloc(sizeof(struct node));
        newnode->data=y;
        newnode->right=newnode->left=NULL;
        return(newnode);
}
void left(struct node *r,int x)
{
        if(r->left!=NULL)
        {
                printf("\nINVALID");
        }
        else
        {
                r->left=make(x);
        }
}
void right(struct node *r,int x)
{
        if(r->right!=NULL)
        {
                printf("\nINVALID");
        }
        else

                r->right=make(x);
}

void inorder(struct node *r)
{
        if(r!=NULL)
        {
                inorder(r->left);
```

```
                printf("\t %d",r->data);
                inorder(r->right);
        }
}

void preorder(struct node *r)
{
        if(r!=NULL)
        {
                printf("\t %d",r->data);
                preorder(r->left);
                preorder(r->right);
        }
}
void postorder(struct node *r)
{
        if(r!=NULL)
        {
                postorder(r->left);
                postorder(r->right);
                printf("\t %d",r->data);
        }
}
void main()
{
        int no;
        int choice;
        clrscr();
        printf("\n\t\t\t\tBINARY TREE TRAVERSALS");
        printf("\nENTER THE ROOT NODE...");
        scanf("%d",&no);
        root=make(no);
        p=root;
        while(1)
        {
                printf("\nDO YOU NEED ANOTHER NODE...");
                printf("\n1.)YES");
                printf("\n2.)NO");
                printf("\nENTER YOUR CHOICE...");
                scanf("%d",&choice);
                if(choice==1)
                {
                        printf("\nENTER ANOTHER NUMBER...");
                        scanf("%d",&no);
                        while(1)
                        {
```

```
                        p=root;
                        q=root;
                        while((no!=p->data)&&(q!=NULL))
                        {
                                p=q;
                                if(no<p->data)
                                {
                                        q=p->left;
                                }
                                else
                                {
                                        q=p->right;
                                }
                        }
                        if(no<p->data)
                        {
                                printf("\nLEFT BRANCH OF %d IS %d",p->data,no);
                                left(p,no);
                                break;
                        }
                        else
                        {
                                right(p,no);
                                printf("\nRIGHT BRANCH OF %d IS %d",p-
>data,no);
                                break;
                        }
                }
        }
        else if(choice==2)
        {
                break;
        }
        else
        {
                printf("\nINVALID OPTION");
                continue;
        }
    }
    while(1)
    {
            printf("\n1.)INORDER TRAVERSALS");
            printf("\n2.)PREORDER TRAVERSALS");
            printf("\n3.)POST TRAVERSALS");
            printf("\n4.)EXIT");
            printf("\nENTER YOUR CHOICE...");
```

```
                scanf("\%d",&choice);
                switch(choice)
                {
                        case 1:
                                inorder(root);
                        break;
                        case 2:
                                preorder(root);
                        break;
                        case 3:
                                postorder(root);
                        break;
                        case 4:
                                exit(0);
                        default:
                                printf("\nINVALID OPTION");
                        break;
                }
                getch();
        }
}
```

Exp. No: Date:

OUTPUT:

```
                                BINARY TREE TRAVERSALS
ENTER THE ROOT NODE...5

DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1

ENTER ANOTHER NUMBER...3

LEFT BRANCH OF 5 IS 3
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1

ENTER ANOTHER NUMBER...7

RIGHT BRANCH OF 5 IS 7
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1
```

```
ENTER YOUR CHOICE...1

ENTER ANOTHER NUMBER...2

LEFT BRANCH OF 3 IS 2
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1

ENTER ANOTHER NUMBER...4

RIGHT BRANCH OF 3 IS 4
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1

ENTER ANOTHER NUMBER...6

LEFT BRANCH OF 7 IS 6
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...
```

Exp. No: **Date:**

```
LEFT BRANCH OF 7 IS 6
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...1

ENTER ANOTHER NUMBER...9

RIGHT BRANCH OF 7 IS 9
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...2

1.)INORDER TRAVERSALS
2.)PREORDER TRAVERSALS
3.)POST TRAVERSALS
4.)EXIT
ENTER YOUR CHOICE...1
        2       3       4       5       6       7       9
1.)INORDER TRAVERSALS
2.)PREORDER TRAVERSALS
3.)POST TRAVERSALS
4.)EXIT
ENTER YOUR CHOICE...2
```

```
RIGHT BRANCH OF 7 IS 9
DO YOU NEED ANOTHER NODE...
1.)YES
2.)NO
ENTER YOUR CHOICE...2

1.)INORDER TRAVERSALS
2.)PREORDER TRAVERSALS
3.)POST TRAVERSALS
4.)EXIT
ENTER YOUR CHOICE...1
        2       3       4       5       6       7       9
1.)INORDER TRAVERSALS
2.)PREORDER TRAVERSALS
3.)POST TRAVERSALS
4.)EXIT
ENTER YOUR CHOICE...2
        5       3       2       4       7       6       9
1.)INORDER TRAVERSALS
2.)PREORDER TRAVERSALS
3.)POST TRAVERSALS
4.)EXIT
ENTER YOUR CHOICE...3
        2       4       3       6       9       7       5
```

Exp. No: Date:

BREADTH FIRST SEARCH

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<stdlib.h>
# define MAX 10
# define TRUE 1
# define FALSE 0
typedef struct node
{
int vertex;
struct node *next;
}node;
node *head[10];
int visited[MAX];
int Queue[MAX];
int front,rear;
int n;
void add(int vertex1, int vertex2)
{
node *New, *first;
New=(node *)malloc(sizeof(node));
if(New==NULL)
printf("\n Insuffucient memory");
New->vertex=vertex2;
New->next=NULL;
first=head[vertex1];
if(first==NULL)
head[vertex1]=New;
else
{
while(first->next!=NULL)
first=first->next;
first->next=New;
}
New=(node *)malloc(sizeof(node));
if(New==NULL)
printf("\n Insufficient memory");
New->vertex=vertex1;
New->next=NULL;
first=head[vertex2];
if(first==NULL)
head[vertex2]=New;
```

```
else
{
while(first->next!=NULL)
first=first->next;
first->next=New;
}
}
void create()
{
int V1,V2;
char ans='y';
for(V1=0;V1<MAX;V1++)
head[V1]=NULL;
printf("\n enter the vertices no. beginning witn 0");
do
{
printf("\n Enter the edge of the graph");
scanf("%d %d",&V1,&V2);
if(V1>=MAX || V2>=MAX)
printf("\n Invalid Vertex value");
else
add(V1,V2);
printf("\n Want to add more edges(y/n)");
ans=getche();
}while(ans=='y');
}
void bfs(int V1)
{
int i;
node *first;
front=-1;
rear=-1;
Queue[++rear]=V1;
while(front!=rear)
{
i=Queue[++front];
if(visited[i]==FALSE)
{
printf("\n %d",i);
visited[i]=TRUE;
}
first=head[i];
while(first!=NULL)
{
if(visited[first->vertex]==FALSE)
Queue[++rear]=first->vertex;
```

```
first=first->next;
}
}
}
void main()
{
int V1,V2;
char ans;
clrscr();
create();
do
{
for(V1=0;V1<n;V1++)
visited[V1]=FALSE;
clrscr();
printf("\n Enter the vertex from which u want to traverse");
scanf("%d",&V1);
if(V1>=MAX)
printf("\n Invalid vertex");
else
{
printf("\n The breadth first search of the Graph is\n");
front=rear=-1;
bfs(V1);
getch();
}
printf("\n Do u want to continue");
ans=getche();
}
while(ans=='y');
exit(0);
}
```

Exp. No: Date:

OUTPUT:

```
 enter the vertices no. beginning witn 0
 Enter the edge of the graph0
1

 Want to add more edges(y/n)y
 Enter the edge of the graph0
2

 Want to add more edges(y/n)y
 Enter the edge of the graph0
3

 Want to add more edges(y/n)y
 Enter the edge of the graph0
4

 Want to add more edges(y/n)_


 Enter the vertex from which u want to traverse0

 The breadth first search of the Graph is

 0
 1
 2
 3
 4
```

Exp. No: Date:

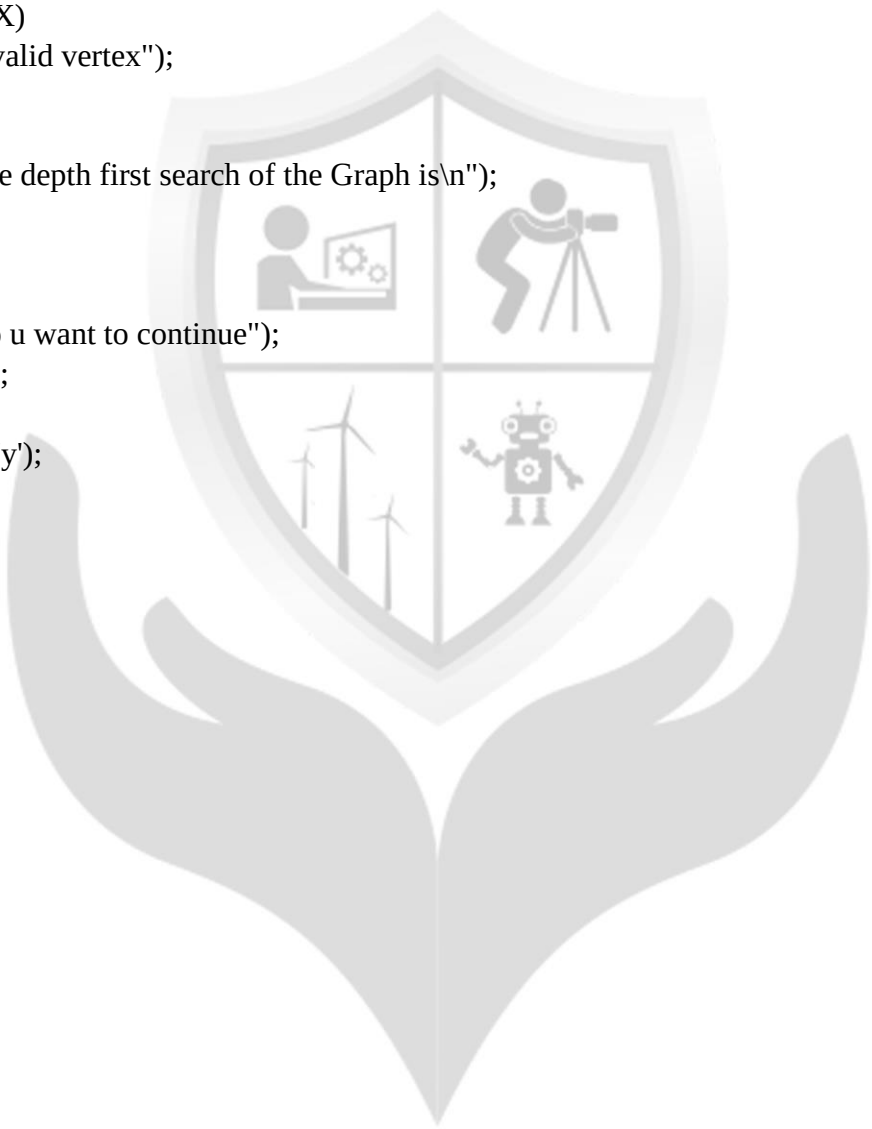
DEPTH FIRST SEARCH

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<stdlib.h>
# define MAX 10
# define TRUE 1
# define FALSE 0
typedef struct node
{
int vertex;
struct node *next;
}node;
node *head[10];
int visited[MAX];
int Queue[MAX];
int front,rear;
int n;
void add(int vertex1, int vertex2)
{
node *New, *first;
New=(node *)malloc(sizeof(node));
if(New==NULL)
printf("\n Insuffucient memory");
New->vertex=vertex2;
New->next=NULL;
first=head[vertex1];
if(first==NULL)
head[vertex1]=New;
else
{
while(first->next!=NULL)
first=first->next;
first->next=New;
}
New=(node *)malloc(sizeof(node));
if(New==NULL)
printf("\n Insufficient memory");
New->vertex=vertex1;
New->next=NULL;
first=head[vertex2];
if(first==NULL)
head[vertex2]=New;
```

```
else
{
while(first->next!=NULL)
first=first->next;
first->next=New;
}
}
void create()
{
int V1,V2;
char ans='y';
for(V1=0;V1<MAX;V1++)
head[V1]=NULL;
printf("\n enter the vertices no. beginning witn 0");
do
{
printf("\n Enter the edge of the graph");
scanf("%d %d",&V1,&V2);
if(V1>=MAX || V2>=MAX)
printf("\n Invalid Vertex value");
else
add(V1,V2);
printf("\n Want to add more edges(y/n)");
ans=getche();
}while(ans=='y');
}
void dfs(int V1)
{
int V2;
node *first;

printf("\n %d",V1);
visited[V1]=TRUE;

first=head[V1];
while(first!=NULL)

if(visited[first->vertex]==FALSE)
dfs(first->vertex);
else
first=first->next;
}
void main()
{
int V1,V2;
char ans='y';
```

```
clrscr();
create();
do
{
for(V1=0;V1<n;V1++)
visited[V1]=FALSE;
clrscr();
printf("\n Enter the vertex from which u want to traverse");
scanf("%d",&V1);
if(V1>=MAX)
printf("\n Invalid vertex");
else
{
printf("\n The depth first search of the Graph is\n");
dfs(V1);
getch();
}
printf("\n Do u want to continue");
ans=getche();
}
while(ans=='y');
exit(0);
}
```



Exp. No: **Date:**

OUTPUT:

```
 enter the vertices no. beginning witn 0
 Enter the edge of the graph0
1

 Want to add more edges(y/n)y
 Enter the edge of the graph0
2

 Want to add more edges(y/n)y
 Enter the edge of the graph1
3

 Want to add more edges(y/n)y
 Enter the edge of the graph2
3

 Want to add more edges(y/n)
```

```
 Enter the vertex from which u want to traverse0

 The depth first search of the Graph is

 0
 1
 3
 2
 Do u want to continue
```

Exp. No: Date:

DIJKSTRA ALGORITHM

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 1000
int n,cost[10][10],dist[10],visit[10],front,rear,v;
void Initial(int n)
{
	int i,j;
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++)
		{
			cost[i][j] = INFINITY;
		}
	}
	for(i=1;i<=n;i++)
	{
		dist[i] = INFINITY;
		visit[i] = 0;
	}
}
int nearest()
{
	int i,node, weight;
	weight = INFINITY;
	for(i=1;i<=n;i++)
	{
		if(visit[i] == 0 &&dist[i] < weight)
		{
			weight = dist[i];
			node=i;
		}
	}
	return node;
}
void Dijkstra(int v)
{
	int i,u,q,alt;
	dist[v]=0;
	for(i=1;i<=n;i++)
	{
		u=nearest();
		for(v=1;v<=n;v++)
```

```
                {
                        if(cost[u][v] == INFINITY)
                        {
                                continue;
                        }
                        alt = dist[u] + cost[u][v];
                        if(alt <dist[v])
                        {
                                dist[v] = alt;
                        }
                }
                visit[u] = 1;
        }
 }
void display()
{
        int i,j;
        printf("\nCOST MATRIX\n");
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        printf("%d\t",cost[i][j]);
                }
                printf("\n");
        }
}
void main()
{
        int i,j,k;
        clrscr();
        printf("ENTER THE NUMBER OF VERTCIES...");
        scanf("%d",&n);
        Initial(n);
        for(i=1;i<=n;i++)
        {
                for(j=i+1;j<=n;j++)
                {
                        printf("ENTER THE SHORTEST PATH BETWEEN %d AND
%d",i,j);
                        printf("\nIF EDGE NOT EXIST ENTER 1000...");
                        scanf("%d",&cost[i][j]);
                        cost[j][i] = cost[i][j];
                }
        }
        display();
```

```
        printf("\nENTER THE SOURCE VERTEX...");
        scanf("%d",&v);
        Dijkstra(v);
        printf("\nRESULT\n");
        for(i=1;i<=n;i++)
        {
                printf("\nDISTANCE OF %d IS...%d",i,dist[i]);
        }
        getch();
}
```

Exp. No: Date:

OUTPUT:

```
ENTER THE NUMBER OF VERTCIES...6
ENTER THE SHORTEST PATH BETWEEN 1 AND 2
IF EDGE NOT EXIST ENTER 1000...1
ENTER THE SHORTEST PATH BETWEEN 1 AND 3
IF EDGE NOT EXIST ENTER 1000...12
ENTER THE SHORTEST PATH BETWEEN 1 AND 4
IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 1 AND 5
IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 1 AND 6
IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 2 AND 3
IF EDGE NOT EXIST ENTER 1000...9
ENTER THE SHORTEST PATH BETWEEN 2 AND 4
IF EDGE NOT EXIST ENTER 1000...3
ENTER THE SHORTEST PATH BETWEEN 2 AND 5
IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 2 AND 6
IF EDGE NOT EXIST ENTER 1000...1000
ENTER THE SHORTEST PATH BETWEEN 3 AND 4
IF EDGE NOT EXIST ENTER 1000...4
ENTER THE SHORTEST PATH BETWEEN 3 AND 5
IF EDGE NOT EXIST ENTER 1000...5
ENTER THE SHORTEST PATH BETWEEN 3 AND 6
IF EDGE NOT EXIST ENTER 1000...
1000
ENTER THE SHORTEST PATH BETWEEN 4 AND 5
IF EDGE NOT EXIST ENTER 1000...13
ENTER THE SHORTEST PATH BETWEEN 4 AND 6
IF EDGE NOT EXIST ENTER 1000...15
ENTER THE SHORTEST PATH BETWEEN 5 AND 6
IF EDGE NOT EXIST ENTER 1000...2

COST MATRIX
1000    1       12      1000    1000    1000
1       1000    9       3       1000    1000
12      9       1000    4       5       1000
1000    3       4       1000    13      15
1000    1000    5       13      1000    2
1000    1000    1000    15      2       1000

ENTER THE SOURCE VERTEX...1

RESULT

DISTANCE OF 1 IS...0
DISTANCE OF 2 IS...1
DISTANCE OF 3 IS...8
DISTANCE OF 4 IS...4
DISTANCE OF 5 IS...13
DISTANCE OF 6 IS...15
```

Exp. No: Date:

HASH TABLES

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#define MAX 10
int a[MAX];
void main()
{
 int j=1,i,n,k,l;
clrscr();
for(i=0;i<5;i++)
a[i]=-1;
while(j!=0)
{
printf("\n Enter the Elements");
scanf("%d",&n);
k=n%10;
i=k;
if(a[k]!=n)
{
while(a[i]!=-1)
{
i++;
i=i%10;
if(i==k)
{
printf("\nthe table is full");
getch();
exit(0);
}
}
a[i]=n;
}
else
a[k]=n;
printf("\na[%d]=%d",i,a[i]);
}
getch();
}
```

Exp. No: Date:

OUTPUT:

```
 Enter the Elements3

a[3]=3
 Enter the Elements2

a[2]=2
 Enter the Elements1

a[1]=1
 Enter the Elements4

a[4]=4
 Enter the Elements5

a[0]=5
 Enter the Elements6

the table is full_
```