/*Write a C program to find the sum of individual digits of a positive integer.*/

```
#include<stdio.h>
#include<conio.h>
main()
{
int n,r,sum=0;
printf("Enter any positive integer:");
scanf("%d",&n);
while(n>0)
{
r=n%10;
sum=sum+r;
n=n/10;
}
printf("\nSum=%d",sum);
getch();
}
```

/* Fibonacci sequence */

```
#include <stdio.h>
#include<conio.h>
main()
{
  int i,n,f0=0,f1=1,f2;
  printf("Enter n value:");
  scanf("%d",&n);
  printf("%d\t%d\t",f0,f1);
  for(i=3;i<=n;i++)
  {
    f2=f1+f0;
    printf("%d\t",f2);
    f0=f1;
    f1=f2;
   }
  getch();
   }
```

/* Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user. */

```
#include <stdio.h>
#include <conio.h>
main()
{
int n,i,j,count=0;
printf("enter any postive integer");
scanf("%d",&n);
for(i=2;i<=n;i++)
{
for(j=1;j<=i;j++)
{
if(i%j==0)
count++;
```

```
}
if(count==2)
printf("%d\t",i);
count=0;
}
getch();
}
```

/* Write a C program to calculate the following Sum: Sum=1-x2/2! +x4/4!-x6/6! +x8/8!-x10/10! */

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int fact(int);
main()
 {
  float sum=0;
  int x,k=1,i;
  printf("Enter the value of x:");
  scanf("%d",&x);
  for(i=0;i<=10;i=i+2)
   {
    sum=sum+(k*pow(x,i))/fact(i);
    k=-k;
   }
  printf("sum of the series=%f",sum);
  getch();
 }
int fact(int n)
 {
   int i,f=1;
   for(i=2;i<=n;i++)
    f=f*i;
   return f;
 }
```

/* Write a C program to find the roots of a quadratic equation. */

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
 main()
{
  float a,b,c,d,r1,r2;
  printf("ENTER THE VALUES OF a,b,c: ");
  scanf("%f %f %f",&a,&b,&c);
  d=(b*b)-(4*a*c);
  if(d==0)
  {
  printf("\n ROOTS ARE EQUAL: ");
  r1=-b/(2*a);
  r2=-b/(2*a);
  printf("\nTHE VALUE OF ROOT1 IS =%f",r1);
```

```
  printf("\nTHE VALUE OF ROOT2 IS =%f",r2);
 }
  else if(d>0)
 {
  printf("ROOTS ARE REAL AND DISTINCT:");
  r1=(-b+sqrt(d))/(2*a);
  r2=(-b-sqrt(d))/(2*a);
  printf("\n THE VALUE OF ROOT1 IS =%f",r1);
  printf("\n THE VALUE OF ROOT2 IS =%f",r2);
 }
 else
 {
 printf("\n ROOTS ARE IMAGINARY");
 }
 getch();
}
```

/*finding factorial of given integer without Recursion */

```
#include<stdio.h>
#include<conio.h>
int fact(int);
 main()
{
  int n,result;

  printf("ENTER n VALUE:");
  scanf("%d",&n);
  result=fact(n);
  printf("FACTORIAL is:%d",result);
  getch();
}
int fact(int n)
{
  int f=1,i;
  for(i=n;i>=1;i--)
  f=f*i;
  return f;
}
```

/* finding factorial of given integer with recursion */

```
#include<stdio.h>
#include<conio.h>
int fact(int);
main()
{
  int n,result;
  printf("ENTER n VALUE:");
  scanf("%d",&n);
  result=fact(n);
  printf("FACTORIAL is:%d",result);
  getch();
}
```

```
int fact(int n)
{
   if(n==0)
return 1;
   else
   return n*fact(n-1);
}
```

/* finding gcd of given integers with recursion */

```
#include<stdio.h>
#include<conio.h>
int gcd(int,int);
 main()
{
  int a,b,result;
  printf("ENTER TWO INTEGERS :");
  scanf("%d%d",&a,&b);
  result=gcd(a,b);
  printf("GCD is:%d", result);
  getch();
}
int gcd(int a,int b)
{
  if(b>a)
  return gcd(b,a);
  if(b==0)
  return a;
  else
  return gcd(b,a%b);
}
```

/* To solve Towers of HANOI problem */

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void hanoi( int n,char s,char i,char d) ;
main()
{
int n,m;
char s='L',i='C',d='R';
printf("\n enter the number of disks");
scanf("%d",&n);
printf("\n Towers of hanoi problem with %d disks",n);
hanoi(n,s,i,d);
m=pow(2,n)-1;
printf("\n Total number of moves=%d", m);
getch();
}
void hanoi(int n,char s,char i,char d)
{
if(n!=0)
```

```
{
hanoi(n-1,s,d,i);
printf("\nmove disk%d from %c to %c",n,s,d);
hanoi(n-1,i,s,d);
}
}
```

/* The total distance travelled by vehicle */

```
#include <stdio.h>
#include<conio.h>
 main()
{
int s,u,a,t=0,t1,interval;
printf("Enter the values for u and a");
scanf("%d%d",&u,&a);
printf("Enter the end time");
scanf("%d",&t1);
printf("Enter the  time interval");
scanf("%d",&interval);
for(t=0;t<=t1;t=t+interval)
{
s=u*t+(0.5)*a*t*t;
printf("\n distance travelled is %d sec=%d\n",t,s);
}
 getch();
}
```

/* Write a C program, which takes two integer operands and one operator form the user,performs the operation and then prints the result. (Consider the operators +,-,*, /, % and use Switch Statement)*/

```
#include<stdio.h>
#include<conio.h>
 main()
{
int a,b,res;
char ch;
printf("\t   **********");
printf("\n\tMENU\n");
printf("\t**********");
printf("\n\t(+)ADDITION");
printf("\n\t(-)SUBTRACTION");
printf("\n\t(*)MULTIPLICATION");
printf("\n\t(/)DIVISION");
printf("\n\t(%)REMAINDER");
printf("\n\t**********");
printf("\n\n\tEnter your choice:");
scanf("%c",&ch);
printf("Enter two numbers:\n");
scanf("%d%d",&a,&b);
switch(ch)
{
```

```
 case '+':
 res=a+b;
 printf("\n Addition:%d",res);
 break;
 case '-':
 res=a-b;
 printf("\n Subtraction:%d",res);
 break;

 case '*':
 res=a*b;
 printf("\n Multiplication:%d",res);
 break;

 case '/':
 res=a/b;
 printf("\n Division:%d",res);
 break;

 case '%':
 res=a%b;
 printf("\n Remainder:%d",res);
 break;
 default:
 printf("\n Invalid Choice");
}
getch();
}
```

// TO FIND LARGEST AND SMALLEST NUMBER IN AN ARRAY

```
#include<stdio.h>
int main()
{
  int a[50],size,i,big,small;

  printf("\nEnter the size of the array: ");
  scanf("%d",&size);
  printf("\nEnter %d elements in to the array: ", size);
  for(i=0;i<size;i++)
      scanf("%d",&a[i]);

  big=a[0];
  for(i=1;i<size;i++){
      if(big<a[i])
          big=a[i];
  }
  printf("\nLargest element: %d\n",big);

  small=a[0];
  for(i=1;i<size;i++){
      if(small>a[i])
          small=a[i];
```

```
  }
  printf("\nSmallest element: %d\n",small);
  getch();
  return 0;
}
```

/* Write a C program that uses functions to perform the following:
i) Addition of Two Matrices ii) Multiplication of Two Matrices*/

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int read_matrix(int a[10][10],int m,int n);
int write_matrix(int a[10][10],int m,int n);
 main()
{
int ch,i,j,m,n,p,q,k,r1,c1,a[10][10],b[10][10],c[10][10];

printf("**********************************");
printf("\n\t\tMENU");
printf("\n**********************************");
printf("\n[1]ADDITION OF TWO MATRICES");
printf("\n[2]MULTIPLICATION OF TWO MATRICES");
printf("\n[0]EXIT");
printf("\n**********************************");
printf("\n\tEnter your choice:\n");
scanf("%d",&ch);

if(ch<=2 & ch>0)
{
 printf("Valid Choice\n");
}

switch(ch)
{
  case 1:
  printf("Input rows and columns of A & B Matrix:");
  scanf("%d%d",&r1,&c1);
  printf("Enter elements of matrix A:\n");
    for(i=0;i<r1;i++)
    {
      for(j=0;j<c1;j++)
      scanf("%d",&a[i][j]);
    }
  printf("Enter elements of matrix B:\n");
    for(i=0;i<r1;i++)
    {
      for(j=0;j<c1;j++)
      scanf("%d",&b[i][j]);
    }
  printf("\n =====Matrix Addition=====\n");
    for(i=0;i<r1;i++)
```

```
    {
      for(j=0;j<c1;j++)
      printf("%5d",a[i][j]+b[i][j]);
      printf("\n");
    }
  break;

  case 2:
  printf("Input rows and columns of A matrix:");
  scanf("%d%d",&m,&n);
  printf("Input rows and columns of B matrix:");
  scanf("%d%d",&p,&q);
  if(n==p)
  {
  printf("matrices can be multiplied\n");
  printf("resultant matrix is %d*%d\n",m,q);
  printf("Input A matrix\n");
  read_matrix(a,m,n);
  printf("Input B matrix\n");
  /*Function call to read the matrix*/
  read_matrix(b,p,q);
  /*Function for Multiplication of two matrices*/
  printf("\n =====Matrix Multiplication=====\n");
  for(i=0;i<m;++i)
    for(j=0;j<q;++j)
    {
      c[i][j]=0;
      for(k=0;k<n;++k)
          c[i][j]=c[i][j]+a[i][k]*b[k][j];
    }

  printf("Resultant of two matrices:\n");
     write_matrix(c,m,q);
    }
    /*end if*/
  else
  {
  printf("Matrices cannot be multiplied.");
  }
  /*end else*/
  break;

  case 0:
  printf("\n Choice Terminated");
  exit(0);
  break;

  default:
  printf("\n Invalid Choice");
}
getch();
}
```

```
/*Function read matrix*/
int read_matrix(int a[10][10],int m,int n)
  {
    int i,j;
      for(i=0;i<m;i++)
          for(j=0;j<n;j++)
          scanf("%d",&a[i][j]);
  return 0;
  }

  /*Function to write the matrix*/
int write_matrix(int a[10][10],int m,int n)
  {
    int i,j;
      for(i=0;i<m;i++)
      {
          for(j=0;j<n;j++)
          printf("%5d",a[i][j]);
          printf("\n");
      }
   return 0;
   }
```

// program to insert substring into a main string from a given position

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
 main()
{
char s1[15],s2[15],s3[15]="";
int i,j,k,pos;

printf("Enter Main String : ");
gets(s1);
printf("Enter Sub String : ");
gets(s2);
printf("enter the position(index) to insert the substring :");
scanf("%d", &pos);
strcpy(s3,s1);
for(i=pos,j=0;j<=strlen(s2);i++,j++)
s1[i]=s2[j];
for(j=pos,i=strlen(s1);j<=strlen(s3);i++,j++)
s1[i]=s3[j];
s1[i]='\0';
puts(s1);
getch();
}
```

// program to delete n characters from a given position in a given string

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

```
#include<string.h>
 main()
{
char s1[15];
int i,j,l,pos,n;
printf("Enter Any String : ");
gets(s1);
printf("Enter the position(index) to        delete the characters :");
scanf("%d", &pos);
printf("Enter the no of characters to     delete:");
scanf("%d", &n);
l=strlen(s1);
if(pos>l)
printf("\n Deletion is not possible");
else
{
 for(i=pos,j=pos+n;i<=l;i++,j++)
s1[i]=s1[j];
s1[i]='\0';
}
puts(s1);
getch();
}
```

// program to determine if the given string is palindrome or not

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
 main()
{
char s1[15];
int i,j,l,pos,n;
printf("Enter Any String : ");
gets(s1);
printf("Enter the position(index) to        delete the characters :");
scanf("%d", &pos);
printf("Enter the no of characters to     delete:");
scanf("%d", &n);
l=strlen(s1);
if(pos>l)
printf("\n Deletion is not possible");
else
{
 for(i=pos,j=pos+n;i<=l;i++,j++)
s1[i]=s1[j];
s1[i]='\0';
}
puts(s1);
getch();
}
```

// program to display the position or index of string or -1 if not present

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
 main()
{
 char s[30],t[20];
 char *found;
 puts("Enter  the main string");
 gets(s);
 puts("Enter the string to be searched");
 gets(t);
 found=strstr(s,t);
 if(found)
 printf("Second string found in the main String at %d index",found-s);
 else
 printf("-1");
 getch();
}
```

// program to count lines, words, and characters in a given text

```
#include<stdio.h>
#include<conio.h>
 main()
{
char str[100],c;
int i=0,wc=1,lc=1,ch=0;

printf("enter text at end give # symbol \n");
c=getchar();
while (c!='#')
{
str[i]=c;
if (c!='\n')
ch++;
i++;
c=getchar();
}
str[i]='\0';
i=0;
while(str[i]!='\0')
{
if(str[i]=='\n')
{
lc++;
wc++;
}
else
{
if(str[i]==' '&&str[i+1]!=' ')
wc++;
}
i++;
```

```
}
printf("\n number of characters=%d",ch);
printf("\n number of words=%d",wc);
printf("\n number of lines=%d",lc);
getch();
}
```

/* program to generate pasal's triangle*/

```
#include<stdio.h>
#include<conio.h>
main()
{
int bin,p,q,r,x;
bin=1;
q=0;
printf("\nEnter number of Rows");
scanf("%d",&r);
printf("\npascal Triangle\n");
while(q<r)
{
for(p=40-3*q;p>0;p--)
printf(" ");
for(x=0;x<=q;x++)
{
if((x==0)||(q==0))
bin=1;
else
bin=(bin*(q-x+1))/x;
printf("%6d",bin);
}
printf("\n");
q++;
}
getch();
}
```

/* Write a C program to construct a pyramid of numbers. */

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
 main()
{
 int num,i,y,x=35;

 printf("\nEnter the number to generate the pyramid:\n");
 scanf("%d",&num);

 for(y=0;y<=num;y++)
 {
  /*(x-coordinate,y-coordinate)*/
  gotoxy(x,y+1);
```

```
  /*for displaying digits towards the left and right of zero*/
  for(i=0-y;i<=y;i++)

  printf("%3d",abs(i));
  x=x-3;
 }
 getch();
}
```

/* write a C program t read in two numbers, x and n, and then compute the sum of theis geometric progression: 1+x+x2+x3+.. xn..*/

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
int sum=0,n,x,i;

pos: printf("\n Enter the  x value: ");
          scanf("%d",&x);
          printf("\n Enter n:");
          scanf("%d",&n);
          if(n<0)
         {
           printf("\n Entered n value is negative so enter  positive value");
           goto pos;
         }
         else
         for(i=0;i<=n;i++)
        {
         sum=sum+pow(x,i);
         }
          printf("\n Sum of series upto %d terms is:%d",n,sum);
           getch();
}
```

// program to find 2's complement of a binary number

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
 main()
{
 char str[10],i,j;

 printf("Enter Binary Number:");
 gets(str);
 for(i=strlen(str)-1;i>=0;i--)
 if(str[i]=='1')
break;
 for(j=0;j<i;j++)
```

```
 if(str[j]=='1')
 str[j]='0';
 else
 str[j]='1';
 printf("2's Complement is:%s",str);
 getch();
}
```

/*write a c program to convert a roman numeral to its decimal equialent.*/

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
int i,l,num;
char rn[15];
int dn[15];
printf("\n enter a roman number:\n");
gets(rn);
l=strlen(rn);
for(i=l-1;i>=0;i--)
{
switch(rn[i])
{
case 'I': dn[i]=1;  break;
case 'V':dn[i]=5;   break;
case 'X':dn[i]=10;  break;
case  'C':dn[i]=100; break;
case 'D':dn[i]=500;  break;
case 'M':dn[i]=1000; break;
}
}
num=dn[l-1];
for(i=l-1;i>0;i--)
{
if(dn[i-1]>=dn[i])
num=num+dn[i-1];
else
num=num-dn[i-1];
}
printf("%d",num);
getch();
}
```

/* Write a C program that uses functions to perform the following operations:

i) Reading a complex number

ii) Writing a complex number

iii) Addition of two complex numbers

iv) Multiplication of two complex numbers

(Note: represent complex number using a structure.) */

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

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

void arithmetic(int opern);

struct comp
{
    double realpart;
    double imgpart;
};

 main()
{
    int opern;
        printf("\n\n \t\t\t***** MAIN MENU *****");
    printf("\n\n Select your option: \n 1 : ADD\n 2 : MULTIPLY\n 0 : EXIT \n\n\t\t Enter
your Option [  ]\b\b");

    scanf("%d",&opern);

    switch(opern)
    {
          case 0:
              exit(0);
          case 1:
          case 2:
              arithmetic(opern);
          default:
              main();
    }

}

void arithmetic(int opern)
{

    struct comp w1, w2, w;

    printf("\n Enter two Complex Numbers  (x+iy):\n Real Part of First Number:");
    scanf("%lf",&w1.realpart);
    printf("\n Imaginary Part of First Number:");
    scanf("%lf",&w1.imgpart);
    printf("\n Real Part of Second Number:");
    scanf("%lf",&w2.realpart);
    printf("\n Imaginary Part of Second Number:");
    scanf("%lf",&w2.imgpart);


    switch(opern)
    {
```

```
    /*addition of complex number*/
    case 1:
        w.realpart = w1.realpart+w2.realpart;
        w.imgpart = w1.imgpart+w2.imgpart;
        break;

    /*multiplication of complex number*/
    case 2:
        w.realpart=(w1.realpart*w2.realpart)-(w1.imgpart*w2.imgpart);
        w.imgpart=(w1.realpart*w2.imgpart)+(w1.imgpart*w2.realpart);
        break;
    }


    if (w.imgpart>0)
        printf("\n Answer = %lf+%lfi",w.realpart,w.imgpart);
    else
        printf("\n Answer = %lf%lfi",w.realpart,w.imgpart);
    getch();
    main();
}
```

// program to copy contents of one file to another

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
 main()
{
char ch;
char str1[20],str2[20];
FILE *f1;
FILE *f2;
printf("Enter the Source File Name:");
gets(str1);
printf("Enter the Destination File Name :");
gets(str2);
f1=fopen(str1,"r");
if(f1==NULL)
{
printf("Error in opening Source file");
getch();
exit(1);
}
f2=fopen(str2,"w");
if(f2==NULL)
{
printf("Error in opening Destination File");
getch();
exit(1);
}
ch=getc(f1);
```

```
while(ch!=EOF)
{
putc(ch,f2);
ch=getc(f1);
}
fclose(f1);
fclose(f2);
printf("File copying successful");
getch();
}
```

// program to reverse the first n characters in a file

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
 main()
{
int i,n,k,j=0,len;
char s[1000],a[1000],str[20];
FILE *fp;
printf("Enter the Source File Name:");
gets(str);
fp=fopen(str,"r");
if(fp==NULL)
{
printf("Error in opening Source File");
getch();
exit(1);
}
printf("Enter the No. of Characters:");
scanf("%d",&k);
n=fread(a,1,k,fp);
a[n]='\0';
len=strlen(a);
for(i=len-1;i>=0;i--)
{
s[j]=a[i];
printf("%c",s[j]);
j=j+1;
}
s[j+1]='\0';
getch();
}
```

//write a c program to display the contents of a file.

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

 main()
{
```

```
char ch;
char str[20];
FILE *fp;
printf("Enter the source file name");
gets(str);
fp=fopen(str,"r");
if(fp==NULL)
{
printf("Error in opening file");
getch();
exit(1);
}
ch=getc(fp);
while(ch!=EOF)
{
putchar(ch);
ch=getc(fp);
}
fclose(fp);
getch();
}
```

// program ti merge two files into a third file

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
 main()
{
FILE *f1,*f2,*f3;
char str1[20],str2[20],str[20];
char ch;

printf("Enter First File Name to Merge:");
gets(str1);
f1=fopen(str1,"r");
if(f1==NULL)
{
printf("Error in opening First File");
getch();
exit(1);
}
printf("Enter Second File Name to Merge:");
gets(str2);
f2=fopen(str2,"r");
if(f2==NULL)
{
printf("Error in opening Second File");
getch();
exit(1);
}
printf("Enter Destination File Name:");
gets(str);
```

```
f3=fopen(str,"w");
if(f3==NULL)
{
printf("Error in opening Destination File");
getch();
exit(1);
}
ch=getc(f1);
while(ch!=EOF)
{
putc(ch,f3);
ch=getc(f1);
}
ch=getc(f2);
while(ch!=EOF)
{
putc(ch,f3);
ch=getc(f2);
}
fclose(f1);
fclose(f2);
fclose(f3);
printf("File Mergeing Successfully Completed");
getch();
}
```

// program to implement singly linked list

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

void create();
void insert();
void delet();
void display();
struct  node
{
int data;
struct node *link;
};
struct node *first=NULL,*last=NULL,*next,*prev,*cur;
void create()
{
  cur=(struct node*)malloc(sizeof(struct node));
  printf("\nENTER THE DATA: ");
  scanf("%d",&cur->data);
  cur->link=NULL;
  first=cur;
  last=cur;
}
void insert()
{
  int pos,c=1;
```

```
  cur=(struct node*)malloc(sizeof(struct node));
  printf("\nENTER THE DATA: ");
  scanf("%d",&cur->data);
  printf("\nENTER THE POSITION: ");
  scanf("%d",&pos);
  if((pos==1) &&(first!=NULL))
  {
   cur->link = first;
   first=cur;
  }
  else
  {
    next=first;
    while(c<pos)
    {
    prev=next;
    next=prev->link;
    c++;
    }
    if(prev==NULL)
    {
    printf("\nINVALID POSITION\n");
    }
    else
    {
    cur->link=prev->link;
    prev->link=cur;
    }
 }
}
void delet()
{
int pos,c=1;
printf("\nENTER THE POSITION : ");
scanf("%d",&pos);
if(first==NULL)
{
printf("\nLIST IS EMPTY\n");
}
else if(pos==1 && first->link==NULL)
{
printf("\n DELETED ELEMENT IS %d\n",first->data);
free(first);
first=NULL;
}
else if(pos==1 && first->link!=NULL)
{
cur=first;
first=first->link;
cur->link=NULL;
printf("\n DELETED ELEMENT IS %d\n",cur->data);
free(cur);
```

```
}
else
{
next=first;
while(c<pos)
{
cur=next;
next=next->link;
c++;
}
cur->link=next->link;
next->link=NULL;
if(next==NULL)
{
printf("\nINVALID POSITION\n");
}
else
{
printf("\n DELETED ELEMENT IS %d\n",next->data);
free(next);
}
}
}
void display()
{
  cur=first;
  printf( " the list is \n");
    while(cur!=NULL)
  {
    printf("\n %d",cur->data);
    cur=cur->link;
  }
}

main()
{
int ch;
printf("\n\nSINGLY LINKED LIST");
do
{
printf("\n\n1.CREATE\n2.INSERT\n3.DELETE\n4.DISPLAY\n5.EXIT\n");
printf("\n\nENTER YOUR CHOICE : ");
scanf("%d",&ch);
switch(ch)
{
case 1:
        create();
        break;
case 2:
        insert();
        break;
case 3:
```

```
        delet();
        break;
case 4:
     display();
     break;
case 5: exit(0);
break;
default:
   printf("Invalid choice...");
}
}while(1);
}
```

/* Write a C program that uses functions to perform the following operations on doubly linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal in both ways */

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

typedef struct dubll
{
 int data;
 struct dubll *leftlink,*rightlink;
}*DUBLL;

DUBLL high,temp_node,low,last,pntr;
int flag=0;

DUBLL NodeAlloc();
DUBLL Search(int,int);

void CreateItem();
void AppendItem();
void PrintItem();
void DeleteItem();
DUBLL Search(int item,int flag);
DUBLL NodeAlloc();
void InsertItem();
main(void)
{
 int choice,Item;
 high=NULL;
 while(1)
 {
  printf("\n \t\t\t***** M A I N   M E N U *****\n\n");
  printf("\n 1: Create Linked List \n 2: Append a Node to the List \n 3: Traverse the
List \n 4: Delete a Node from the List  \n 5: Search a Node \n 6: Insert a Node to the
List \n 7: Close \n\n\t\t Enter your Option [  ]\b\b");
  scanf("%d",&choice);
  switch(choice)
```

```
  {
    case 1:
         CreateItem();
         puts("\nPress any key to go back to main menu.");
           getch();
         break;
    case 2:
         AppendItem();
         break;
    case 3:
           PrintItem();
         puts("\nPress any key to go back to main menu.");
         getch();
           break;
    case 4:
         DeleteItem();
         break;
    case 5:
         printf("Find an Item: ");
         scanf("%d",&Item);
         temp_node=Search(Item,0);
         if(temp_node)
         {
            puts("The item is available in the Linked List.");
         }
         else
           {
          puts("The item is not found in the Linked List.");
          }
          getch();
            break;
    case 6:
         InsertItem();
         break;
    case 7:
         exit(0);
    default:
           puts("Invalid choice.");
           puts("\nPress any key to go back to main menu.");
             getch();
           break;
   }
  }
}

/* Function to Create the list*/
void CreateItem()
{
   if(high==NULL)
   {
    printf("\n --Creating the list--");
    temp_node=NodeAlloc();
```

```
    printf("\n Enter starting data (as integer value) :");
    scanf("%d",&temp_node->data);
    high=temp_node;
   }
   else{ printf("\n List already created @ %d with %d as data.",high,high->data);}
}

/* Function to Append items to the list*/
void AppendItem()
{
   low=high;
   if(high==NULL)
   {
     CreateItem();
   }
   else
   {
     temp_node=NodeAlloc();
     printf("\n Enter Item (in integer) :");
     scanf("%d",&temp_node->data);
     temp_node->rightlink=NULL;

     while(low->rightlink!=NULL)
      low=low->rightlink;
      low->rightlink=temp_node;
      temp_node->leftlink=low;
      last=low->rightlink;

    }
}

/* Function to Traverse the list both ways and print the data*/
void PrintItem()
{
   DUBLL temp_node;
   if(high==NULL)
   {
     printf("\n List is not available. Please create a list first.");
     getch();
     CreateItem();
   }
   temp_node=high;
   last=low->rightlink;
   printf("\n--Printing The List In Forward direction--\n");

   while(temp_node!=NULL)              //In forward direction
          {
           printf("\t %d",temp_node->data);
           temp_node = temp_node->rightlink;
          }
   printf("\n");
   printf("\n--Printing The List In Backward direction--\n");
```

```
            temp_node=high;
            if(temp_node->rightlink==NULL){printf("%d",temp_node->data);return; }
              while(last!=NULL)              //In backward direction
               {
                 printf("\t %d",last->data);
                 last = last->leftlink;
               }
}

/* Function to Delete items of the list*/
void DeleteItem()
{
 int value;
 DUBLL temp_node;
 if(high==NULL)
 {
   printf("\n List is not available. Please create a list first.");
   getch();
   CreateItem();
 }
 printf("\n Item to delete :");
 scanf("%d",&value);
 pntr=Search(value,1);
 pntr->leftlink->rightlink=pntr->rightlink;
 pntr->rightlink->leftlink=pntr->leftlink;
 temp_node=pntr;
 free(temp_node);
}

/* Function to Search an item from the list*/
DUBLL Search(int item,int flag)
{
   temp_node = high;
   if(high==NULL)
   {
     printf("\n List is not available. Please create a list first.");
     getch();
     CreateItem();
   }
   while(temp_node!=NULL)
   {
    if(temp_node->data==item )
    {
     if(flag==0)
     {
       return(1);
     }
     else
     {
      return(temp_node);
     }
    }
```

```
    temp_node=temp_node->rightlink;
   }
}

/* Function to Allocate nodes*/
DUBLL NodeAlloc()
{
  DUBLL tmep_node;
  tmep_node=malloc(sizeof(struct dubll));
   if(tmep_node==NULL)
     {
      printf("\n No memory available. Node allocation cannot be done.");
     }
     tmep_node->rightlink=tmep_node->leftlink=NULL;
  return(tmep_node);
 }

/* Function to Insert items in the middle of the list*/
void InsertItem()
 {
  int node;
  DUBLL temp_node;

  if(high==NULL)
  {
    printf("\n List is not available. Please create a list first.");
    getch();
    CreateItem();
  }
  temp_node=NodeAlloc();
  printf("Position At which node to be inserted: ___ & New Item Value: ___ ");
  scanf("%d",&node);
  scanf("%d",&temp_node->data);
  pntr=Search(node,1);

  if(pntr->rightlink==NULL){printf("\n The operation is not possible."); getch();return;}
     temp_node->leftlink=pntr;                  //creating link to new node
     temp_node->rightlink=pntr->rightlink;

     pntr->rightlink->leftlink=temp_node;
     pntr->rightlink=temp_node;

     printf("\n Item has been Inserted.");
     getch();
 }
```

// program to implement stack using arrays

```
#include<stdio.h>
#include<conio.h>
#define SIZE 5
int top=-1;
int s[SIZE];
```

```
void push(int);
int pop();
void display ();
 main()
{
int ch;
int item,y;
do
{
printf("\n\t MENU ");
printf("\n\t1.Push");
printf("\n\t2.Pop");
printf("\n\t3.Display.");
printf("\n\t4.Exit");
printf("\n Enter ur choice");
scanf ( "%d" , &ch) ;
switch(ch)
{
case 1 : printf("\n\t Enter the element: ");
          scanf("%d",& item );
          push(item);
          break;
case 2:  y=pop();
        printf("\n\t the deleted value %d" ,y);
        break;
case 3 : display();
          break;
case 4 : exit(0);
        break;
default: printf("Invalid Choice");
        break;
}
}while(ch<=4);
}
void push(int item )
{
if(top==SIZE-1)
printf("\t stack is overflow");
else
{
top++;
s[top]= item ;
}
}
int pop()
{
int y;
if(top==-1)
printf("\n\t the stack is underflow");
else
{
y=s[top];
```

```
top--;
}
return y;
}
void display()
{
int i;
for(i=top;i>=0;i--)
printf( "%d\t",s[i]);
}
```

/* Program to implement stack (its operations) using Pointers */

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

struct st_point
{
  int ele;
  struct st_point *l;
}*t;
int i;

void push_ele(int j);
int pop_ele();
void display_ele();

main()
{
  char choice,num1=0,num2=0;
  int i;
  while(1)
  {
    printf("=====================================");
    printf("\n\t\t MENU ");
    printf("\n=====================================");
    printf("\n[1] Using Push Function");
    printf("\n[2] Using Pop Function");
    printf("\n[3] Elements present in Stack");
    printf("\n[4] Exit\n");
    printf("\n\tEnter your choice: ");
    fflush(stdin);
    scanf("%c",&choice);

    switch(choice-'0')
    {
      case 1:
      {
    printf("\n\tElement to be pushed:");
    scanf("%d",&num1);
    push_ele(num1);
```

```
    break;
      }

      case 2:
      {
    num2=pop_ele();
    printf("\n\tElement to be popped: %d\n\t",num2);
    getch();
    break;
      }

      case 3:
      {
    printf("\n\tElements present in the stack are:\n\t");
    display_ele();
    getch();
    break;
      }

      case 4:
    exit(1);
    break;

      default:
    printf("\nYour choice is invalid.\n");
    break;
    }
  }
}

/*Inserting the elements using push function*/
void push_ele(int j)
{
  struct st_point *m;
  m=(struct st_point*)malloc(sizeof(struct st_point));
  m->ele=j;
  m->l=t;
  t=m;
  return;
}

/*Removing the elements using pop function*/
int pop_ele()
{
  if(t==NULL)
  {
    printf("\n\STACK is Empty.");
    getch();
    exit(1);
  }
  else
  {
```

```
    int i=t->ele;
    t=t->l;
    return (i);
  }
return 0;
}

/*Displaying the elements */
void display_ele()
{
  struct st_point *pointer=NULL;
  pointer=t;
  while(pointer!=NULL)
  {
    printf("%d\t",pointer->ele);
    pointer=pointer->l;
  }
}
```

// Program to implement queues using arrays

```
#include<stdio.h>
#include<conio.h>
#define SIZE 5
int r=-1;
int f=-1;
int q[SIZE];
void insert(int);
int del();
void display();
main ( )
{
int y,item,ch;

do
{
printf("\n\t MENU");
printf("\n\t 1.Insert.");
printf("\n\t 2.Delete.");
printf("\n\t 3.Display.");
printf("\n\t 4. Exit.\n");
printf("\n\t Enter ur choice");
scanf("%d",&ch) ;
switch(ch)
{
case 1:
printf("\n\t Enter the element to insert:");
scanf("%d",&item ) ;
insert(item) ;
break;
case 2:          y=del() ;
printf("\n\tThe deleted value:%d\n",y);
break;
```

```
case 3: display();
break;
case 4: exit(0);
default: printf("Invalid Choice\n");
        break;
}
}while(ch<=4) ;
getch () ;
}
void insert(int item)
{
if(r==SIZE-1)
{
printf("\n\t Queue is overflow\n");
return;
}
else
{
r++;
q[r]=item;
printf("\n\t The value is inserted\n");
if(f==-1)
f=0;
}
}
int del()
{
int y;
if(f==-1)
printf("\n\t Queue is Empty\n");
y=q[f];
if(f==r)
f=r=-1;
else
f++;
return y;
}
void display()
{
int i;
for (i=f;i<=r;i++)
printf("\t%d",q[i]);
getch();
}
```

/* Write C programs that implement Queue (its operations) using ii) Pointers */

```
#define true 1
#define false 0

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

```
#include<stdlib.h>

struct q_point
{
  int ele;
  struct q_point* n;
};

struct q_point *f_ptr = NULL;

int e_que(void);
void add_ele(int);
int rem_ele(void);
void show_ele();

/*main function*/
 main()
{
  int ele,choice,j;
  while(1)
  {

    printf("\n\n****IMPLEMENTATION OF QUEUE USING POINTERS****\n");
    printf("================================================");
    printf("\n\t\t  MENU\n");
    printf("================================================");
    printf("\n\t[1] To insert an element");
    printf("\n\t[2] To remove an element");
    printf("\n\t[3] To display all the elements");
    printf("\n\t[4] Exit");
    printf("\n\n\tEnter your choice:");
    scanf("%d", &choice);

    switch(choice)
    {
      case 1:
      {
          printf("\n\tElement to be inserted:");
          scanf("%d",&ele);
          add_ele(ele);
          getch();
          break;
      }

      case 2:
      {
          if(!e_que())
          {
            j=rem_ele();
            printf("\n\t%d is removed from the queue",j);
            getch();
          }
```

```
            else
            {
              printf("\n\tQueue is Empty.");
              getch();
            }
            break;
       }

       case 3:
            show_ele();
            getch();
            break;

       case 4:
            exit(1);
            break;

       default:
            printf("\n\tInvalid choice.");
            getch();
            break;
     }

  }
}

/* Function to check if the queue is empty*/
int e_que(void)
{
  if(f_ptr==NULL)
  return true;
  return false;
}

/* Function to add an element to the queue*/
void add_ele(int ele)
{
  struct q_point *queue = (struct q_point*)malloc(sizeof(struct q_point));
  queue->ele = ele;
  queue->n = NULL;
  if(f_ptr==NULL)
    f_ptr = queue;
  else
  {
    struct q_point* ptr;
    ptr = f_ptr;
    for(ptr=f_ptr ;ptr->n!=NULL; ptr=ptr->n);
      ptr->n = queue;
  }
}

/* Function to remove an element from the queue*/
```

```
int rem_ele()
{
  struct q_point* queue=NULL;
  if(e_que()==false)
  {
    int j = f_ptr->ele;
    queue=f_ptr;
    f_ptr = f_ptr->n;
    free (queue);
    return j;
  }
  else
  {
    printf("\n\tQueue is empty.");
    return -9999;
  }
}

/* Function to display the queue*/
void show_ele()
{
  struct q_point *ptr=NULL;
  ptr=f_ptr;
  if(e_que())
  {
    printf("\n\tQUEUE is Empty.");
    return;
  }
  else
  {
    printf("\n\tElements present in Queue are:\n\t");
    while(ptr!=NULL)
    {
      printf("%d\t",ptr->ele);
      ptr=ptr->n;
    }
  }
}
```

//INFIX TO POSTFIX

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

int st[100];
int st_top=-1;

int cal(char post[]);
void in_post(char in[]);
```

```
void push_item(int it);
int pop_item();
int st_ISP(char t);
int st_ICP(char t);

/*main function*/
 main()
{
  char in[100],post[100];
  printf("\n\tEnter the Infix Expression: ");
  gets(in);
  in_post(in);
  getch();
}
/*end main*/

void push_item(int it)
{
  if(st_top==99)
  {
    printf("\n\n\t*STACK is Full*");
    getch();
    exit(1);
  }
  st[++st_top]=it;
}

int pop_item()
{
  int it;
  if(st_top==-1)
  {
    getch();
  }
  return(st[st_top--]);
}

/*Function for converting an infix expression to a postfix expression. */
void in_post(char in[])
{
  int x=0,y=0,z,result=0;
  char a,c, post[100];
  char t;
  push_item('\0');
  t=in[x];
  while(t!='\0')
  {
    if(isalnum(t))
    /*For checking whether the value in t is an alphabet or number. */
    {
      post[y]=t;
      y++;
```

```
    }
    else if(t=='(')
    {
      push_item('(');
    }
    else if(t==')')
    {
      while(st[st_top]!='(')
      {
          c=pop_item();
          post[y]=c;
          y++;
      }
    c=pop_item();
    }
    else
    {
      while(st_ISP(st[st_top])>=st_ICP(t))
      {
          c=pop_item();
          post[y]=c;
          y++;
      }
      push_item(t);
    }
    x++;
    t=in[x];
  }

  while(st_top!=-1)
  {
    c=pop_item();
    post[y]=c;
    y++;
  }
  printf("\n\tThe Postfix Expression is:");

  for(z=0;z<y;z++)
    printf("%c",post[z]);
  printf("\n\nDo you want to evaluate the Result of Postfix Expression?(Y/N):");
  scanf("%c",&a);
  if(a=='y' || a=='Y')
  {
    result=cal(post);
    printf("\n\n\tResult is: %d\n",result);
    getch();
  }
  else if(a=='n' || a=='N')
  {
    exit(0);
  }
}
```

```
/*Determining priority of inside elements*/
int st_ISP(char t)
{
  switch(t)
  {
    case '(':return (10);
    case ')':return (9);
    case '+':return (7);
    case '-':return (7);
    case '*':return (8);
    case '/':return (8);
    case '\0':return (0);
    default: printf("Expression is invalid.");
    break;
  }
  return 0;
}

/*Determining priority of approaching elements*/
int st_ICP(char t)
{
  switch(t)
  {

    case '(':return (10);
    case ')':return (9);
    case '+':return (7);
    case '-':return (7);
    case '*':return (8);
    case '/':return (8);
    case '\0':return (0);
    default: printf("Expression is invalid.");
    break;
  }
  return 0;
}

/*Evaluating the result of postfix expression*/
int cal(char post[])
{
  int m,n,x,y,j=0,len;
  len=strlen(post);
  while(j<len)
  {
    if(isdigit(post[j]))
    {
      x=post[j]-'0';
      push_item(x);
    }
    else
    {
```

```
      m=pop_item();
      n=pop_item();

      switch(post[j])
      {
          case '+':x=n+m;
          break;
          case '-':x=n-m;
          break;
          case '*':x=n*m;
          break;
          case '/':x=n/m;
          break;
      }
      push_item(x);
    }
    j++;
  }
  if(st_top>0)
  {
    printf("Number of Operands are more than Operators.");
    exit(0);
  }
  else
  {
    y=pop_item();
    return (y);
  }
  return 0;
}
```

//bubble sort

```
#include<stdio.h>
#include<conio.h>
 main()
{
int i,j,temp,n,a[20];
printf("Enter the number of elements:");
scanf("%d",&n);
printf("\nEnter the elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n-1;i++)
for(j=0;j<n-i-1;j++)
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
printf("\nElements after sorting:");
for(i=0;i<n;i++)
```

```
printf("\n%d",a[i]);
getch();
}
/* selection sort */
#include<stdio.h>
#include<conio.h>
main()
{
int i,j,temp,n,a[20];
printf("Enter the number of elements:");
scanf("%d",&n);
printf("\nEnter the elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n-1;i++)
for(j=i+1;j<n;j++)
if(a[i]>a[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
printf("\nElements after sorting:");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
getch();
}
```

/* Write C programs that use both recursive and non recursive functions to perform the following searching operation for a Key value in a given list of integers :
i) Linear search */

```
#include <stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX_LEN 10

void l_search_recursive(int l[],int num,int ele);
void l_search_nonrecursive(int l[],int num,int ele);
void l_search(int l[],int num,int ele);
void read_list(int l[],int num);
void print_list(int l[],int num);

 main()
{
    int l[MAX_LEN], num, ele;
    int ch;
    printf("=====================================================");
    printf("\n\t\t\tMENU");
```

```
    printf("\n=====================================================");
    printf("\n[1] Linary Search using Recursion method");
    printf("\n[2] Linary Search using Non-Recursion method");
    printf("\n\nEnter your Choice:");
    scanf("%d",&ch);

    if(ch<=2 & ch>0)
    {
     printf("Enter the number of elements :");
     scanf("%d",&num);
     read_list(l,num);
     printf("\nElements present in the list are:\n\n");
     print_list(l,num);
     printf("\n\nElement you want to search:\n\n");
     scanf("%d",&ele);

     switch(ch)
     {
       case 1:printf("\n**Recursion method**\n");
              l_search_recursive(l,num,ele);
              getch();
              break;

       case 2:printf("\n**Non-Recursion method**\n");
               l_search_nonrecursive(l,num,ele);
               getch();
               break;
     }
     }
     else
     printf("\ninvalid choice\n");

    getch();
}
/*end main*/

/* Non-Recursive method*/
void l_search_nonrecursive(int l[],int num,int ele)
{
   int j, f=0;
   for(j=0;j<num;j++)
   if( l[j] == ele)
   {
      printf("\nThe element %d is present at position %d in list\n",ele,j);
      f=1;
      break;
   }
   if(f==0)
       printf("\nThe element is %d is not present in the list\n",ele);
}

/* Recursive method*/
```

```
void l_search_recursive(int l[],int num,int ele)
{
   int f = 0;

   if( l[num] == ele)
   {
      printf("\nThe element %d is present at position %d in list\n",ele,num);
      getch();
      exit(1);
      f=1;
   }
   else
   {
    if((num==0) && (f==0))
    {
     printf("The element %d is not found.",ele);
     getch();
     exit(2);
    }
    else
    {
     l_search_recursive(l,num-1,ele);
    }
   }
   getch();
}

void read_list(int l[],int num)
{
   int j;
   printf("\nEnter the elements:\n");
   for(j=0;j<num;j++)
       scanf("%d",&l[j]);
}

void print_list(int l[],int num)
{
   int j;
   for(j=0;j<num;j++)
       printf("%d\t",l[j]);
}
```

/* Write C programs that use both recursive and non recursive functions to perform the following searching operations for a Key value in a given list of integers : ii) Binary search*/

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

```
#define MAX_LEN 10

/* Non-Recursive function*/
void b_search_nonrecursive(int l[],int num,int ele)
{
   int l1,i,j, flag = 0;
   l1 = 0;
   i = num-1;
   while(l1 <= i)
   {
      j = (l1+i)/2;
      if( l[j] == ele)
      {
          printf("\nThe element %d is present at position %d in list\n",ele,j);
                    flag =1;
                    break;
      }
      else
              if(l[j] < ele)
                       l1 = j+1;
              else
                       i = j-1;
   }
   if( flag == 0)
   printf("\nThe element %d is not present in the list\n",ele);
}

/* Recursive function*/
int b_search_recursive(int l[],int arrayStart,int arrayEnd,int a)
{
  int m,pos;
  if (arrayStart<=arrayEnd)
  {
    m=(arrayStart+arrayEnd)/2;
    if (l[m]==a)
      return m;
    else if (a<l[m])
      return b_search_recursive(l,arrayStart,m-1,a);
    else
      return b_search_recursive(l,m+1,arrayEnd,a);
   }
   return -1;
}

void read_list(int l[],int n)
{
   int i;
   printf("\nEnter the elements:\n");
   for(i=0;i<n;i++)
       scanf("%d",&l[i]);
}
```

```
void print_list(int l[],int n)
{
    int i;
   for(i=0;i<n;i++)
       printf("%d\t",l[i]);
}

/*main function*/
main()
{
   int l[MAX_LEN], num, ele,f,l1,a;
   int ch,pos;
   printf("=======================================================");
   printf("\n\t\t\tMENU");
   printf("\n=======================================================");
   printf("\n[1] Binary Search using Recursion method");
   printf("\n[2] Binary Search using Non-Recursion method");
   printf("\n\nEnter your Choice:");
   scanf("%d",&ch);

   if(ch<=2 & ch>0)
   {
     printf("\nEnter the number of elements : ");
     scanf("%d",&num);
     read_list(l,num);
     printf("\nElements present in the list are:\n\n");
     print_list(l,num);
     printf("\n\nEnter the element you want to search:\n\n");
     scanf("%d",&ele);


   switch(ch)
   {
     case 1:printf("\nRecursive method:\n");
              pos=b_search_recursive(l,0,num,ele);
              if(pos==-1)
              {
                    printf("Element is not found");
              }
              else
              {
                    printf("Element is found at %d position",pos);
              }
              getch();
              break;

     case 2:printf("\nNon-Recursive method:\n");
              b_search_nonrecursive(l,num,ele);
              getch();
              break;
     }
   }
```

```
getch();
}
```

//quicksort

```
#include <stdio.h>
#include <conio.h>
swap(int *a,int *b)
{
 int temp;
 temp=*a;
 *a=*b;
 *b=temp;
 return;
}
quicksort(int a[10] ,int lb,int ub)
{
int i=lb,j=ub,key=lb;
if (lb<ub)
{
while (i<j)
{
while((a[i]<a[key])&&(lb<ub))
i++;
while(a[j]>a[key])
j--;
if(i<j)
swap(&a[i] ,&a[j]);
}
                        swap(&a[j],&a[key]);
quicksort(a,lb,j-1) ;
quicksort (a,j+1,ub) ;
}
return;
}
//MAIN PROGRAM STARTS
main ()
{
int i,n,a[100];
printf("\n\t How many numbers do u want to enter?");
scanf("%d",&n) ;
printf("\n\t Enter the numbers:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
quicksort(a,0,n-1);
printf("\n\t  Numbers after sorting:");
for(i=0;i<n;i++)
```

```
printf("%d ",a[i]);
getch () ;
}
```

//MERGE SORT

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int a[20],n;
void merge_sort(int x[], int end, int start);
 main()
{
int j = 0;

printf("\n\t How many numbers do u want to enter?");
scanf("%d",&n) ;
printf("\n\nEnter the elements to be sorted: \n");
for(j=0;j<n;j++)
scanf("%d",&a[j]);
merge_sort(a,0,n-1);
printf("After Merge Sort :");
 for(j = 0; j < n; j++)
 printf(" %d", a[j]);
 getch();
}
void merge_sort(int x[], int end, int start)
{
int j = 0;
const int size = start - end + 1;
int mid  = 0;
int mrg1 = 0;
int mrg2 = 0;
int executing[20];
if(end == start)
return;
mid  = (end + start) / 2;
merge_sort(x, end, mid);
merge_sort(x, mid + 1, start);
for(j = 0; j < size; j++)
executing[j] = x[end + j];
mrg1 = 0;
mrg2 = mid - end + 1;
for(j = 0; j < size; j++)
{
if(mrg2 <= start - end)
if(mrg1 <= mid - end)
if(executing[mrg1] > executing[mrg2])
     x[j + end] = executing[mrg2++];
else
     x[j + end] = executing[mrg1++];
else
    x[j + end] = executing[mrg2++];
```

```
else
   x[j + end] = executing[mrg1++];
}
}
```