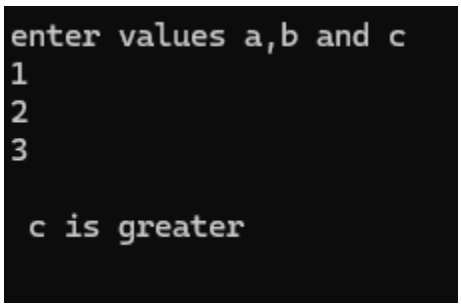
Subject & code : Data Structures using C Lab (24MCA112)

CONTROL STRUCTURES

1a) To implement a C program to find Greatest among three numbers.

```
#include<stdio.h>
#include<conio.h>
main()
{

int a,b,c;
printf("Enter values a,b and c");
scanf("%d%d%d",&a,&b,&c);

if(a>b&&a>c)
    {
        printf("\n a is greater");
     }
else if(b>a&&b>c)
     {
       printf("\n b is greater");
      }
else
      {
printf("\n c is greater");
       }
getch();
}
```

Output:



1b) To implement a C program for problem solving using branching and logical expressions:

```
#include<stdio.h>
int main()
{
int marks;
printf("enter your marks?\n");
scanf("%d",&marks);
if(marks>85&marks<=100)
        {
                printf("congrats! your scored A..");
        }
else if(marks>60&&marks<=85)
        {
                printf("you scored grade B+...");
        }
else if(marks>40&&marks<=60)
        {
                printf("you scored B");
        }
else if(marks>30&&marks<=40)
        {
              printf("you scored grade C...");
```

```
        }
else
        {
                printf("Sorry you are fail...");
        }
}
```

OUTPUT:

```
enter your marks?
100
congrats! your scored A..
```

1c) To implement a C program to find Factorial of a number using while loop.

```
#include<stdio.h>

int main()

{

int n,i,f;

f=i=1;

printf("enter a number to find factorial:");

scanf("%d",&n);

while(i<=n)

        {

                f*=i;

                i++;

        }

printf("The factorial of %d is:%d",n,f);

return 0;

}
```

Output:

```
enter a number to find factorial:3
The factorial of 3 is:6
```

1d) To implement a C program for Arithmetic operations using switch case.

```
#include<stdio.h>
#include<conio.h>
main()
{
int a,b;
int op;
printf("\t\t\t ARITHMETIC OPERATIONS\n");
printf("1.Addition\n2.Sustraction\n3.Multiplication\n4.Division\n5.Exit\n");
printf("\nEnter the values of a and b \n");
scanf("%d%d",&a,&b);
do
        {
                printf("\n\nEnter your choice:");
                scanf("%d",&op);
                 switch(op)
        {
                case 1:
                printf("\n Sum of %d and %d is %d",a,b,a+b);
                break;
                case 2:
                 printf("\n Difference of %d and %d is:%d",a,b,a-b);
```

```
                    break;

                    case 3:

                    printf("\n Division of %d and %d is %d",a,b,a/b);

                    break;

                     case 4:

                     printf("\n multiplication of %d and %d is %d",a,b,a*b);

                     break;

                     case 5:

                     printf("\n Exit");

                     break;

          }

           }

while(op<5);

}
```

Output:

```
                        ARITHMETIC OPERATIONS
1.Addition
2.Sustraction
3.Multiplication
4.Division
5.Exit

Enter the values of a and b
4
5


Enter your choice:1

 Sum of 4 and 5 is 9
```

1e) To implement a C program to display pattern.

Program:

```
#include<stdio.h>

#include<conio.h>

main()

{

int i,j,n;

printf("enter value for n:");

scanf("%d",&n);

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

{

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

                {

                        printf("%d\t",j);

                }

printf("\n");

}

}
```

OUTPUT:

```
enter value for n:5
1
1       2
1       2       3
1       2       3       4
1       2       3       4       5
```

RECURSION

2a) To implement C program to find the factorial using Recursion.

```
#include<stdio.h>

int fact(int n)

{

        if(n==1)

        return 1;

        else

        return n*fact(n-1);

}

int main()

{

         int n;

         printf("Enter the number:");

         scanf("%d",&n);

         printf("Factorial of the given number %d is %d",n,fact(n));
```

```
}
```

OUTPUT:

```
Enter the number:5
Factorial of the given number 5 is 120
```

2b) To implement a c program to find the sum of natural numbers

```
#include<stdio.h>

int sum(int n);

int main()

{

    int num,result;

    printf("Enter a positive integer:");

    scanf("%d",&num);

    result=sum(num);

    printf("sum=%d",result);

    return 0;

}

int sum(int n)

    {

        if(n!=0)

        return n+sum(n-1);

        else

        return n;
```

```
}
```

OUTPUT:

```
Enter a positive integer:5
sum=15
```

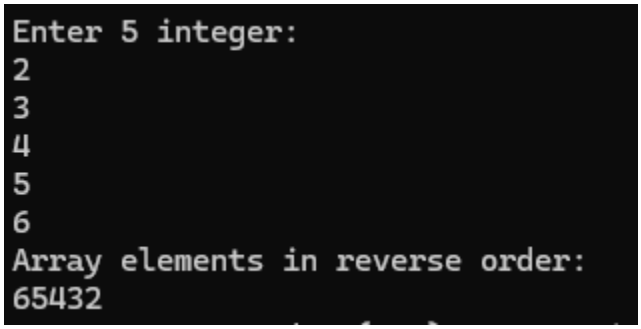
ARRAYS

3a) To implement a C- Program to print arrays in reverse order.

```
#include<stdio.h>

int main()

{

        int numbers[5];

        int size=5;

        printf("Enter 5 integer:\n");

for(int i=0; i<size; i++)

          {

              scanf("%d",&numbers[i]);

            }

printf("Array elements in reverse order:\n");

for(int i=size-1; i>=0; i--)

          {

               printf("%d",numbers[i]);
```

```
        }
printf("\t");
return 0;
}
```

OUTPUT:



3b) To implement a C-program to copy one Array to another

```
#include<stdio.h>
int main()
{
        int source[5],destination[5];
        int size=5;
        printf("Enter 5 integers for the source array:\n");
for(int i=0;i<size;i++)
        {
            scanf("%d",&source[i]);
```

```
        }
   for(int i=0;i<size;i++)
          {
              destination[i]=source[i];
           }
printf("Elements of the destination array:\n");
       for(int i=0;i<size;i++)
             {
                  printf("%d",destination[i]);
              }
  printf("\n");
  return 0;
  }
```

OUTPUT:

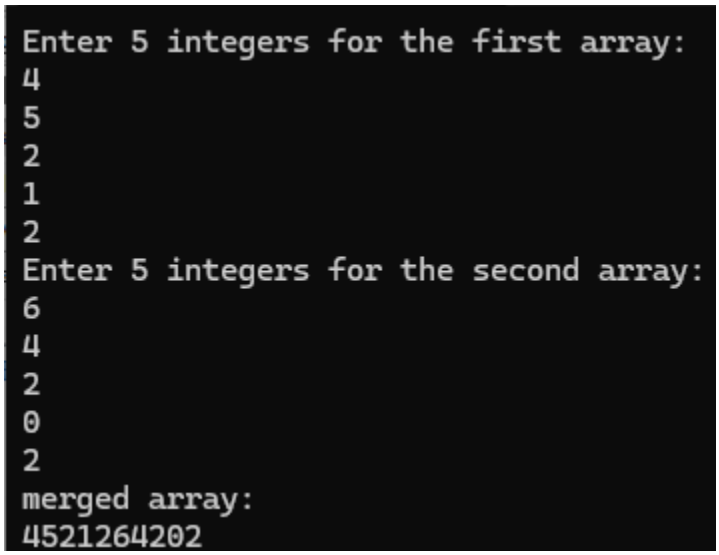
```
Enter 5 integers for the source array:
3
4
5
6
7
Elements of the destination array:
34567
```

3c) To implement a C-program to merge two Arrays.

```
#include<stdio.h>

int main()

{

       int arr1[5],arr2[5],merged[10];

       int size1=5,size2=5;

       printf("Enter 5 integers for the first array:\n");

       for(int i=0;i<size1;i++){

       scanf("%d",&arr1[i]);

 }

 printf("Enter 5 integers for the second array:\n");

          for(int i=0;i<size2;i++)

               {

                    scanf("%d",&arr2[i]);

                }

            for(int i=0;i<size1;i++)

                {

                     merged[i]=arr1[i];

                }

            for(int i=0;i<size2;i++)

                 {

                       merged[size1+i]=arr2[i];
```

```
            }

 printf("merged array:\n");

        for(int i=0;i<size1+size2;i++)

            {

                printf("%d",merged[i]);

            }

printf("\n");

return 0;

}
```

OUTPUT:



MATRIX OPERATIONS

4a) To implement a C-program for matrix manipulation Addition and Subtraction.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[3][3],b[3][3],c[3][3],i,j;
printf("\t\t\t MATRIX MANIPULATION \n");
printf("\n Enter values for Matrix A(3*3):\n");
for(i=0;i<3;i++)
{
 for(j=0;j<3;j++)
 {
  scanf("%d",&a[i][j]);
 }
}
printf("\n Enter values for Matrix B(3*3):\n");
for(i=0;i<3;i++)
{
 for(j=0;j<3;j++)
 {
  scanf("%d",&b[i][j]);
```

```
 }
}
 for(i=0;i<3;i++)
 {
    for(j=0;j<3;j++)
 {
   c[i][j]=a[i][j]+b[i][j];
 }
 }
 printf("\n\n Result of Matrix Addition:\n");
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
  {
     printf("%d\t",c[i][j]);
  }
  printf("\n");
  }
  for(i=0;i<3;i++)
  {
     for(j=0;j<3;j++)
     {
        c[i][j]=a[i][j]-b[i][j];
```

```
    }
  }
  printf("\n\n Result of Matrix Subtraction:\n");
  for(i=0;i<3;i++)
  {
   for(j=0;j<3;j++)
   {
    printf("%d\t",c[i][j]);
   }
   printf("\n");
  }
  getch();
}
```

OUTPUT:

```
                          MATRIX MANIPULATION

 Enter values for Matrix A(3*3):
9 8 7
6 5 4
3 2 1

 Enter values for Matrix B(3*3):
1 2 3
4 5 6
7 8 9


 Result of Matrix Addition:
10      10      10
10      10      10
10      10      10


 Result of Matrix Subtraction:
8       6       4
2       0       -2
-4      -6      -8
```

4b)To implement a C-program for matrix multiplication.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
 printf("\t\t\t MATRIX MANIPULATION-MULTIPLICATION\n");
 printf("Enter the number of rows:");
 scanf("%d",&r);
 printf("Enter the number of columns:");
 scanf("%d",&c);
 printf("\n Enter the elements of first matrix\n");
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
  {
   scanf("%d",&a[i][j]);
  }
 }
 printf("\n Enter the elements of second matrix\n");
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
```

```
{
 scanf("%d",&b[i][j]);
 }
}
printf("\n\n Result of Matrix Multiplication \n");
for(i=0;i<r;i++)
{
 for(j=0;j<c;j++)
 {
  mul[i][j]=0;
  for(k=0;k<c;k++)
  {
     mul[i][j]+=a[i][k]*b[k][j];
  }
 }
}
for(i=0;i<r;i++)
{
 for(j=0;j<c;j++)
 {
   printf("%d\t",mul[i][j]);
 }
 printf("\n");
```

```
}

}
```

OUTPUT:

```
                        MATRIX MANIPULATION-MULTIPLICATION
Enter the number of rows:3
Enter the number of columns:3

 Enter the elements of first matrix
1 2 1
2 1 2
1 2 1

 Enter the elements of second matrix
2 1 2
1 2 1
2 1 2


 Result of Matrix Multiplication
6       6       6
9       6       9
6       6       6
```

STRUCTURES

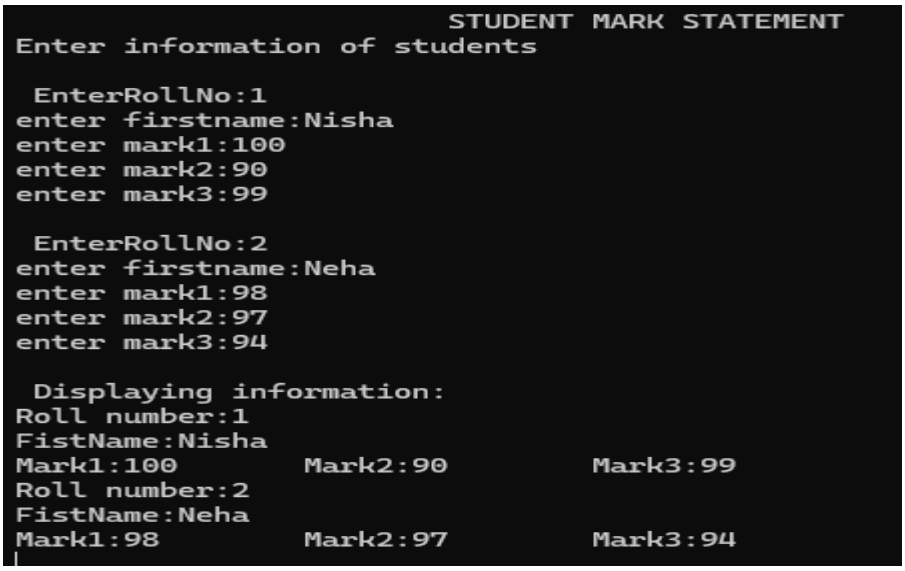
5a) To implement a C-program to display Students Mark Statement using Array Structures.

```
#include<stdio.h>

#include<conio.h>

struct student

{

int roll;

char firstname[15];

int mark1;
```

```
int mark2;
int mark3;
}
s[10];
void main()
{
int i;
printf("\t\t\tSTUDENT MARK STATEMENT\n");
printf("Enter information of students\n");
for(i=0;i<2;i++)
{
   s[i].roll=i+1;
   printf("\n EnterRollNo:%d\n",s[i].roll);
   printf("enter firstname:");
   scanf("%s",&s[i].firstname);
   printf("enter mark1:");
   scanf("%d",&s[i].mark1);
   printf("enter mark2:");
   scanf("%d",&s[i].mark2);
   printf("enter mark3:");
   scanf("%d",&s[i].mark3);
}
   printf("\n Displaying information:\n");
```

```
  for(i=0;i<2;i++)
  {
    printf("Roll number:%d\n",i+1);
     printf("FistName:");
     puts(s[i].firstname);
     printf("Mark1:%d\t",s[i].mark1);
     printf("Mark2:%d\t",s[i].mark2);
     printf("Mark3:%d\t",s[i].mark3);
     printf("\n");
  }
  getch();
}
```

OUTPUT:



5b) To implement a C-program to display student information using pointer structure

```
#include<stdio.h>
#include<string.h>
struct Student{
   char name[50];
   int age;
   float marks;
   };
int main(){
   struct Student student1, *ptr;
   ptr=&student1;
   printf("Enter student's name:");
   scanf("%[^\n]s",&ptr->name);
   printf("Enter student's age:");
   scanf("%d",&ptr->age);
   printf("Enter student's marks:");
   scanf("%f",&ptr->marks);
   //Displaying student information using pointer
   printf("\nStudent Information:\n");
   printf("Name:%s\n",ptr->name);
   printf("Age:%d\n",ptr->age);
   printf("Marks:%.2f\n",ptr->marks);
```

```
 return 0;

}
```

OUTPUT:

```
Enter student's name:Lavanya
Enter student's age:25
Enter student's marks:90

Student Information:
Name:Lavanya
Age:25
Marks:90.00
```

UNION

6a)Display Employee information using Union Program

```
#include<stdio.h>

#include<string.h>

//Define a union for Employee

union Employee{

   int id;

   char name[50];

   float salary;

   };

   int main(){

   union Employee emp;

   int choice;

   printf("select the information you want to input for the Employee:\n");
```

```
printf(" 1.Id\n 2.Name\n 3.Salary\n");

printf("Enter your choice(1/2/3):");

scanf("%d",&choice);

switch (choice){

   case 1:

   printf("Enter Employee ID:");

   scanf("%d",&emp.id);

   printf("Employee ID:%d\n",emp.id);

   break;

   case 2:

   printf("Enter Employee Name:");

   scanf("%s",emp.name);

   printf("Employee Name:%s\n",emp.name);

   break;

   case 3:

   printf("Enter Employee Salary:");

   scanf("%f",&emp.salary);

   printf("Employee Salary:%.2f\n",emp.salary);

   break;

   default:

      printf("Invalid choice!\n");

      break;

}
```

```
 return 0;

}
```

OUTPUT:

```
select the information you want to input for the Employee:
1.Id
 2.Name
 3.Salary
Enter your choice(1/2/3):3
Enter Employee Salary:4000
Employee Salary:4000.00
```

POINTERS

7a) To implement a C program to find sum of 10 array in pointers

```
#include<stdio.h>

#include<malloc.h>

void main()

{

int  i, n, sum=0;

int *a;

printf("Enter the size of array A\n");

scanf("%d",&n);

a=(int *)malloc(n * sizeof(int));

printf("Enter Element of the List\n");

for(i=0;i<n;i++)

{
```

```
scanf("%d",a+i);

}

for(i=0;i<n;i++)

{

sum=sum + *(a+i);

}

printf("sum of all elements in array=%d\n",sum);

return 0;

}
```

OUTPUT:

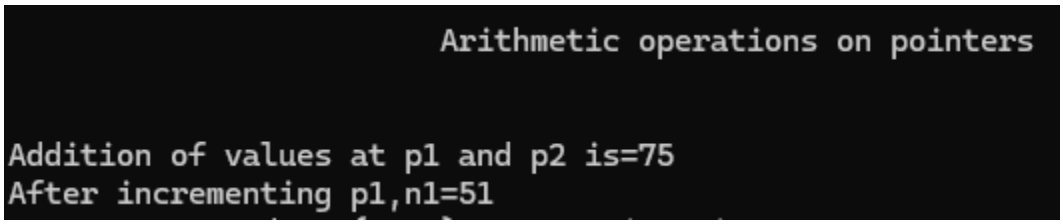
```
Enter the size of array A
5
Enter Element of the List
4
9
10
56
100
sum of all elements in array=179
```

7b) To implement a C program to demonstrate the arithmetic operation using pointers

```
#include<stdio.h>

#include<conio.h>

void main()

{

 int n1=50,n2=25,res=0;

int *p1,*p2;
```

```
p1=&n1;

p2=&n2;

printf("\t\t\t Arithmetic operations on pointers\n\n");

res=*p1+*p2;

printf("\n Addition of values at p1 and p2 is=%d",res);

++*p1;

printf("\n After incrementing p1,n1=%d",n1);

}
```

OUTPUT:



7c) To implement a C-program use pointers to read on an array of integer and print its element in reverse order

```
#include<stdio.h>

int main()

{

 int arr[5]={1,2,3,4,5};

 int *ptr1,*ptr2,temp;

 printf("Original array:");

 for(int i=0;i<5;i++)

{
```

```
 printf("%d",arr[i]);

}

ptr1=arr;

ptr2=arr+4;

while(ptr1<ptr2)

{

  temp=*ptr1;

  *ptr1=*ptr2;

  *ptr2=temp;

  ptr1++;

  ptr2--;

  }

  printf("\nReversed array:");

  for(int i=0;i<5;i++){

  printf("%d",arr[i]);

  }

  return 0;

}
```

OUTPUT:

```
Original array:12345
Reversed array:54321
```

Files

8a) To implement a C program to open a file, write in it, and close the file

```
#include<stdio.h>
#include<string.h>
int main()
{
 FILE*fp;
 char filetext[50]="This is the simple text";
 fp=fopen("gsTest.txt","w");
 if(fp==NULL)
 {
 printf("\n File Does not exist");
 }
 else
 {
  printf("The file is now opened\n");
 if(strlen(filetext)>0)
 {
 fputs(filetext,fp);
 fputs("\n",fp);
 }
 fclose(fp);
```

```
 printf("Data successfully written in file gsTest.txt\n");

 printf("The file is now closed");

}

return 0;

}
```

```
The file is now opened
Data successfully written in file gsTest.txt
The file is now closed
```

8b) Write a C program to open a file , read from it, and close the file

```
#include<stdio.h>

#include<string.h>

int main()

{

 FILE*fp;

 char readdata[50];

 fp=fopen("gsTest.txt","r");

 if(fp==NULL)

 {

  printf("gsTest.txt failed to open.");

}

else

{
```

```
printf("The file is now opened.\n");

while(fgets(readdata,50,fp)!=NULL)

{

printf("%s",readdata);

}

fclose(fp);

printf("Data successfully read from file gsTest.txt\n");

printf("The file is now closed.");

}

return 0;

}
```

OUTPUT:

```
The file is now opened.
This is the simple text
Data successfully read from file gsTest.txt
The file is now closed.
```

LINKED LISTS

9a) To implement a C-program for Single Linked List and its operations.

```
#include<stdio.h>
#include<stdlib.h>
 void creation();
 void insert_front();
 void insert_end();
 void insert_any();
 void delete_front();
 void delete_end();
 void delete_any();
 void display();
 void traverse();
 void exit();
 int x,key;
  struct node
  {
  int data;
  struct node*link;
  }*ptr,*ptr1,*header,*new;
  void main()
  {
  int ch;
  while(1)
  {
  printf("\n 1.creation\n 2.insert at front\n 3.insert at end\n 4.insert at any\n 5.delete at front\n
6.delete at end\n 7.delete at any\n 8.display\n 9.traverse\n 10.exit\n");
  printf("Enter the choice:");
  scanf("%d",&ch);
   switch(ch)
     {
     case 1:creation();
          break;
     case 2:insert_front();
          break;
     case 3:insert_end();
          break;
```

```
  case 4:insert_any();
        break;
  case 5:delete_front();
        break;
  case 6:delete_end();
        break;
  case 7:delete_any();
        break;
  case 8:display();
        break;
  case 9:traverse();
        break;
  case 10:exit(0);
         break;
  default:
     printf("Enter valid choice");
  }
}
}
void creation()
{
 header=(struct node*)malloc(sizeof(struct node));
 header->data=NULL;
 header->link=NULL;
 printf("\n Empty header is created");
}
void insert_front()
{
 new=(struct node*)malloc(sizeof(struct node));
 if(new==NULL)
 {
    printf("Insertion not possible");
 }
 else
 {
    printf("Enter data to insert:");
    scanf("%d",&x);
    new->data=x;
    new->link=header;
    header=new;
    printf("\n Node inserted");
```

```
  }
}
void insert_end()
{
   new=(struct node*)malloc(sizeof(struct node));
   if(new==NULL)
   {
      printf("Insertion not possible");
   }
   else
   {
      printf("Enter the data:");
      scanf("%d",&x);
      new->data=x;
      if(header==NULL)
      {
         new->link=NULL;
         header=new;
         printf("\n Node inserted");
      }
      else
      {
         ptr=header;
         while(ptr->link!=NULL)
         {
            ptr=ptr->link;
         }
         ptr->link=new;
         new->link=NULL;
         printf("\n Node inserted");
      }
   }
}
void insert_any()
{
   new=(struct node*)malloc(sizeof(struct node));
   if(new==NULL)
   {
      printf("Insertion is not possible");
   }
   else
```

```
    {
        printf("\n Enter the Element:");
        scanf("%d",&x);
        new->data=x;
        printf("Enter the key value after you want to insert:");
        scanf("%d",&key);
        ptr=header;
        while((ptr->link!=NULL)&&(ptr->data!=key))
        {
            ptr=ptr->link;
            if(ptr==NULL)
            {
                printf("\n Can't insert");
            }
        }
        new->link=ptr->link;
        ptr->link=new;
        printf("\n node inserted");
    }
}
void delete_front()
{
    if(header==NULL)
    {
        printf("\n list empty,no deletion");
    }
    else
    {
        ptr=header;
        header=ptr->link;
        free(ptr);
        printf("\n Node deleted from the beginning..\n");
    }
}
void delete_end()
{
    if(header==NULL)
    {
        printf("List is empty");
    }
    else if(header->link==NULL)
```

```
    {
       header=NULL;
       free(header);
       printf("\n Only node of the list is deleted");
    }
    else
    {
       ptr=header;
       while(ptr->link!=NULL)
       {
          ptr1=ptr;
          ptr=ptr->link;
       }
       ptr1->link=NULL;
       free(ptr);
       printf("\n Deleted Node from the last...\n");
    }
}
void delete_any()
{
    printf("Enter the key element to delete:");
    scanf("%d",&key);
    ptr=header;
    while((ptr->link!=NULL)&&(ptr->data!=key))
    {
       ptr1=ptr;
       ptr=ptr->link;
       if(ptr==NULL)
       {
          printf("\n Can't delete");
       }
    }
    x=ptr->data;
    ptr1->link=ptr->link;
    free(ptr);
    printf("\n Deleted %d",x);
}
void display()
{
    ptr=header;
    if(ptr==NULL)
```

```
    {
        printf("Nothing to print");
    }
    else
    {
        printf("\n element are \n");
        while(ptr!=NULL)
        {
            x=ptr->data;
            printf("\t %d",x);
            ptr=ptr->link;
        }
    }
}
void traverse()
{
    ptr=header;
    if(ptr==NULL)
    {
        printf("Empty list...");
    }
    else
    {
        printf("visited node are \n");
        while(ptr!=NULL)
        {
            printf("\t %d",ptr->data);
            ptr=ptr->link;
        }
    }
}
```

OUTPUT:

```
 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:2
Enter data to insert:12

 Node inserted
 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:2
Enter data to insert:13

 Node inserted
 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:2
Enter data to insert:14
```

```
 Node inserted
 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:2
Enter data to insert:15

 Node inserted
 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:3
Enter the data:10

 Node inserted
 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:4

 Enter the Element:11
```

```
Enter the key value after you want to insert:10

 node inserted
 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:8

 element are
        15      14      13      12      10      11
 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:5

 Node deleted from the beginning..

 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:6
```

```
 Deleted Node from the last...

 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:7
Enter the key element to delete:12

 Deleted 12
 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:9
visited node are
         14      13      10
 1.creation
 2.insert at front
 3.insert at end
 4.insert at any
 5.delete at front
 6.delete at end
 7.delete at any
 8.display
 9.traverse
 10.exit
Enter the choice:10
```

Stack Operations

10a) To implement a C-program using Stack Operations.

```
#include<stdio.h>
#include<stdlib.h>
  void push();
  void pop();
  void display();
  void status();
  void topelement();
  int item,top=0,maxsize,i,a[50];
  int main()
  {
   int ch=0;
   printf("enter size of stack:");
   scanf("%d",&maxsize);
   printf("enter operations given below:\n 1.push\n 2.pop\n 3.display\n 4.status\n
5.topelement\n 6.exit\n");
     while(1)
     {

        printf("enter the choice:");
        scanf("%d",&ch);
        switch(ch)
        {
         case 1:push();
                break;
         case 2:pop();
                break;
         case 3:display();
                break;
         case 4:status();
                break;
         case 5:topelement();
                break;
         case 6:exit(0);
                break;
```

```
        default:
           printf("invalid choice");
       }
    }
}
void push()
{
   printf("enter an element to insert");
   scanf("%d",&item);
   if(top==maxsize)
   {
      printf("stack is full,insertion is not possible");
   }
   else
    {
    top=top+1;
   a[top]=item;
   }
}
 void pop()
{
   if(top==0)
   {
      printf("stack is empty pop is not possible");
   }
   else
    {
    item=a[top];
    top=top-1;
    printf("deleted item is %d",item);
   }
}
 void display()
{
   if(top==0)
   {
      printf("stack is empty");
   }
   else
    {
    printf("\n elements in stack are");
```

```
    i=top;
    while(i!=0)
    {
     printf("%d\t",a[i]);
     i--;
    }
    }
}
void status()
{
   if(top==0)
   {
      printf("stack is empty");
   }
   else if(top==maxsize)
    {
   printf("stack is full");
   }
   else
   {
    printf("\n number of items in stack is %d",top);
    printf("\n number of free space in stack are %d",maxsize-top);
   }
}
void topelement()
{
  if(top==0)
  {
   printf("stack is empty");
  }
   else
   {
    printf("\n top element in stack is %d",a[top]);
   }
}
```

OUTPUT:

```
enter size of stack:5
enter operations given below:
 1.push
 2.pop
 3.display
 4.status
 5.topelement
 6.exit
enter the choice:1
enter an element to insert1
enter the choice:1
enter an element to insert2
enter the choice:1
enter an element to insert3
enter the choice:2
deleted item is 3enter the choice:3

 elements in stack are2 1       enter the choice:4

 number of items in stack is 2
 number of free space in stack are 3enter the choice:5

 top element in stack is 2enter the choice:6
```

Queue Operations

11a)To implement a C-program for Queue Operations.

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

 void enqueue();
 void dequeue();
 void display();
```

```
void status();

int que[100];
int n,front=-1,rear=-1,ele,j;
int main()
{
int c;
printf("enter size of queue:\n");
scanf("%d",&n);
printf("\n the operations are\n 1.enqueue\n 2.dequeue\n 3.display\n 4.status\n 5.exit");
 while(1)
 {
   printf("\n enter your choice:");
   scanf("%d",&c);
   switch(c)
   {
    case 1:enqueue();
           break;
    case 2:dequeue();
           break;
    case 3:display();
           break;
    case 4:status();
           break;
    case 5:exit(0);
           break;
   }
 }
}
void enqueue()
{
   if(rear==n-1)
   {
    printf("\n queue is full insertion not possible");
   }
   else
   {
    printf("\n enter element to insert:");
    scanf("%d",&ele);
    {
       front=0;
```

```
 }
 rear++;
 que[rear]=ele;
  }
}
void dequeue()
{
  if(rear==0&&front==0)
  {
     printf("\n queue is empty deletion is not possible");
  }
   else
    {
     ele=que[front];
     printf("\n deleted item is %d",ele);
     front++;
    }
  }
void display()
{
 if(front==0&&rear==0)
 {
    printf("queue is empty");
 }
   else
   {
      for(j=front;j<=rear;j++)
      {
         printf("\n elements in queue are %d",que[j]);
      }
   }
}
void status()
{
   if(front==0&&rear==0)
   {
      printf("\n queue is empty");
   }
   else if(rear==n-1)
   {
      printf("\n queue is full");
```

```
    }
    else
     {
      printf("number of elements in queues is %d",rear);
      printf("number of free space in queue is %d",n-rear);
     }
  }
```

OUTPUT:

```
enter size of queue:
5

 the operations are
 1.enqueue
 2.dequeue
 3.display
 4.status
 5.exit
 enter your choice:1

 enter element to insert:12

 enter your choice:1

 enter element to insert:13

 enter your choice:1

 enter element to insert:14

 enter your choice:1

 enter element to insert:15

 enter your choice:1

 enter element to insert:16

 enter your choice:2
```

```
deleted item is 12
enter your choice:3

elements in queue are 13
elements in queue are 14
elements in queue are 15
elements in queue are 16
enter your choice:4

queue is full
enter your choice:5
```

Searching

12a) To implement a C-program for Linear Search.

```
#include<stdio.h>
void main()
 {
 int a[50],i,key,n;
 printf("enter number of elements:");
 scanf("%d",&n);
 printf("the elements are:");
 for(i=0;i<n;i++)
 {
 scanf("%d",&a[i]);
 }
 for(i=0;i<n;i++)
 {
 printf("%d",a[i]);
 }
 printf("enter search element");
 scanf("%d",&key);
 lsearch(a,n,key);
 getch();
 }
 int lsearch(int a[],int n,int key)
 {
 int j;
 for(j=0;j<n;j++)
```

```
{
if(a[j]==key)
{
printf("element found at %d position",j);
return j;
}
}
printf("element not found");
return -1;
}
```

OUTPUT:

```
enter number of elements:5
the elements are:1 2 3 4 5
12345enter search element 3
element found at 2 position
```

12b) To implement a C-program for Binary Search.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a[50],i,key,n;
  printf("enter number of elements:");
  scanf("%d",&n);
  printf("the elements are:");
   for(i=0;i<n;i++)
   {
    scanf("%d",&a[i]);
   }
    for(i=0;i<n;i++)
    {
    printf("%d",a[i]);
    }
    printf("enter search element");
    scanf("%d",&key);
    Bsearch(a,n,key);
    getch();
}
```

```
int Bsearch(int a[],int n,int key)
{
 int mid,start,end;
 start=0;
 end=n-1;
 while(start<=end)
 {
    mid=(start+end)/2;
    if(key==a[mid])
    {
       printf("element found");
       return mid;
    }
    else if(key<a[mid])
    {
       end=mid-1;
    }
    else if(key>a[mid])
    {
       start=mid+1;
    }
 }
 printf("element not found");
 return (-1);
}
```

OUTPUT:

```
enter number of elements:5
the elements are:1 2 3 4 5
12345enter search element 3
element found
```

Sorting

13a) To implement a C-program for Bubble Sort

```
#include<stdio.h>
#include<conio.h>
main()
{
 int a[50],k,n;
 printf("enter number of elements for sorting:");
 scanf("%d",&n);
 printf("the elements to be sorted are:");
   for(k=0;k<n;k++)
   {
    scanf("%d",&a[k]);
   }
   bubblesort(a,n);
   for(k=0;k<n;k++)
   {
    printf("%d\t",a[k]);
   }
   return 0;
}
 int bubblesort(int a[],int n)
 {
     int i,j,temp;
     for(i=0;i<n;i++)
     {
     for(j=1;j<n;j++)
     {
       if(a[j]<a[j-1])
       {
          temp=a[j-1];
          a[j-1]=a[j];
          a[j]=temp;
     }
     }
     }
 }
```

OUTPUT

```
enter number of elements for sorting:5
the elements to be sorted are:1 2 3 4 5
1       2       3       4       5
```

13b) To implement a C-program for Selection Sort

```
#include<stdio.h>
#include<conio.h>
 void main()
 {
 int a[50],k,n,temp;
 printf("enter number of elements for sorting:");
 scanf("%d",&n);
 printf("the elements to be sorted are:");
  for(k=0;k<n;k++)
  {
   scanf("%d",&a[k]);
  }
  selection(a,n);
  for(k=0;k<n;k++)
  {
   printf("%d\t",a[k]);
  }
  getch();
  }
  int selection(int a[],int n)
  {
   int i,j,temp;
   for(i=0;i<n;i++)
   {
      for(j=i+1;j<n;j++)
      {
      if(a[i]>a[j])
      {
         temp=a[i];
         a[i]=a[j];
         a[j]=temp;
      }
   }
```

```
  }
  }
```

OUTPUT:

```
enter number of elements for sorting:5
the elements to be sorted are:2 5 3 4 1
1       2       3       4       5
```

13c) To implement a C-program for Quick Sort

```
#include<stdio.h>
#include<conio.h>
int main()
{
 int a[50],k,n,low,high;
 printf("enter number of elements for sorting:");
 scanf("%d",&n);
 printf("the elements to be sorted are:");
 for(k=0;k<n;k++)
 {
  scanf("%d",&a[k]);
 }
 quicksort(a,0,n-1);
 printf("sorted elements are:");
  for(k=0;k<n;k++)
  {
     printf("%d\t",a[k]);
  }
  return 0;
}
 int quicksort(int a[],int low, int high)
{
   int pivot,i,j,temp;
   if(low<high)
   {
      pivot=low;
      i=low;
      j=high;
```

```
    while(i<j)
    {
     while(a[i]<=a[pivot]&&(i<high))
     i++;
     while(a[j]>a[pivot]&&(j>low))
     j--;
     if(i<j)
     {
        temp=a[i];
        a[i]=a[j];
        a[j]=temp;
     }
    }
    temp=a[pivot];
    a[pivot]=a[j];
    a[j]=temp;
    quicksort(a,low,j-1);
    quicksort(a,j+1,high);
  }
}
```

OUTPUT

```
enter number of elements for sorting:5
the elements to be sorted are:23 22 24 21 25
sorted elements are:21  22      23      24      25
```

13d) To implement a C-program for Merge Sort

```
#include<stdio.h>
int mergesort();
int merge();
int main()
{
  int a[20],n,k,i;
  printf("enter number of elements:");
  scanf("%d",&n);
  printf("enter elements:");
  for(i=0;i<n;i++)
```

```
  {
   scanf("%d",&a[i]);
  }
  mergesort(a,0,n-1);
  printf("sorted elements are:");
  for(i=0;i<n;i++)
  {
   printf("%d",a[i]);
  }
  return 0;
}
int mergesort(int a[],int low,int high)
{
 int mid;
 if(low<high)
 {
  mid=(low+high)/2;
  mergesort(a,low,mid);
  mergesort(a,mid+1,high);
  merge(a,low,mid,high);
 }
}
int merge(int a[],int low,int mid,int high)
{
   int i,k,h,j,temp[20];
   i=low;
   h=low;
   j=mid+1;
   while((h<=mid)&&(j<=high))
   {
     if(a[h]<a[j])
     {
        temp[i++]=a[h];
        h++;
     }
     else
      {
       temp[i++]=a[j];
        j++;
      }
   }
```

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

OUTPUT:

```
enter number of elements:5
enter elements:
43
42
44
45
41
sorted elements are:    41      42      43      44      45
```