INVERTIS UNIVERSITY, BAREILLY

NH-24, Invertis Village Delhi Luckow Highway Bareilly-243123 (UP) India
Ph.:+91-(0581)-2460442, 2460443, 3501350
Web : www.invertisuniversity.ac.in

INVERTIS UNIVERSITY,BAREILLY

LAB MANUALS

DATA STRUCTURE USING 'C'

BCS-351

(CSE DEPARTMENT)

w.e.f.2023 to

INVERTIS UNIVERSITY, BAREILLY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Data Structure Using 'C' (BCS-351)

LIST OF PROGRAMS

S. No.	Program No.	NAME OF THE EXPERIMENT
1.	Prog.-1	*Write a program to insert into and delete element from the array.*
2.	Prog.-2	*Write a program to implement stack using array*
3.	Prog.-3	*Write a program to implement Stack using Linked List.*
4.	Prog.-4	*Write a program to implement Queue using array.*
5.	Prog.-5	*Write a program to implement Linked List.*
6.	Prog.-6	*Write a program to implement Sequential search algorithm.*
7.	Prog-7	*Write a program to implement Binary search algorithm*
8.	Prog-8	*Write a program to implement Bubble sort algorithm.*
9	Prog-9	*Write a program to implement Insertion sort algorithm*
10	Prog-10	*Write a program to implement quick sort algorithm.*

Faculty Incharge
(Mr. Ratnesh Kumar Pandey)

H.O.D (CS/IT)
(Dr. Gaurav Agarwal)

Data Structure Using 'C'

(BCS-351)

PROGRAM NO: 01

DEPARTMENT OF CSE/IT

INVERTIS UNIVERSITY BAREILLY -243123 UTTAR PRADESH (INDIA)

Program No: 1
TITLE: INSERTION & DELETION IN ARRAY

1.1 Objective 1.2 Theory 1.3 Flow chart /Algorithm 1.4. Coding .1.5 Output

***1.1* OBJECTIVE:** *Write a program to insert into and delete element from the array.*

1.2 THEORY: Array is collection of homogeneous data elements.

- A linear array is a list of finite number n of homogenous data elements (i.e., data elements of same type).
- Like ordinary variables arrays too can be initialized during declaration.
  int num[6]={2,4,12,5,45,5}
  int n[]={2,4,12,5,45,5}
- Array elements are stored in contiguous memory locations.
- The size of the array should be mentioned while declaring it. e.g. n[5]
- Array elements are always counted from 0 onwards.
- Array elements can be accessed using the position of the element in the array. e.g. n[i] refers to ith element.

Length of Array

- Length = UB – LB + 1
- UB is the largest index, called the upper bound.
- LB is the smallest index, called the lower bound.

Location of element

- LOC(K) = BASE(LA) + W(K- LB)
  LA – LINEAR ARRAY
  W - SIZE TAKEN BY VARIABLE
  LB – LOWER BOUND

Operation on Array

- Traversal. -> Processing each element in the list
- Search. -> Find the location of the element
- Insertion -> Adding a new element to the list
- Deletion -> Removing an element from the list
- Sorting -> Arranging the elements in order
- Merging -> Combining two list into a single list.

1.3 FLOW CHART / ALGORITHM :

INSERTION IN ARRAY

Let **A** is the Array of **N** elements. A new element **data** is to be inserted in array at **K**th position.

```
Step 1:   Repeat step 2  for I=N to K
Step 2:                   Set A[I+1]=A[I]
Step 3:  Set A[K] =Data.
Step 4:  Set N=N+1
Step  5: Exit.
```

DELETION FROM ARRAY

Let **A** is the Array of **N** elements. A element **data** is to be deleted from **K**th position of array.

```
Step 1:  Repeat step2 for I=K to N-1
Step 2:                  Set A[I] =A[I+1]
Step 3:  Set N=N-1
Step 4: Exit
```

1.4 CODING:

```
#include<conio.h>
#include<stdio.h>
void insertion();
void deletion();
static int a[20];
int n=0;
void main()
{  int ch,x; clrscr();
do{
printf("Press 1 for inertion\n ");
printf(" Press 2 for deletion\n");
printf(" Press 3 for exit");
printf("ENTER YOU CHOICE")
scanf(" %d",&ch);
switch(ch)
{
case 1: insertion(); break;
case 2: deletion(); break;
case 3: exit(1); break;
default: printf(" wrong choice.");
}
printf(" Would u like to continue\n press 1 if yes\n");
scanf("%d",&x);
}while(x==1);



getch();
}
void insertion()
{      int k,i ,value;
printf(" give the position of array where a element is to be inserted");
scanf("%d",&k);
printf(" Enter the element to be inserted");
scanf("%d",&value);
for(i=n;i>=k;i--)
{
a[i+1]=a[i];
}
a[k]=value;
n=n+1;
printf(" Elements of array");
for(i=0;i<n;i++)
printf("%d\n",a[i]);
}
void deletion()
{
  int k,i ,value;
printf(" give the position of array where a element is to be deleted");
scanf("%d",&k);
 for(i=k;i<=n;i++)
a[i]=a[i+1];
value=a[k];  printf(" deleted element is =%d",value);
n=n-1;
printf(" Elements of array");
for(i=0;i<n;i++)
printf(" %d",a[i]);
}
```

1.5 OUTPUT:

```
Press 1 for insertion
Press 2  for deletion
Press 3 for exit
ENTER YOU CHOICE 1
give the position of array where a element is to be inserted 0
Enter the element to be inserted 111
Elements of array 111

Would u like to continue\n press 1 if yes
 1

Press 1 for insertion
Press 2  for deletion
Press 3 for exit
ENTER YOU CHOICE 1
give the position of array where a element is to be inserted 1
Enter the element to be inserted 222
Elements of array 111  222

Would u like to continue\n press 1 if yes
 1

Press 1 for insertion
Press 2 for deletion
Press 3 for exit
ENTER YOU CHOICE 1
ENTER YOU CHOICE 2

the position of array where a element is to be deleted 0
Elements of array

Would u like to continue\n press 1 if yes
 1

Press 1 for insertion
Press 2 for deletion
Press 3 for exit
ENTER YOU CHOICE 3
```

Data Structure Using 'C'
(BCS-351)

PROGRAM NO: 02

DEPARTMENT OF CSE

INVERTIS UNIVERSITY BAREILLY -243123 UTTAR PRADESH (INDIA)

Program No: 2
TITLE: STACK USING ARRAY

1.1 Objective 1.2 Theory 1.3 Flow chart /Algorithm 1.4. Coding .1.5 Output

***1.1* OBJECTIVE:** *Write a program to implement stack using array.*

1.2 THEORY: A collection of items in which only the most recently added item may be removed. The latest added item is at the top. Basic operations are push and pop. . Also known as "last-in, first-out" or LIFO.

1.3 FLOW CHART / ALGORITHM :

PUSH DATA ON TO STACK
Let **S** is the Stack of maximum size **MAXSTACK**. A new element **data** is to be inserted on to stack. Initially **TOP**= (-1)

```
Step 1: if TOP=MAXSTACK Then Print : OVERFLOW and Exit.
Step 2:  else TOP=TOP+1
Step 3:          S[TOP]=data
Step 4:  Exit.
```

POP DATA FROM STACK
Let **S** is the Stack of maximum size **MAXSTACK**. A element **data** is to be deleted from stack.

```
Step 1: if TOP>0 Then Print : UNDERFLOW and Exit.
Step 2: else Data=S[TOP]
Step 3:  TOP=TOP-1
Step 4: Exit.
```

1.4 CODING:

```
#include<stdio.h>
#include<conio.h>
#define MAXSIZE 10
void push();
int pop();
void traverse();
int stack[MAXSIZE];
int Top=-1;
void main()
{
 int choice;
 char ch;
 do
  {
    clrscr();
    printf("\n1. PUSH ");
    printf("\n2. POP ");
    printf("\n3. TRAVERSE ");
    printf("\nEnter your choice");
    scanf("%d",&choice);
    switch(choice)
     {
      case 1:  push();
                  break;
      case 2:  printf("\nThe deleted element is %d",pop());
                  break;
      case 3:  traverse();
                  break;
      default: printf("\nYou Entered Wrong Choice");
      }
    printf("\nDo You Wish To Continue (Y/N)");
    fflush(stdin);
    scanf("%c",&ch);
    }
 while(ch=='Y' || ch=='y');
}

void push()
{
 int item;
 if(Top == MAXSIZE - 1)
  {
    printf("\nThe Stack Is Full");
    getch();
    exit(0);
    }
 else
  {
    printf("Enter the element to be inserted");
    scanf("%d",&item);
    Top= Top+1;
    stack[Top] = item;
    }
}

int pop()
{
 int item;
 if(Top == -1)
  {
    printf("The stack is Empty");
    getch();
    exit(0);
    }
 else
   {
    item = stack[Top];
    Top = Top-1;
    }
return(item);
}

void traverse()
{
 int i;
 if(Top == -1)
  {
    printf("The Stack is Empty");
    getch();
    exit(0);
    }
 else
  {
    for(i=Top;i>=0;i--)
     {
      printf("Traverse the element");
       printf("\n%d",stack[i]);
       }
    }
}
```

1.5 OUTPUT:

```
1. PUSH
2. POP
3. TRAVERSE
Enter your choice
1
Enter the element to be inserted
11
Do You Wish To Continue (Y/N)");
Y

1. PUSH
2. POP
3. TRAVERSE
Enter your choice
1
Enter the element to be inserted
22
Do You Wish To Continue (Y/N)");
```

```
Y
1. PUSH
2. POP
3. TRAVERSE
Enter your choice
1
Enter the element to be inserted
33
Do You Wish To Continue (Y/N)");
Y

1. PUSH
2. POP
3. TRAVERSE
Enter your choice
3
Traverse the element
    33 22 11

Do You Wish To Continue (Y/N)");
Y
1. PUSH
2. POP
3. TRAVERSE
Enter your choice
2
The deleted element is 33

Do You Wish To Continue (Y/N)");
Y
1. PUSH
2. POP
3. TRAVERSE

Enter your choice
3
Traverse the element
22 11
```

Data Structure Using 'C'

(BCS-351)

PROGRAM NO: 03

DEPARTMENT OF CSE

INVERTIS UNIVERSITY BAREILLY -243123 UTTAR PRADESH

Program No: 3
TITLE: STACK USING LINKED LIST

1.1 Objective 1.2 Theory 1.3 Flow chart /Algorithm 1.4. Coding .1.5 Output

***1.1* OBJECTIVE:** *Write a program to implement Stack using Linked List.*

1.2 THEORY: A collection of items in which only the most recently added item may be removed. The latest added item is at the top. Basic operations are push and pop. . Also known as "last-in, first-out" or LIFO.

1.3 FLOW CHART / ALGORITHM :

PUSH OPERATION

```
Step 1:  node=(st *)malloc(sizeof(st));          // Create new node
Step 2:  node->no= data ;                         // insert data in node
Step 3:  node->next=start;
Step4 :  start=node;
```

POP OPERATION:

```
Step 1:  temp=start;
Step 2:  if(start==NULL)
                 printf("stack is already empty"); exit();
Step 3: else    start=start->next;
Step 4:  free(temp);
```

1.4 CODING:

```
#include<stdio.h>
#include<conio.h>
struct stack
{
int no;
struct stack *next;
}
*start=NULL;
typedef struct stack st;
void push();
int pop();
void display();
void main()
{
char ch;
int choice,item;
do
{
clrscr();
printf("\n 1: push");
printf("\n 2: pop");
printf("\n 3: display");
printf("\n Enter your choice");
scanf("%d",&choice);
switch (choice)
{
case 1: push();
break;
case 2: item=pop();
printf("The delete element in %d",item);
break;
case 3: display();
break;
default : printf("\n Wrong choice");
};
printf("\n do you want to continue(Y/N)");
fflush(stdin);
scanf("%c",&ch);
}
while (ch=='Y'||ch=='y');
}
void push()
{
st *node;
node=(st *)malloc(sizeof(st));
printf("\n Enter the number to be insert");
scanf("%d",&node->no);
node->next=start;
start=node;
}
int pop()
{
st *temp;
temp=start;
if(start==NULL)
{
printf("stack is already empty");
getch();
exit();
}
else
{
```

```
start=start->next;
free(temp);
}
return(temp->no);
}
void display()
{
st *temp;
temp=start;
while(temp->next!=NULL)
{
printf("\nno=%d",temp->no);
temp=temp->next;
}
printf("\nno=%d",temp->no);
}
```

1.5 OUTPUT:

```
1. PUSH
2. POP
3. TRAVERSE
Enter your choice
1
Enter the element to be inserted
11
Do You Wish To Continue (Y/N)");
Y

1. PUSH
2. POP
3. TRAVERSE
Enter your choice
1
Enter the element to be inserted
22
Do You Wish To Continue (Y/N)");
Y
1. PUSH
2. POP
3. TRAVERSE
Enter your choice
1
Enter the element to be inserted
33
Do You Wish To Continue (Y/N)");
Y

1. PUSH
```

```
2. POP
3. TRAVERSE
Enter your choice
3
Traverse the element
    34 22 11

Do You Wish To Continue (Y/N)");
Y
1. PUSH
2. POP
3. TRAVERSE
Enter your choice
2
The deleted element is 33

Do You Wish To Continue (Y/N)");
Y
1. PUSH
2. POP
3. TRAVERSE

Enter your choice
3
Traverse the element
22 11
```

Data Structure Using 'C'

(BCS-351)

PROGRAM NO: 04

DEPARTMENT OF CSE

INVERTIS UNIVERSITY BAREILLY -243123 UTTAR PRADESH (INDIA)

Program No: 4
TITLE: QUEUE

1.1 Objective 1.2 Theory 1.3 Flow chart /Algorithm 1.4. Coding .1.5 Output

***1.1* OBJECTIVE:** *Write a program to implement Queue using array.*

1.2 THEORY: : A collection of items in which only the earliest added item may be accessed. Basic operations are add (to the *REAR*) or enqueue and delete (from the *FRONT*) or dequeue. Delete returns the item removed. Also known as "first-in, first-out" or FIFO

1.3 FLOW CHART / ALGORITHM :

INSERTION IN QUEUE
Let **Q** is the Queue of N elements. A element **data** is to be inserted in Queue.
Initially **rear=(-1)** and **front=(-1)**

```
Step1: if rear>=N then Print: OVERFLOW and exit.
Step2: Set rear=rear+1
Step3: Set Q[rear]=Data
Step4: if front= (-1) then front=0
Step5: Exit
```

DELETION FROM QUEUE
Let **Q** is the Queue of N elements. A element **data** is to be deleted from Queue.

```
Step1: if front<0 || front>rear then print: UNDERFLOW and exit
Step2: Set Data=Q[front]
Step3: Set front=front+1
Step4: exit
```

1.4 CODING:

```
#include<stdio.h>
#include<conio.h>
#define MAXSIZE 5
int front=-1, rear=-1,choice;
int q[10];
void main()
{
clrscr();
do
{
printf("\n1-->insert\n");
printf("2-->delete\n");
printf("3-->display\n");
printf("4-->exit\n");
printf("enter your choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:qinsert();
      break;
case 2:qdelete();
       break;
case 3:qdisplay();
       break;
case 4:return;
}
}
while(choice!=4);
}
 qinsert()
{
int num;
if(rear==(MAXSIZE-1))
{
printf("queue is full\n");
return;
}
else
{
printf("enter no\n");
scanf("%d",&num);
rear=rear+1;
q[rear]=num;
if(front==-1)
{
 front++;
 }
 }
 return;
 }
qdelete()
{
int num;
if(front==-1)
{
printf("queue empty\n");
return;
}
else
{
if(front==rear)
front=rear=-1;
else
{
```

```
num=q[front];
printf("deleted item=%d",q[front]);
front++;
}
}
return(num);
}
qdisplay()
{
int i;
if(front==-1)
{
printf("queue empty\n");
return;
}
else
{
printf("\nThe status of the queu\n");
for(i=front;i<=rear;i++)
{
printf("%d\n",q[i]);
}
}
printf("\n");
}
```

1.5 OUTPUT:

```
1…> Insert
2….> delete
3…..> display
4……>exit
enter your choice
1
enter no 33
Do u want to Comntine y
1…> Insert
2….> delete
3…..> display
4……>exit
enter your choice
1
enter no 44
Do u want to Comntine y
1…> Insert
2….> delete
3…..> display
4……>exit
enter your choice
```

```
1
enter no 55

Do u want to Comntine y
1…> Insert
2….> delete
3…..> display
4……>exit
enter your choice
3
The status of the queue
33 44 55
Do u want to Comntine y
1…> Insert
2….> delete
3…..> display
4……>exit
enter your choice
2
deleted item 33
Do u want to Comntine y
1…> Insert
2….> delete
3…..> display
4……>exit
enter your choice
3
The status of the queue
 44 55
```

Data Structure Using 'C'
(BCS-351)

PROGRAM NO: 05

DEPARTMENT OF CSE

INVERTIS UNIVERSITY BAREILLY -243123 UTTAR PRADESH (INDIA)

Program No: 5
TITLE: LINKED LIST

1.1 Objective 1.2 Theory 1.3 Flow chart /Algorithm 1.4. Coding .1.5 Output

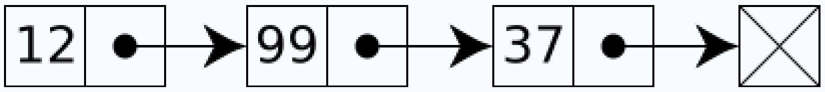
***1.1* OBJECTIVE:** *Write a program to implement Linked List.*

1.2 THEORY: a **linked list** is one of the fundamental data structures, and can be used to implement other data structures. It consists of a sequence of nodes, each containing arbitrary data fields and one or two references ("links") pointing to the next and/or previous nodes. The principal benefit of a linked list over a conventional array is that the order of the linked items may be different from the order that the data items are stored in memory or on disk, allowing the list of items to be traversed in a different order. A linked list is a self-referential datatype because it contains a pointer or link to another datum of the same type. Linked lists permit insertion and removal of nodes at any point in the list in constant time,[1] but do not allow random access. Several different types of linked list exist: singly-linked lists, doubly-linked lists, and circularly-linked lists.

Types of linked lists

Singly-linked list

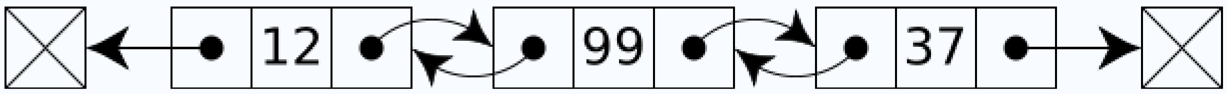
The simplest kind of linked list is a **singly-linked list** (or **slist** for short), which has one link per node. This link points to the next node in the list, or to a null value or empty list if it is the final node.



A singly-linked list containing three integer values

Doubly-linked list

A more sophisticated kind of linked list is a **doubly-linked list** or **two-way linked list**. Each node has two links: one points to the previous node, or points to a null value or empty list if it is the first node; and one points to the next, or points to a null value or empty list if it is the final node.



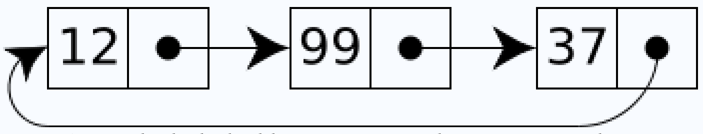
A doubly-linked list containing three integer values

In some very low level languages, Xor-linking offers a way to implement doubly-linked lists using a single word for both links, although the use of this technique is usually discouraged. A similar link storage preserving technique is the subtraction edge.

Circularly-linked list

In a **circularly-linked list**, the first and final nodes are linked together. This can be done for both singly and doubly linked lists. To traverse a circular linked list, you begin at any node and follow the list in either direction until you return to the original node. Viewed another way, circularly-linked lists can be seen as having no beginning or end. This type of list is most useful for managing buffers for data ingest, and in cases where you have one object in a list and wish to see all other objects in the list.

The pointer pointing to the whole list may be called the access pointer.



A circularly-linked list containing three integer values

Singly-circularly-linked list

In a **singly-circularly-linked list**, each node has one link, similar to an ordinary *singly-linked list*, except that the next link of the last node points back to the first node. As in a singly-linked list, new nodes can only be efficiently inserted after a node we already have a reference to. For this reason, it's usual to retain a reference to only the last element in a singly-circularly-linked list, as this allows quick insertion at the beginning, and also

allows access to the first node through the last node's next pointer.

Doubly-circularly-linked list

In a **doubly-circularly-linked list**, each node has two links, similar to a *doubly-linked list*, except that the previous link of the first node points to the last node and the next

1.3 FLOW CHART / ALGORITHM :

1.4 CODING:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct node
{
int info;
struct node *next;
};
typedef struct node NODE;
NODE *start;
void createmptylist(NODE **start)
{
```

```
*start=(NODE *)NULL;
}
void traversinorder(NODE *start)
{
while(start != (NODE *) NULL)
{
printf("%d\n",start->info);
start=start->next;
}
}
void insertatbegin(int item)
{
NODE *ptr;
ptr=(NODE *)malloc(sizeof(NODE));
ptr->info=item;
if(start==(NODE *)NULL)
ptr->next=(NODE *)NULL;
else
ptr->next=start;
start=ptr;
}
void insert_at_end(int item)
{
NODE *ptr,*loc;
ptr=(NODE *)malloc(sizeof(NODE));
ptr->info=item;
ptr->next=(NODE *)NULL;
if(start==(NODE*)NULL)
start=ptr;
else
{
loc=start;
while(loc->next!=(NODE *)NULL)
loc=loc->next;
loc->next=ptr;
}
}
void insert_spe(NODE *start,int item)
{
NODE *ptr,*loc;
int temp,k;
for(k=0,loc=start;k<temp;k++)
{
loc=loc->next;
if(loc==NULL)
{
printf("node in the list at less than one\n");
return;
}
}
```

```
ptr=(NODE *)malloc(sizeof(NODE));
ptr->info=item;
ptr->next=loc->next;;
loc->next=ptr;
}
void main()
{
int choice,item,after;
char ch;
clrscr();
createmptylist(start);
do
{
printf("1.Insert element at begin \n");
printf("2. insert element at end positon\n");
printf("3. insert specific the position\n");
printf("4.travers the list in order\n");
printf("5. exit\n");
printf("enter your choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("Enter the item\n");
        scanf("%d",&item);
        insertatbegin(item); break;
case 2:  printf("Enter the item\n");
        scanf("%d",&item);
        insert_at_end(item); break;
case 3: printf("Enter the item\n");
        scanf("%d",&item);
        insert_spe(start,item);
        break;
case 4: printf("\ntravers the list\n");
        traversinorder(start);
        break;
case 5: return;
}
fflush(stdin);
printf("do your want continous\n");
scanf("%c",&ch);
}while((ch='y')||(ch='y'));
getch();
}
```

1.5 OUTPUT:

```
1.Insert element at begin
2. insert element at end positon
3. insert specific the position
4.travers the list in order
5. exit
enter your choice1
Enter the item  111
do your want continue    y
1.Insert element at begin
2. insert element at end positon
3. insert specific the position
4.travers the list in order
5. exit

enter your choice2
Enter the item  222
do your want continue    y
1.Insert element at begin
2. insert element at end positon
3. insert specific the position
4.travers the list in order
5. exit

enter your choice2

Enter the item  333
do your want continue    y
1.Insert element at begin
2. insert element at end positon
3. insert specific the position
4.travers the list in order
5. exit

enter your choice 4
Elements of Linked List are :  111  222   333
```

Data Structure Using 'C'

(BCS-351)

PROGRAM NO: 06

DEPARTMENT OF CSE

INVERTIS UNIVERSITY BAREILLY -243123 UTTAR PRADESH (INDIA)

Program No: 6
TITLE: LINEAR SEARCH

1.1 Objective 1.2 Theory 1.3 Flow chart /Algorithm 1.4. Coding .1.5 Output

1.1 **OBJECTIVE:** *Write a program to implement Sequential search algorithm.*

1.2 THEORY: linear search is a the simplest type of searching technique use to search the data element in the data structure this is also known as the sequential search due to it's processing the search of data start from the beginning of data structure (Array)and continuous in linearly until data is found .

1.3 FLOW CHART / ALGORITHM :

LINEAR SEARCH
Let **A** is the Array of **N** elements. **Data** is the element whose location is to be search and the **LOC** is the position of searched data.

```
Step1: Set I=1
Step2: repeat step3 while A[I]=Data && I<N
Step3:     Set I=I+1
Step4:     if Data=A[I] then LOC=I
Step5:     else LOC=NULL
Step6: Exit
```

1.4 CODING:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],n,i,item,loc=-1;
clrscr();
printf("\nEnter the number of element:");
scanf("%d",&n);
printf("Enter the number:\n");
for(i=0;i<=n-1;i++)
   {
    scanf("%d",&a[i]);
   }
printf("Enter the no. to be search\n");
scanf("%d",&item);
for(i=0;i<=n-1;i++)
   {
    if(item==a[i])
      {
       loc=i;
       break;
     }
   }
if(loc>=0)

   printf("\n%dis found in position%d",item,loc+1);
   else
   printf("\nItem does not exits");
   getch();
     }
```

1.5 OUTPUT:

```
How many elements
5
Enter the element of the array
     7    8 3 2 9
Enter the element to be searching
9
search is successful
position of the item 9 is 5
```

Data Structure Using 'C'

(BCS-351)

PROGRAM NO: 07

DEPARTMENT OF CSE

INVERTIS UNIVERSITY BAREILLY -243123 UTTAR PRADESH (INDIA)

Program No: 7
TITLE: BINARY SEARCH

1.1 Objective 1.2 Theory 1.3 Flow chart /Algorithm 1.4. Coding .1.5 Output

1.1 **OBJECTIVE:** *Write a program to implement Binary search algorithm. .*

1.2 THEORY: binary search technique is a special search technique for sorted array .in sorted array the binary search algorithm is efficient to find the location of given data item in binary search method , search starts from mid sub script array ,the left of array contains all a element lesser than the mid element .the right of array contains all element s greater than the mid element so the searching f data continuous from mid to either left or right .

1.3 FLOW CHART / ALGORITHM :

BINARY SEARCH

Let **A** is the Array of **N** sorted elements. **Data** is the element whose location is to be search and the **LOC** is the position of searched data. **BEG, END, MID** are the data elements used to hold the steps subscripts during search.

```
Step1: Set BEG=0
       Set END=N-1
       Set MID=(BEG+END)/2
Step2: repeat step3 & 4 while Data!=A[MID] and BEG<=END
Step3:     If Data >A[MID] then Set BEG=MID+1
           Else END=MID-1
Step4:  Set MID=(BEG+END)/2
Step5:  if Data =A[MID]  Then LOC=MID
        Else LOC=NULL
Step6: Exit
```

1.4 CODING:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],i,loc,mid,beg,end,n,flag=0,item;
clrscr();
printf("How many elements");
scanf("%d",&n);
printf("Enter the element of the array\n");
for(i=0;i<=n-1;i++)
{
scanf("%d",&a[i]);
}
printf("Enter the element to be searching\n");
scanf("%d",&item);
loc=0;
beg=0;
end=n-1;
while((beg<=end)&&(item!=a[mid]))
{
    mid=((beg+end)/2);
      if(item==a[mid])
           {
             printf("search is successfull\n");
             loc=mid;
             printf("position of the item%d\n",loc+1);
             flag=flag+1;
           }
     if(item<a[mid])
          end=mid-1;
 else
          beg=mid+1;
}
if(flag==0)
{
printf("search is not successfull\n");
}
getch();
}
```

1.5 OUTPUT:

```
How many elements

5

Enter the element of the array

     8    8 3 2 9

Enter the element to be searching

8

search is successful

position of the item 8 is 2
```

Data Structure Using 'C'
(BCS-351)

PROGRAM NO: 08

DEPARTMENT OF CSE

INVERTIS UNIVERSITY BAREILLY -243123 UTTAR PRADESH
(INDIA)

Program No: 8
TITLE: BUBBLE SORT

1.1 Objective 1.2 Theory 1.3 Flow chart /Algorithm 1.4. Coding .1.5 Output

***1.1* OBJECTIVE:** *Write a program to implement Bubble sort algorithm.*

1.2 THEORY: it is very simple sorting technique. However, this sorting technique is not efficient in comparison to others shorting techniques but for smaller list this shorting techniques suppose we short the elements in ascending order the bubble shorts loops through the elements in the list comparing the adjacent elements and loops the largest element to the bottom of the list **.**

1.3 FLOW CHART / ALGORITHM :

BUBBLE SORT

```
Let A is the Array of N elements to be sorted in ascending order.
Step1: repeat step 2&3 for I=1 to N
Step2:     repeat steps for J=1 to n
Step3:         if A[J]>A[J+1]
                Then  Temp = A[J]
                      A[J] = A[J+1]
                      A[J+1] = Temp
Step4: Exit.
```

1.4 CODING:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],n,i,j,temp;
clrscr();
printf("How many elements");
scanf("%d",&n);
printf("Enter the element of array");
for(i=0;i<=n-1;i++)
 {
 scanf("%d",&a[i]);
 }
for(i=0;i<=n-1;i++)
{
 for(j=0;j<=n-1-i;j++)
    {
    if(a[j]>a[j+1])
         {
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
           }
      }
}
printf("Element of array after the sorting are:\n");
for(i=0;i<=n-1;i++)
{
printf("%d\n",a[i]);
}
getch();
}
```

1.5 OUTPUT:

```
How many elements

5

Enter the element of array

3 5 2 1 4

Element of array after the sorting are:

1 2 3 4 5
```

Data Structure Using 'C'
(BCS-351)

PROGRAM NO: 09

DEPARTMENT OF CSE

INVERTIS UNIVERSITY BAREILLY -243123 UTTAR PRADESH (INDIA)

Program No: 9
TITLE: INSERTION SORT

1.1 Objective 1.2 Theory 1.3 Flow chart /Algorithm 1.4. Coding .1.5 Output

***1.1* OBJECTIVE:** *Write a program to implement Insertion sort algorithm.*

1.2 THEORY: Insertion sort process is used to sort the list of elements by inserting each data element at its appropriate position in array.

1.3 FLOW CHART / ALGORITHM :

INSERTION SORT

Let **A** is an array of **N** Elements

```
Step 1. initialize A[0]=0
Step 2. Repeat step 3 to 5 for i=1 to N
Step 3. temp=A[i]
        K=i-1
Step 4. while(temp<A[K])
        A[K+1]=A[K]
        K++
Step 5. A[K]=temp
Step 6. Exit
```

1.4 CODING:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],n,k,i,j,temp;
clrscr();
printf("How many elements\n");
scanf("%d",&n);
printf("Enter the element of array");
for(i=0;i<=n-1;i++)
{
scanf("%d",&a[i]);
}
for(k=1;k<=n-1;k++)
{
temp=a[k];
j=k-1;
while((temp<a[j])&&(j>=0))
{
```

```
a[j+1]=a[j];
j=j-1;
}
a[j+1]=temp;
}
printf("Element of array after sorting\n");
for(i=0;i<=n-1;i++)
{
printf("%d\n",a[i]);
}
getch();
}
```

1.5 OUTPUT:

```
How many elements
5
Enter the element of array
3 5 2 1 4
Element of array after the sorting are:
1 2 3 4 5
```

Data Structure Using 'C'

(BCS-351)

PROGRAM NO: 10

DEPARTMENT OF CSE

INVERTIS UNIVERSITY BAREILLY -243123 UTTAR PRADESH (INDIA)

D

Program No: 10
TITLE: QUICK SORT

1.1 Objective 1.2 Theory 1.3 Flow chart /Algorithm 1.4. Coding .1.5 Output

***1.1* OBJECTIVE:** *Write a program to implement quick sort algorithm.*

1.2 THEORY: Quick sort treats an array as list of elements. When ort begins it select the middle element of list as a pivot The sort then divides the list into two sub lists, one with elements that are less than the list pivot and second list with elements greater than or equal to list pivot. Sort then recursively invokes itself with both lists. Each time when the sort invoked, it further divides the elements into smaller sub lists.

1.3 FLOW CHART / ALGORITHM :

QUICK SORT

Q_sort(A,first,last)

Let **A** is an array of **N** Elements

```
Step 1. low=first
          high =last
          pivot=A[(low+high)/2]
Step 2. Repeat tep7 while(low<=high)
Step 3. while( A[low]>pivot) repeat step4
Step 4. low=low+1
Step 5 . while( A[high]>pivot) repeat step6
Step 6. high=high-1
Step 7. if(low<=high)
          Temp=A[low]
       A[low]=A[high]
       A[high]=temp
       Low=low+1
       High=high-1
Step 8. if(first<high) then Q_sort(A, first, high)
Step 9. if(low<last) then Q_sort(A, low, last)
Step 10. Exit.
```

1.4 CODING:

```
#include<stdio.h>
#include<conio.h>
#define max 100
int a[max],n,i,l,h;
void main()
{
void input(void);
```

```
input();
getch();
}

void input(void)
{
void output(int a[],int n);
void quick_sort(int a[],int l,int h);
printf("How many elements in the array : ");
scanf("%d",&n);
printf("\n");
printf("Enter the elemennts : \n");
for(i=0;i<=n-1;i++)
{
scanf("%d",&a[i]);
}
l=0;
h=n-1;
quick_sort(a,l,h);
printf("Sorted Array :\n ");
output(a,n);
}

void quick_sort(int a[],int l, int h)
{
        int temp,key,low,high;
        low=l;
        high=h;
        key=a[(low+high)/2];
        do
        {
                while(key>a[low])
                {
                        low++;
                }
                while(key<a[high])
                {
                        high--;
                }
                if(low<=high)
                {
                        temp=a[low];
                        a[low++]=a[high];
                        a[high--]=temp;
                }
        }  while(low<=high);
        if(l<high)
        quick_sort(a,l,high);
        if(low<h)
        quick_sort(a,low,h);
```

```
}
void output(int a[],int n)
{
for(i=0;i<=n-1;i++)
{
printf("%d\n",a[i]);
}
}
```

1.5 OUTPUT:

```
How many elements
5
Enter the element of array
3 5 2 1 4
Element of array after the sorting are:
1 2 3 4 5
```