# SAMPLE OF LAB MANUAL

**R. D. ENGINEERING COLLEGE, DUHAI, GHAZIABAD**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

# B.Tech Second Year (Semester-III)

**Branch: CS/AIML/DS/IOT**

**LAB MANUAL**

# Data Structure Lab Using C (KCS-351)

R.D. Engineering College
Head
CSE
Ghaziabad

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

# B. TECH (COMPUTER SCIENCE & ENGINEERING)

## B. TECH(CSIT) 2nd YEAR

### SEMESTER-III

| SN | | Subject Code | Subject | Type | Category | Periods | | | Sessional Component | | Sessional (SW)(TS/PS) | End Semester Examination (ESE) | Total SW+ESE | Credit Cr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | L | T | P | CT | TA | CT+TA | TE/PE | | |
| 1 | | BOE3** / BAS303 | Science Based Open Elective/BSC (MathsIII/Math IV/ Math V) | T | ES/BS | 3 | 1 | 0 | 20 | 10 | 30 | 70 | 100 | 4 |
| 2 | | BVE301 / BAS301 | Universal Human Value and Professional Ethics/ Technical Communication | T | VA/HS | 2 | 1 | 0 | 20 | 10 | 30 | 70 | 100 | 3 |
| 3 | | BCS301 | Data Structure | T | PC | 3 | 1 | 0 | 20 | 10 | 30 | 70 | 100 | 4 |
| 4 | | BCS302 | Computer Organization and Architecture | T | PC | 3 | 1 | 0 | 20 | 10 | 30 | 70 | 100 | 4 |
| 5 | | BCS303 | Discrete Structures & Theory of Logic | T | PC | 2 | 1 | 0 | 20 | 10 | 30 | 70 | 100 | 3 |
| 6 | | BCS351 | Data Structure Lab | P | PC | 0 | 0 | 2 | | 50 | 50 | 50 | 100 | 1 |
| 7 | | BCS352 | Computer Organization and Architecture Lab | P | PC | 0 | 0 | 2 | | 50 | 50 | 50 | 100 | 1 |
| 8 | | BCS353 | Web Designing Workshop | P | PC | 0 | 0 | 2 | | 50 | 50 | 50 | 100 | 1 |
| 10 | | BCC301 / BCC302 | Cyber Security/Python programming | T | VA | 2 | 0 | 0 | 20 | 10 | 30 | 70 | 100 | 2 |
| 11 | | BCC351 | Internship Assessment /Mini Project* | P | | | | | | | 100 | | 100 | 2 |
| | | | Total | | | 15 | 5 | 6 | | | | | | 25 |

R.D. Engineering College Ghaziabad
Head CSE

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

# DATA STRUCTURER LAB (BCS-351) SYLLABUS

**List of Experiments**

| S. NO. | PROGRAM NAME |
|---|---|
| 1. | To implement addition and multiplication of two 2D arrays. |
| 2. | To transpose a 2D array. |
| 3. | To implement insertion, deletion and traversal of an element into singly linked list. |
| 4. | To implement stack using array. |
| 5. | To implement Queue using array. |
| 6. | To implement Circular queue using array. |
| 7. | To implement Linear Search. |
| 8. | To implement Binary Search. |
| 9. | To implement Bubble sort. |
| 10. | To implement Selection sort. |

ALUE ADDITION:

| | |
|---|---|
| 11. | To implement Insertion sort. |
| 12. | To implement Quick sort. |

R.D. Engineering College Ghaziabad Head CSE

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

## DATA STRUCTURER LAB (BCS-351) TEXT/REFERENCE BOOKS

List of standard / text / reference books, other study material / Web links

| S. No. |  | Description |
|---|---|---|
| 1 | Book | Horowitz and Sahani, "Fundamentals of Data Structures", Galgotia Publications Pvt Ltd Delhi India. |
| 2 | Book | Rajesh K. Shukla, "Data Structure Using C and C++" Wiley Dreamtech Publication. |
| 3 | Book | Thareja, "Data Structure Using C" Oxford Higher Education. |
| 4 | Website | www.javatpoint.com/data-structure-tutorial/ |
| 5 | Website | www.geeksforgeeks.org/data-structures/ |
|  |  |  |

R.D. Engineering College Ghaziabad Head CSE

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

# BASIC COURSE RELATED TO LAB

## 1.To implement addition and multiplication of two 2D arrays.

**Description**:

Here A is a two – dimensional array with M rows and N columns and B is a two – dimensional array with X rows and Y columns.

This algorithm adds these two arrays.

```
1. If (M ≠ X) or (N ≠ Y) Then
2. Print: Addition is not possible.
3. Exit [End of If]
4. Repeat For I = 1 to M
5. Repeat For J = 1 to N
6. Set C[I][J] = A[I][J] + B[I][J]
[End of Step 5 For Loop]
[End of Step 6 For Loop]
7. Exit
```

**Explanation**: First, we have to check whether the rows of array A are equal to the rows of array B or the columns of array A are equal to the columns of array B. if they are not equal, then addition is not possible and the algorithm exits. But if they are equal, then first for loop iterates to the total number of rows i.e. M and the second for loop iterates to the total number of columns i.e. N.

In step 6, the element A[I][J] is added to the element B[I][J] and is stored in C[I][J] by the statement: C[I][J] = A[I][J] + B[I][J]

**Multiply ( ):**

**Description**: Here A is a two – dimensional array with M rows and N columns and B is a two – dimensional array with X rows and Y columns.

This algorithm multiplies these two arrays.

```
1. If (M ≠ Y) or (N ≠ X) Then
2. Print: Multiplication is not possible.
3. Else
4. Repeat For I = 1 to N
```

R.D. Engineering College Ghaziabad Head CSE

Director
R.D. Engineering College
Duhai, Ghaziabad

```
5. Repeat For J = 1 to X
6. Set C[I][J] = 0
7. Repeat For K = 1 to Y
8. Set C[I][J] = C[I][J] + A[I][K] * B[K][J]
[End of Step 7 For Loop]
[End of Step 5 For Loop]
[End of Step 4 For Loop]
[End of If]
9. Exit
```

**Explanation**: First we check whether the rows of A are equal to columns of B or the columns of A are equal to rows of B. If they are not equal, then multiplication is not possible. But, if they are equal, the first for loop iterates to total number of columns of A i.e. N and the second for loop iterates to the total number of rows of B i.e. X. In step 6, all the elements of C are set to zero. Then the third for loop iterates to total number of columns of B i.e. Y. In step 8, the element A[I][K] is multiplied with B[K][J] and added to C[I][J] and the result is assigned to C[I][J] by the statement:

C[I][J] = C[I][J] + A[I][K] * B[K][J]

R.D. Engineering College Ghaziabad Head CSE

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

## 2.To transpose a 2D array.

**Transpose ( ):**

Description: Here A is a two – dimensional array with M rows and N columns. This algorithm transposes the array.

```
1. Repeat For I = 1 to M
    2. Repeat For J = 1 to N
        3. Set B[J][I] = A[I][J]
    [End of Step 2 For Loop]
[End of Step 1 For Loop]
4. Exit
```

**Explanation:**

The first for loop iterates from 1 to M i.e. total number of rows and second for loop iterates from 1 to N i.e. total number of columns. In step 3, the element at location A[I][J] is assigned to B[J][I] by the statement B[J][I] = A[I][J].

R.D. Engineering College * Ghaziabad * Head CSE

[illegible]

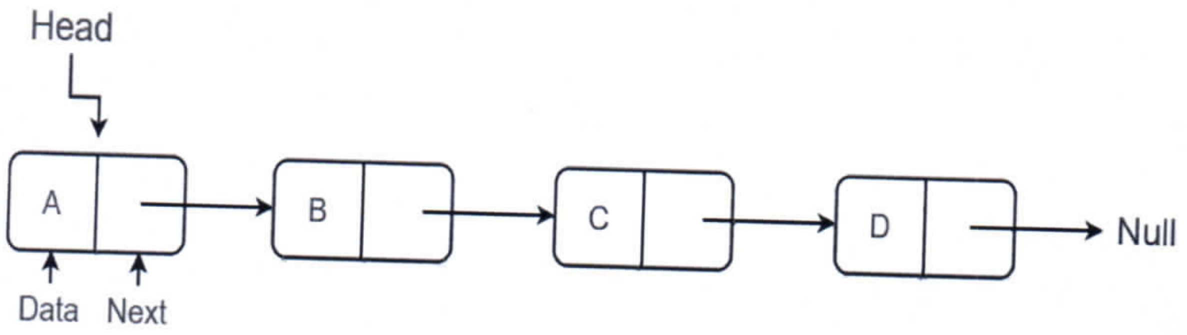
Director
R.D. Engineering College
Duhai, Ghaziabad

## 3. To implement insertion, deletion and traversal of an element into singly linked list.

A Linked List is a **linear data structure** which looks like a chain of nodes, where each node is a different element. Unlike Arrays, Linked List elements are not stored at a contiguous location.

It is basically **chains of nodes**, each node contains information such as **data** and a **pointer to the next node** in the chain. In the linked list there is a **head pointer**, which points to the first element of the linked list, and if the list is empty then it simply points to null or nothing.

*A **singly linked list** is a linear data structure in which the elements are not stored in contiguous memory locations and each element is connected only to its next element using a pointer.*



### A node creation:

```
// A Single linked list node
class Node {
public:
   int data;
  Node* next;
      };
```

R.D. Engineering College Head CSE Ghaziabad

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

# 4. To implement stack using array.

**STACK**: A stack is one of the most important and useful non-primitive linear data structure in computer science. It is an ordered collection of items into which new data items may be added/inserted and from which items may be deleted at only one end, called the top of the stack. As all the addition and deletion in a stack is done from the top of the stack, the last added element will be first removed from the stack. That is why the stack is also called Last-in-First-out (LIFO).

## STACK USING ARRAYS

Implementation of stack using arrays is a very simple technique. Algorithm for pushing (or add or insert) a new element at the top of the stack and popping (or delete) an element from the stack is given below.

Algorithm for push Suppose STACK[SIZE] is a one dimensional array for implementing the stack, which will hold the data items. TOP is the pointer that points to the top most element of the stack. Let DATA is the data item to be pushed.

```
1. If TOP = SIZE – 1, then:
    (a) Display "The stack is in overflow condition"
    (b) Exit
2. TOP = TOP + 1
3. STACK [TOP] = ITEM
4. Exit
```

Algorithm for pop Suppose STACK[SIZE] is a one dimensional array for implementing the stack, which will hold the data items. TOP is the pointer that points to the top most element of the stack. DATA is the popped (or deleted) data item from the top of the stack.

```
1. If TOP < 0, then
    (a) Display "The Stack is empty"
    (b) Exit
2. Else remove the Top most element
3. DATA = STACK[TOP]
4. TOP = TOP – 1        5.Exit
```

R.D. Engineering College • Ghaziabad • Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 5. To implement queue using array

**Queue:** We define a queue to be a list in which all additions to the list are made at one end, and all deletions from the list are made at the other end. The element which is first pushed into the order, the operation is first performed on that.

- A Queue is like a line waiting to purchase tickets, where the first person in line is the first person served. (i.e. First come first serve).
- Position of the entry in a queue ready to be served, that is, the first entry that will be removed from the queue, is called the **front** of the queue(sometimes, **head** of the queue), similarly, the position of the last entry in the queue, that is, the one most recently added, is called the **rear** (or the **tail**) of the queue.

### Algorithm:

Check if the queue is already full by comparing rear to max - 1. if so, then return an overflow error. If the item is to be inserted as the first element in the list, in that case set the value of front and rear to 0 and insert the element at the rear end.

Otherwise keep increasing the value of rear and insert each element one by one having rear as the index.

```
Step 1: IF REAR = MAX - 1
Write OVERFLOW
Go to step
[END OF IF]
Step 2: IF FRONT = -1 and REAR = -1
SET FRONT = REAR = 0
ELSE
SET REAR = REAR + 1
[END OF IF]
Step 3: Set QUEUE[REAR] = NUM
Step 4: EXIT
```

R.D. Engineering College • Ghaziabad • Head CSE

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

## 6. To implement circular queue using array

CIRCULAR QUEUE In circular queues the elements Q[0],Q[1],Q[2] .... Q[n – 1] is represented in a circular fashion with Q[1] following Q[n]. A circular queue is one in which the insertion of a new element is done at the very first location of the queue if the last location at the queue is full. Suppose Q is a queue array of 6 elements. Push and pop operation can be performed on circular. The following figures will illustrate the same.

**Inserting an element to circular Queue**

```
1. Initialize FRONT = – 1; REAR = 1
2. REAR = (REAR + 1) % SIZE
3. If (FRONT is equal to REAR) (a) Display "Queue is full" (b) Exit
4. Else (a) Input the value to be inserted and assign to variable "DATA"
5. If (FRONT is equal to – 1) (a) FRONT = 0 (b) REAR = 0
6. Q[REAR] = DATA
7. Repeat steps 2 to 5 if we want to insert more elements
8. Exit
```

**Deleting an element from a circular queue**

```
1. If (FRONT is equal to – 1) (a) Display "Queue is empty" (b) Exit
2. Else (a) DATA = Q[FRONT]
3. If (REAR is equal to FRONT) (a) FRONT = –1 (b) REAR = –1
4. Else (a) FRONT = (FRONT +1) % SIZE
5. Repeat the steps 1, 2 and 3 if we want to delete more elements
6. Exit
```

R.D. Engineering College Ghaziabad
Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

We will use an array of fixed size and maintain two variables **front** ( stores the index of the front element of the queue ) and **rear** ( stores the index of last element in the queue ). For **enqueing ( inserting )** an element, we increment **rear** and insert the element. If we have reached the end of the queue, then if there is a space available at front ( space might have been created due to some dequeue operation ), the element can be inserted there also. For **dequeing ( removing )** an element, we increment **front**.

## 7. To implement Linear Search.

**linear search** or **sequential search** is a method for finding a particular value in a list that checks each element in sequence until the desired element is found or the list is exhausted.[1] The list need not be ordered.

**Linear Search ( Array A, Value x)**

```
Step 1: Set i to 1
Step 2: if i > n then go to step 7
Step 3: if A[i] = x then go to step 6
Step 4: Set i to i + 1
Step 5: Go to Step 2
Step 6: Print Element x Found at index i and go to step 8
Step 7: Print element not found
Step 8: Exit
```

R.D. Engineering College • Ghaziabad
Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 8 . To implement Binary Search.

In a binary search we use the information that all the elements are sorted

```
binarySearch(A, x):
n = len(A)
beg = 0
end = n - 1
result = -1
while (beg <= end):
mid = (beg + end) / 2
if (A[mid] <= x):
beg = mid + 1
result = mid
else:
end = mid – 1
return result
```

R.D. Engineering College • Ghaziabad
Head
CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 9. To implement Bubble Sorting.

The **bubble sort** makes multiple passes through a list. It compares adjacent items and exchanges those that are out of order. Each pass through the list places the next largest value in its proper place. In essence, each item "bubbles" up to the location where it belongs.

```
procedure bubbleSort( A : list of sortable items )
  n = length(A)
  repeat
    swapped = false
    for i = 1 to n-1 inclusive do
      /* if this pair is out of order */
      if A[i-1] > A[i] then
        /* swap them and remember something changed */
        swap( A[i-1], A[i] )
        swapped = true
      end if
    end for
  until not swapped
end procedure
```

R.D. Engineering College Ghaziabad
Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 10. To implement Selection Sorting.

This type of sorting is called "Selection Sort" because it works by repeatedly element. It works as follows: first find the smallest in the array and exchange it with the element in the first position, then find the second smallest element and exchange it with the element in the second position, and continue in this way until the entire array is sorted.

```
for $i \leftarrow 1$ to $n$-1 do
  min $j \leftarrow i$;
  min $x \leftarrow A[i]$
  for $j \leftarrow i + 1$ to n do
    If $A[j] <$ min x then
      min $j \leftarrow j$
      min $x \leftarrow A[j]$
  A[min $j$] $\leftarrow$ A [$i$]
  A[$i$] $\leftarrow$ min $x$
```

R.D. Engineering College • Ghaziabad • Head CSE

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

## 11. To implement Insertion Sorting

Insertion sort iterates, consuming one input element each repetition, and growing a sorted output list. Each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain.

Sorting is typically done in-place, by iterating up the array, growing the sorted list behind it. At each array-position, it checks the value there against the largest value in the sorted list (which happens to be next to it, in the previous array-position checked). If larger, it leaves the element in place and moves to the next. If smaller, it finds the correct position within the sorted list, shifts all the larger values up to make a space, and inserts into that correct position.

The resulting array after *k* iterations has the property where the first *k* + 1 entries are sorted ("+1" because the first entry is skipped). In each iteration the first remaining entry of the input is removed, and inserted into the result at the correct position, thus extending the resul

```
for i = 1 to length(A) - 1
   x = A[i]
   j = i
   while j > 0 and A[j-1] > x
      A[j] = A[j-1]
      j = j - 1
   end while
   A[j] = x[3]
end for
```

R.D. Engineering College • Ghaziabad • Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 12.To implement Quick Sorting.

## Quick Sort Algorithm

In this article, we will discuss the Quicksort Algorithm. The working procedure of Quicksort is also simple. This article will be very helpful and interesting to students as they might face quicksort as a question in their examinations. So, it is important to discuss the topic.

Sorting is a way of arranging items in a systematic manner. Quicksort is the widely used sorting algorithm that makes **n log n** comparisons in average case for sorting an array of n elements. It is a faster and highly efficient sorting algorithm. This algorithm follows the divide and conquer approach. Divide and conquer is a technique of breaking down the algorithms into subproblems, then solving the subproblems, and combining the results back together to solve the original problem.

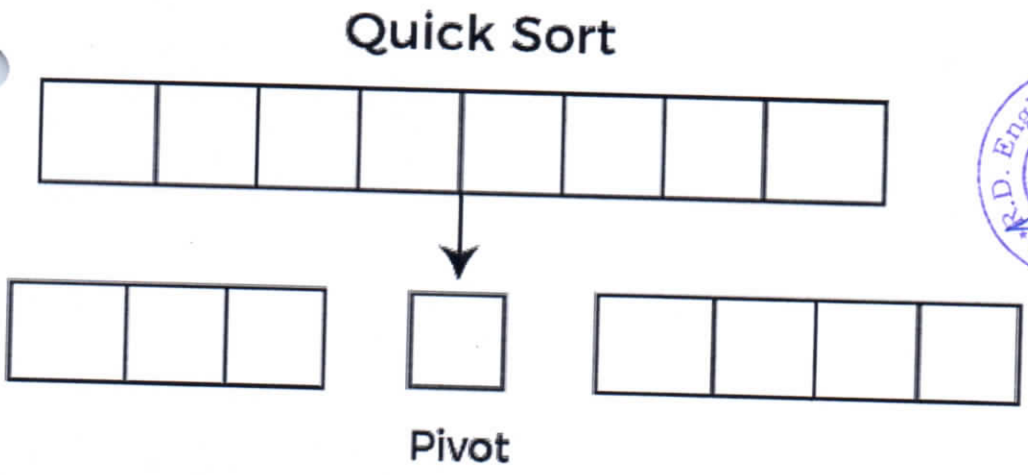
**Divide:** In Divide, first pick a pivot element. After that, partition or rearrange the array into two sub-arrays such that each element in the left sub-array is less than or equal to the pivot element and each element in the right sub-array is larger than the pivot element.

**Conquer:** Recursively, sort two subarrays with Quicksort.

**Combine:** Combine the already sorted array.

Quicksort picks an element as pivot, and then it partitions the given array around the picked pivot element. In quick sort, a large array is divided into two arrays in which one holds values that are smaller than the specified value (Pivot), and another array holds the values that are greater than the pivot.

After that, left and right sub-arrays are also partitioned using the same approach. It will continue until the single element remains in the sub-array.



R.D. Engineering College • Ghaziabad • Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

Choosing the pivot

Picking a good pivot is necessary for the fast implementation of quicksort. However, it is typical to determine a good pivot. Some of the ways of choosing a pivot are as follows –

- Pivot can be random, i.e. select the random pivot from the given array.
- Pivot can either be the rightmost element of the leftmost element of the given array.
- Select median as the pivot element.

Algorithm

**Algorithm:**

```
1. QUICKSORT (array A, start, end)
2. {
3.  1 if (start < end)
4.  2 {
5.  3 p = partition(A, start, end)
6.  4 QUICKSORT (A, start, p - 1)
7.  5 QUICKSORT (A, p + 1, end)
8.  6 }
9. }
```

**Partition Algorithm:**

The partition algorithm rearranges the sub-arrays in a place.

```
1. PARTITION (array A, start, end)
2. {
3.  1 pivot ? A[end]
4.  2 i ? start-1
5.  3 for j ? start to end -1 {
6.  4 do if (A[j] < pivot) {
7.  5 then i ? i + 1
8.  6 swap A[i] with A[j]
9.  7 }}
10. 8 swap A[i+1] with A[end]
11. 9 return i+1
12. }
```

R.D. Engineering College
Head
CSE
Ghaziabad

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

# DATA STRUCTURE (BCS-351) LAB MANUAL

1. To implement addition and multiplication of two 2D arrays.

```
#include<stdio.h>

void Matrix_Display(int a[][20],int n)
{
      int i,j;
      for(i=0; i<n; i++)
      {
       for(j=0; j<n; j++)
        {
         printf(" %d",a[i][j]);
        }
       printf("\n");
      }

}

int main()
{
     int n,i,j,k;
     int a[20][20];
     int b[20][20];
     int c[20][20];

    printf("\n Enter the dimensions of the 2 Square matrices: ");
    scanf("%d",&n);

    printf("\n Enter elements of Matrix A: ");
    for(i=0; i<n; i++)
    for(j=0; j<n; j++)
    scanf("%d",&a[i][j]);

    printf("\n Enter elements of Matrix B: ");
    for(i=0; i<n; i++)
    for(j=0; j<n; j++)
    scanf("%d",&b[i][j]);

    printf("\n Matrix A: \n");
    Matrix_Display(a,n);

    printf("\n\n Matrix B: \n");
    Matrix_Display(b,n);

    //Addition
    for(i=0; i<n; i++)
    for(j=0; j<n; j++)
    c[i][j]=a[i][j]+b[i][j];

    printf("\n\n Addition of A and B gives: \n");
    Matrix_Display(c,n);
```

R.D. Engineering College
Head
CSE
Ghaziabad

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

```
     //Subtraction
    for(i=0; i<n; i++)
    for(j=0; j<n; j++)
    c[i][j]=a[i][j]-b[i][j];

    printf("\n\n Subtraction of A and B gives: \n");
    Matrix_Display(c,n);

    //Multiplication
    for(i=0; i<n; i++)
    for(j=0; j<n; j++)
    {
     c[i][j]=0;
     for(k=0; k<n; k++)
     c[i][j]+=a[i][k]*b[k][j];
    }

    printf("\n\n Multiplication of A and B gives: \n");
    Matrix_Display(c,n);

}
```

**OUTPUT:**

```
 Enter the dimensions of the 2 Square matrices: 2


 Enter elements of Matrix A: 1 2 3 4


 Enter elements of Matrix B: 4 3 2 1


 Matrix A:

 1 2

 3 4

Matrix B:

 4 3

 2 1

 Addition of A and B gives:

 5 5

 5 5

Subtraction of A and B gives:

 -3 -1

 1 3

 Multiplication of A and B gives:

 8 5

 20 3
```

R.D. Engineering College Ghaziabad Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 2. To transpose a 2D array.

```
#include <stdio.h>

int main()
{
    int a[10][10], transpose[10][10], r, c, i, j;
    printf("Enter rows and columns of matrix: ");
    scanf("%d %d", &r, &c);

    // Storing elements of the matrix
    printf("\nEnter elements of matrix:\n");
    for(i=0; i<r; ++i)
        for(j=0; j<c; ++j)
        {
            printf("Enter element a%d%d: ",i+1, j+1);
            scanf("%d", &a[i][j]);
        }

    // Displaying the matrix a[][] */
    printf("\nEntered Matrix: \n");
    for(i=0; i<r; ++i)
        for(j=0; j<c; ++j)
        {
            printf("%d  ", a[i][j]);
            if (j == c-1)
                printf("\n\n");
        }

    // Finding the transpose of matrix a
    for(i=0; i<r; ++i)
        for(j=0; j<c; ++j)
        {
            transpose[j][i] = a[i][j];
        }
```

R.D. Engineering College Ghaziabad Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

```
    // Displaying the transpose of matrix a
    printf("\nTranspose of Matrix:\n");
    for(i=0; i<c; ++i)
        for(j=0; j<r; ++j)
        {
            printf("%d  ",transpose[i][j]);
            if(j==r-1)
                printf("\n\n");
        }

    return 0;
}
```

R.D. Engineering College Head CSE Ghaziabad

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

## 3. To implement insertion, deletion,traversal of an element into singly linked list.

```
#include<stdio.h>
struct node{
struct node *link;
int info;
};
struct node *start=NULL;
struct node* createNode(){
struct node *n;
n=(struct node *)=malloc(sizeof(struct node));
return (n);
}
void insertNode(){
struct node *temp, *t;
temp=createNode();
printf("Enter a number");
scanf("%d",&temp->info);
temp->link=NULL;
if(start==NULL)
start=temp;
else{
t=start;
while(t->link!=NULL){
t=t->link;
t->link=temp;
```

R.D. Engineering College • Ghaziabad • Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

```
                }
            }
    }
    void deleteNode(){
    struct node *r;
    if(start==NULL)
    printf("List is empty");
    else{
      r =start;
      start=start->link;
      free(r);
         }
    }
    void viewList(){
    struct node *t;
    if(start==NULL)
    printf("List is empty");
    else{
    t=start;
    while(t!=NULL){
    printf("%d",t->info);
    t=t->link;
          }
      }
    }
    int menu(){
```

R.D. Engineering College
Head
CSE
Ghaziabad

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

```
int ch;
printf("\n 1. Add the value to the list  ");
printf("\n 2. Delete first value");
printf("\n 3. View list ");
printf("\n 4. Exit");
printf("\n 5. Enter your choice");
scanf("%d",&ch);
return(ch);
}
void main(){
while(1){
clrscr();
switch(menu()){
case 1: insertNode();
        break;
case 2: deleteNode();
         break;
case 3: viewList();
         break;
case 4: exit(0);
default: printf("Invalid choice");
}
getch();
}
}
```

R.D. Engineering College * Ghaziabad
Head
CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 4. To implement Stack using Array.

```
#include<stdio.h>
#include<conio.h>
int stack[100],choice,n,top,x,i;
void push();
void pop();
void display();
void main()
{
    //clrscr();
    top=-1;
    printf("\n Enter the size of STACK[MAX=100]:");
    scanf("%d",&n);
    printf("\n\t STACK OPERATIONS USING ARRAY");
    printf("\n\t--------------------------------");
    printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
    do
    {
        printf("\n Enter the Choice:");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
        {
            push();
            break;
        }
        case 2:
        {
            pop();
            break;
        }
        case 3:
        {
            display();
            break;
        }
        case 4:
        {
            printf("\n\t EXIT POINT ");
            break;
        }
        default:
        {
            printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
        }
        getch();
```

R.D. Engineering College • Ghaziabad • Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

```
            }
        }
        while(choice!=4);
    }
    void push()
    {
        if(top>=n-1)
        {
            printf("\n\tSTACK is over flow");
            getch();
        }
        else
        {
            printf(" Enter a value to be pushed:");
            scanf("%d",&x);
            top++;
            stack[top]=x;
        }
    }
    void pop()
    {
        if(top<=-1)
        {
            printf("\n\t Stack is under flow");
        }
        else
        {
            printf("\n\t The popped elements is %d",stack[top])
            top--;
        }
    }
    void display()
    {
        if(top>=0)
        {
            printf("\n The elements in STACK \n");
            for(i=top; i>=0; i--)
                printf("\n%d",stack[i]);
            printf("\n Press Next Choice");
        }
        else
        {
            printf("\n The STACK is empty");
        }
    }
```

R.D. Engineering College Head CSE Ghaziabad [illegible]

[illegible] Director R.D. Engineering College Duhai, Ghaziabad

OUTPUT:

```
Enter the size of STACK[MAX=100]:10

         STACK OPERATIONS USING ARRAY
        --------------------------------
         1.PUSH
         2.POP
         3.DISPLAY
         4.EXIT
 Enter the Choice:1
 Enter a value to be pushed:12

 Enter the Choice:1
 Enter a value to be pushed:24

 Enter the Choice:1
 Enter a value to be pushed:98

 Enter the Choice:3

 The elements in STACK

98
24
12
 Press Next Choice
 Enter the Choice:2

        The popped elements is 98
 Enter the Choice:3

 The elements in STACK

24
12
 Press Next Choice
 Enter the Choice:4

        EXIT POINT
```

R.D. Engineering College Head CSE Ghaziabad

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 5. To implement queue using array.

```
#include<stdio.h>
#include<stdlib.h>
#define maxsize 5
void insert();
void delete();
void display();
int front = -1, rear = -1;
int queue[maxsize];
void main ()
{
   int choice;
  while(choice != 4)
   {
       printf("\n*************************Main Menu****************************\
n");
       printf("\n======================================
========================\n");
       printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\
n4.Exit\n");
       printf("\nEnter your choice ?");
       scanf("%d",&choice);
       switch(choice)
       {
          case 1:
        insert();
          break;
          case 2:
          delete();
         break;
         case 3:
         display();
         break;
        case 4:
         exit(0);
         break;
```

R.D. Engineering College * Ghaziabad * Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

```
            default:
            printf("\nEnter valid choice??\n");
          }
        }
    }
    void insert()
        {
          int item;
          printf("\nEnter the element\n");
          scanf("\n%d",&item);
          if(rear == maxsize-1)
          {
            printf("\nOVERFLOW\n");
            return;
          }
          if(front == -1 && rear == -1)
          {
            front = 0;
            rear = 0;
          }
          else
          {
            rear = rear+1;
          }
          queue[rear] = item;
          printf("\nValue inserted ");

      }
      void delete()
      {
        int item;
        if (front == -1 || front > rear)
        {
          printf("\nUNDERFLOW\n");
          return;

        }
```

R.D. Engineering College • Head CSE • Ghaziabad •

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

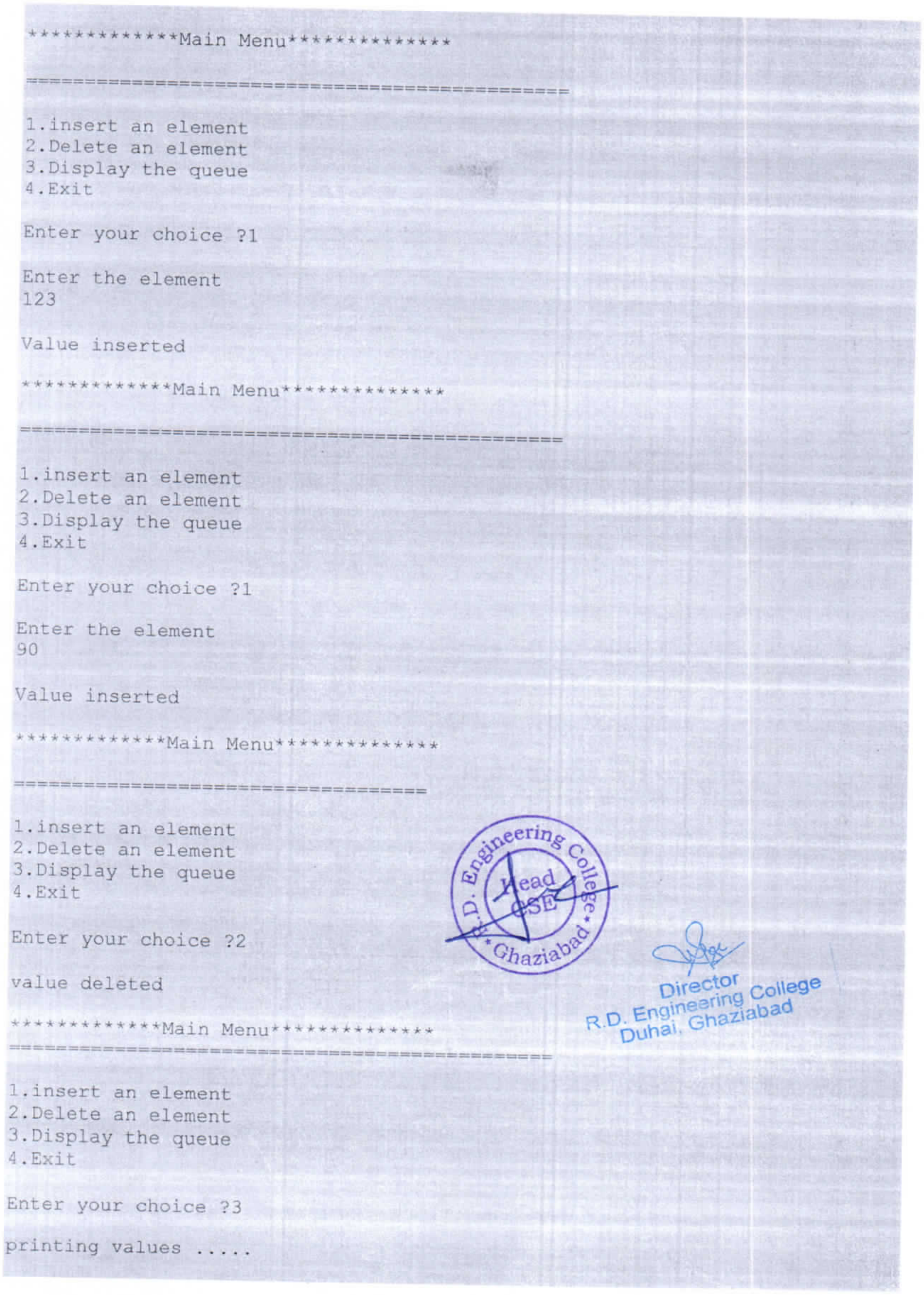
```
    else
    {
       item = queue[front];
       if(front == rear)
       {
          front = -1;
          rear = -1 ;
       }
       else
       {
          front = front + 1;
       }
       printf("\nvalue deleted ");
    }


}

void display()
{
   int i;
   if(rear == -1)
   {
      printf("\nEmpty queue\n");
   }
   else
   {  printf("\nprinting values .....\n");
      for(i=front;i<=rear;i++)
      {
         printf("\n%d\n",queue[i]);
      }
   }
}
```

R.D. Engineering College Ghaziabad Head CSE

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

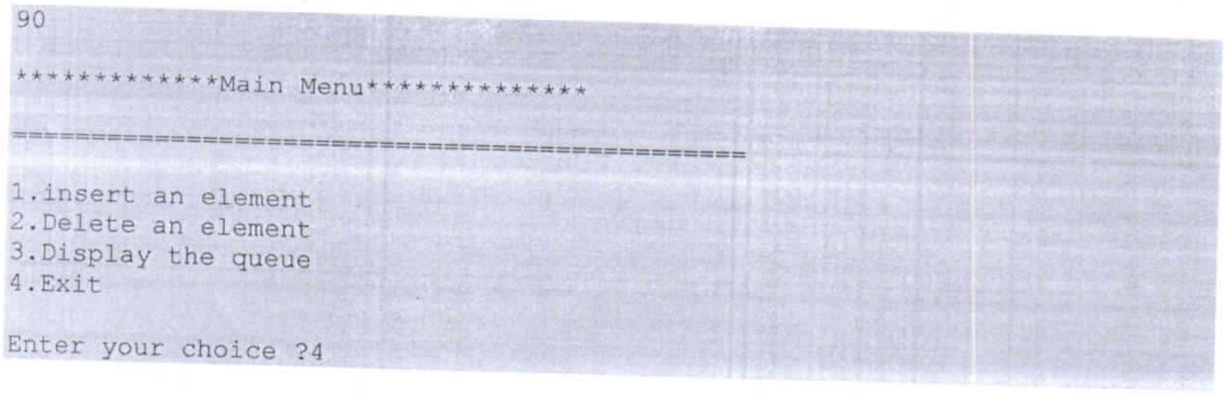
**Output:**



```
*************Main Menu**************

==============================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?1

Enter the element
123

Value inserted

*************Main Menu**************

==============================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?1

Enter the element
90

Value inserted

*************Main Menu**************

===================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?2

value deleted

*************Main Menu**************
==============================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?3

printing values .....
```

R.D. Engineering College * Ghaziabad *
Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad



```
90

*************Main Menu**************

===============================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?4
```

R.D. Engineering College
Head
CSE
*Ghaziabad*

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

## 6. To implement circular queue using array

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 10

int cqueue_arr[MAX];
int front=-1;
int rear=-1;

void display( );
void insert(int item);
int del();
int peek();
int isEmpty();
int isFull();

int main()
{
        int choice,item;
        while(1)
        {
                printf("\n1.Insert\n");
                printf("2.Delete\n");
                printf("3.Peek\n");
                printf("4.Display\n");
                printf("5.Quit\n");
                printf("\nEnter your choice : ");
                scanf("%d",&choice);

                switch(choice)
                {
                case 1 :
                        printf("\nInput the element for insertion : ");
                        scanf("%d",&item);
                        insert(item);
                        break;
                case 2 :
                        printf("\nElement deleted is : %d\n",del());
                        break;
                case 3:
                        printf("\nElement at the front is  : %d\n",peek());
                        break;
                case 4:
                        display();
                        break;
                case 5:
                        exit(1);
```

R.D. Engineering College • Ghaziabad • Head CSE

Director
R.D. Engineering College
Duhai, Ghaziabad

```
                default:
                        printf("\nWrong choice\n");
                }/*End of switch*/
        }/*End of while */

        return 0;

}/*End of main()*/

void insert(int item)
{
        if( isFull() )
        {
                printf("\nQueue Overflow\n");
                return;
        }
        if(front == -1 )
                front=0;

        if(rear==MAX-1)/*rear is at last position of queue*/
                rear=0;
        else
                rear=rear+1;
        cqueue_arr[rear]=item ;
}/*End of insert()*/

int del()
{
        int item;
        if( isEmpty() )
        {
                printf("\nQueue Underflow\n");
                exit(1);
        }
        item=cqueue_arr[front];
        if(front==rear) /* queue has only one element */
        {
                front=-1;
                rear=-1;
        }
        else if(front==MAX-1)
                front=0;
        else
                front=front+1;
        return item;
}/*End of del() */

int isEmpty()
{
        if(front==-1)
```

R.D. Engineering College * Ghaziabad * Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

```
                return 1;
        else
                return 0;
}/*End of isEmpty()*/

int isFull()
{
        if((front==0 && rear==MAX-1) || (front==rear+1))
                return 1;
        else
                return 0;
}/*End of isFull()*/

int peek()
{
        if( isEmpty() )
        {
                printf("\nQueue Underflow\n");
                exit(1);
        }
        return cqueue_arr[front];
}/*End of peek()*/

void display()
{
        int i;
        if(isEmpty())
        {
                printf("\nQueue is empty\n");
                return;
        }
        printf("\nQueue elements :\n");
        i=front;
        if( front<=rear )
        {
                while(i<=rear)
                        printf("%d ",cqueue_arr[i++]);
        }
        else
        {
                while(i<=MAX-1)
                        printf("%d ",cqueue_arr[i++]);
                i=0;
                while(i<=rear)
                        printf("%d ",cqueue_arr[i++]);
        }
        printf("\n");
}
```

R.D. Engineering College * Ghaziabad * Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 7. To implement Linear Search.

```
#include <stdio.h>

int main()
{
   int array[100], search, c, n;

   printf("Enter the number of elements in array\n");
   scanf("%d",&n);

   printf("Enter %d integer(s)\n", n);

   for (c = 0; c < n; c++)
      scanf("%d", &array[c]);

   printf("Enter the number to search\n");
   scanf("%d", &search);

   for (c = 0; c < n; c++)
   {
      if (array[c] == search)     /* if required element found */
      {
         printf("%d is present at location %d.\n", search, c+1);
         break;
      }
   }
   if (c == n)
      printf("%d is not present in array.\n", search);

   return 0;
}
```

R.D. Engineering College * Ghaziabad * Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 8. To implement Binary Search.

```
#include <stdio.h>

int main()
{
   int c, first, last, middle, n, search, array[100];

   printf("Enter number of elements\n");
   scanf("%d",&n);

   printf("Enter %d integers\n", n);

   for (c = 0; c < n; c++)
      scanf("%d",&array[c]);

   printf("Enter value to find\n");
   scanf("%d", &search);

   first = 0;
   last = n - 1;
   middle = (first+last)/2;

   while (first <= last) {
      if (array[middle] < search)
         first = middle + 1;
      else if (array[middle] == search) {
         printf("%d found at location %d.\n", search, middle+1);
         break;
      }
      else
         last = middle - 1;

      middle = (first + last)/2;
   }
   if (first > last)
      printf("Not found! %d is not present in the list.\n", search);

   return 0;
}
```

R.D. Engineering College * Ghaziabad * Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 9. To implement Bubble Sorting

```
#include <stdio.h>

int main()
{
  int array[100], n, c, d, swap;

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  for (c = 0 ; c < ( n - 1 ); c++)
  {
    for (d = 0 ; d < n - c - 1; d++)
    {
      if (array[d] > array[d+1]) /* For decreasing order use < */
      {
        swap       = array[d];
        array[d]   = array[d+1];
        array[d+1] = swap;
      }
    }
  }

  printf("Sorted list in ascending order:\n");

  for ( c = 0 ; c < n ; c++ )
     printf("%d\n", array[c]);

  return 0;
}
```

R.D. Engineering College Head CSE Ghaziabad

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

## 10. To implement Selection Sorting

```c
#include <stdio.h>

int main()
{
   int array[100], n, c, d, position, swap;

   printf("Enter number of elements\n");
   scanf("%d", &n);

   printf("Enter %d integers\n", n);

   for ( c = 0 ; c < n ; c++ )
      scanf("%d", &array[c]);

   for ( c = 0 ; c < ( n - 1 ) ; c++ )
   {
      position = c;

      for ( d = c + 1 ; d < n ; d++ )
      {
         if ( array[position] > array[d] )
            position = d;
      }
      if ( position != c )
      {
         swap = array[c];
         array[c] = array[position];
         array[position] = swap;
      }
   }

   printf("Sorted list in ascending order:\n");

   for ( c = 0 ; c < n ; c++ )
      printf("%d\n", array[c]);

   return 0;
}
```

R.D. Engineering College
Head
CSE
Ghaziabad

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 11. To implement Insertion Sorting

```
#include <stdio.h>

int main()
{
  int n, array[1000], c, d, t;

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++) {
    scanf("%d", &array[c]);
  }

  for (c = 1 ; c <= n - 1; c++) {
    d = c;

    while ( d > 0 && array[d-1] > array[d]) {
      t          = array[d];
      array[d]   = array[d-1];
      array[d-1] = t;

      d--;
    }
  }

  printf("Sorted list in ascending order:\n");

  for (c = 0; c <= n - 1; c++) {
    printf("%d\n", array[c]);
  }

  return 0;
}
```

R.D. Engineering College * Ghaziabad
Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

## 12. To implement Quick Sorting

```c
#include <stdio.h>
#define max 10

int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
int b[10];

void merging(int low, int mid, int high) {
   int l1, l2, i;

   for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
      if(a[l1] <= a[l2])
         b[i] = a[l1++];
      else
         b[i] = a[l2++];
   }

   while(l1 <= mid)
      b[i++] = a[l1++];

   while(l2 <= high)
      b[i++] = a[l2++];

   for(i = low; i <= high; i++)
      a[i] = b[i];
}

void sort(int low, int high) {
   int mid;

   if(low < high) {
      mid = (low + high) / 2;
      sort(low, mid);
      sort(mid+1, high);
      merging(low, mid, high);
   } else {
      return;
   }
}

int main() {
   int i;

   printf("List before sorting\n");

   for(i = 0; i <= max; i++)
      printf("%d ", a[i]);

   sort(0, max);
```

R.D. Engineering College • Ghaziabad • Head CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad

```
   printf("\nList after sorting\n");

   for(i = 0; i <= max; i++)
      printf("%d ", a[i]);
}
```

Output

```
List before sorting
10 14 19 26 27 31 33 35 42 44 0
List after sorting
0 10 14 19 26 27 31 33 35 42 44
```

R.D. Engineering College Ghaziabad Head CSE

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

## Viva Questions

1. What is data structure.
2. What is an algorithm.
3. What is time complexity.
4. What is space complexity.
5. What is the difference between linear and non-linear search.
6. What are the limitations of singly linked list.
7. What do you mean by overflow and underflow.
8. What is a priority queue.
9. List out few applications of tree data structure.
10. What is the difference between stack and queue.
11. What is a graph.
12. What is a spanning tree.
13. What do you mean by asymptotic notations.
14. Define time complexity of selection sort and quick sort.
15. What is the time complexity of bubble sort.
16. What is the difference between selection sort and insertion sort.
17. Are linked lists considered linear or non-linear data structure.
18. What are the advantages of a linked list over an array? In which scenarios do we use linked list and when array?
19. Why do we need to do an algorithm analysis.
20. Where are stacks used.
21. What is sequential search.
22. What are the types of hashing.
23. Define hashing.

R.D. Engineering College • Ghaziabad
Head CSE

[illegible]

Director
R.D. Engineering College
Duhai, Ghaziabad

24. Advantage of quick sort.

25.Define hash table.

26.Define insertion sort.

27.Define radix sort.

28. What are the postfix and prefix forms of the expression.

29. How do you test for an empty queue.

30.What are the advantages of linked list.

R.D. Engineering College • Ghaziabad
Head
CSE

[illegible]
Director
R.D. Engineering College
Duhai, Ghaziabad