| S.NO | Programs | Remarks |
|---|---|---|
| 1. | Write a Program to Perform different operation on Array | |
| 2. | Write a program on linear search | |
| 3. | Write a program on binary search | |
| 4. | Write a program on stack | |
| 5. | Write a program on array implementation of circular queue | |
| 6. | Write a program on array implementation of deque | |
| 7. | Write a program to perform different operation on linked list | |
| 8. | Write a program on bubble sort | |
| 9. | Write a program on selection sort | |
| 10. | Write a program on insertion sort | |

# Program 1

```c
// Different operations on array

#include <stdio.h>
#include <stdlib.h>
// declaration

void Insertion(), Deletion(), Traversal(), exit();
int a[10], n, i, s, p, choice;
void main()

{
    printf("Enter the numbers of elements :");
    scanf("%d", &n);

    printf("Enter the elements");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    printf("Elements are: ");
    for (i = 0; i < n; i++)
        printf("\na[%d]:%d", i, a[i]);

    // calling

    do
    {
        printf("\n1.Insertion \n2.Deletion \n3.Traversal \n4.exit");
        printf("\n Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            Insertion();
            break;

        case 2:
            Deletion();
            break;

        case 3:
            Traversal();
            break;

        case 4:
            exit(0);
            break;

        default:
            printf("Invalid choice\n");
        }

```

```
    } while (choice != 4);
}

// definition

void Insertion()
{
    printf("Enter the element and position to insert at:");
    scanf("%d %d", &s, &p);

    for (i = n - 1; i >= p; i--)
        a[i + 1] = a[i];
    n = n + 1;
    a[p] = s;

    printf("Array after Insertion \n");
    for (i = 0; i < n; i++)
        printf("\n a[%d]:%d", i, a[i]);
}

void Deletion()
{
    printf("Enter the position number to delete :");
    scanf("%d", &p);

    s = a[p];
    for (i = p; i <= n - 2; i++)
        a[i] = a[i + 1];
    n = n - 1;

    printf("Array after Deletion \n");
    for (i = 0; i < n; i++)

        printf("\n a[%d]:%d", i, a[i]);
}

void Traversal()
{
    for (i = 0; i < n; i++)
        printf("\n a[%d] :%d", i, a[i]);
}

```

# Output :

```
Enter the numbers of elements :5
Enter the elements11
22
33
44
55
Elements are:
a[0]:11
a[1]:22
a[2]:33
a[3]:44
a[4]:55
1.Insertion
2.Deletion
3.Traversal
4.exit
 Enter your choice: 1
Enter the element and position to insert at:66 5
Array after Insertion

 a[0]:11
 a[1]:22
 a[2]:33
 a[3]:44
 a[4]:55
 a[5]:66
1.Insertion
2.Deletion
3.Traversal
4.exit
 Enter your choice: 2
Enter the position number to delete :5
Array after Deletion

 a[0]:11
 a[1]:22
 a[2]:33
 a[3]:44
 a[4]:55
1.Insertion
2.Deletion
3.Traversal
4.exit
 Enter your choice: 3

 a[0] :11
 a[1] :22
 a[2] :33
 a[3] :44
 a[4] :55
1.Insertion
2.Deletion
3.Traversal
4.exit
 Enter your choice: 4
PS C:\Users\tushar\Desktop\DSA File\Code Files>
```

# Program 2

```c
//  Linear search

#include <stdio.h>

void main()
{
    int a[10], i, n, flag = 0, item, loc;

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

        printf("\n a[%d]:%d", i, a[i]);
    printf("\n Enter the elements to be searched : ");
    scanf("%d", &item);

    for (i = 0; i < n; i++)
    {
        if (a[i] == item)

        {
            loc = i;
            flag = flag + 1;
            printf("Data is found on %d position", loc);
            break;
        }
    }
    if (flag == 0)
    {
        printf("Data is not found");
    }
}
```

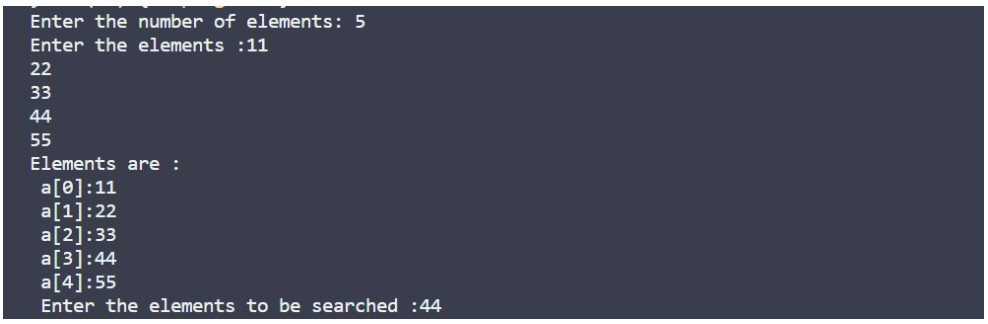
# Output :

```
Enter the number of elements :5
Enter the elements :11
22
33
44
55

 a[0]:11
 a[1]:22
 a[2]:33
 a[3]:44
 a[4]:55
 Enter the elements to be searched : 22
Data is found on 1 position
```

# Program 3

```c
// Binary search

#include <stdio.h>

void main()
{
    int a[100], i, n, item, loc, beg, end, mid;

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

    printf("Enter the elements :");
    for (i = 0; i < n; i++)
        scanf("\n %d", &a[i]);
    printf("Elements are :");
    for (i = 0; i < n; i++)
        printf("\n a[%d]:%d", i, a[i]);
    printf("\n Enter the elements to be searched :");
    scanf("%d", &item);

    loc = 0;
    beg = 0;
    end = n - 1;
    mid = ((beg + end) / 2);
    while ((beg <= end) && (item != a[mid]))
    {
        if (item < a[mid])
            end = mid - 1;

        else
            beg = mid + 1;
        mid = ((beg + end) / 2);
    }
    if (item == a[mid])
    {
        loc = mid;
        printf("\n Data is found on %d position", loc);
    }
    else
    {
        printf("Search is unseccesfull");
    }
}
```

# Output :


Enter the number of elements: 5
Enter the elements :11
22
33
44
55
Elements are :
a[0]:11
a[1]:22
a[2]:33
a[3]:44
a[4]:55
Enter the elements to be searched :44

# Program 4

```c
// Array Implementation of Stack

#include <stdio.h>

int stack[100], choice, n, top, x, i;
void push(void);
void pop(void);
void traversal(void);
int main()
{
    top = -1;
    printf("\n Enter the size of STACK :");
    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.Traversal\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:
        {
            traversal();
            break;
        }
        case 4:
        {
            printf("\n\t EXIT ");
            break;
        }
        default:
        {
            printf("\n\t Invalid choice ");
        }
```

```
        }
    } while (choice != 4);
    return 0;
}
void push()
{
    if (top >= n - 1)
    {
        printf("\n\tSTACK is Overflow");
    }
    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 Underflow");
    }
    else
    {
        printf("\n\t The popped elements is %d", stack[top]);
        top--;
    }
}
void traversal()
{
    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");
    }
}

```

# Output :

```
Enter the size of STACK :6

        STACK OPERATIONS USING ARRAY
       --------------------------------
        1.PUSH
        2.POP
        3.Traversal
        4.EXIT
Enter the Choice:1
Enter a value to be pushed:11

Enter the Choice:1
Enter a value to be pushed:22

Enter the Choice:1
Enter a value to be pushed:33

Enter the Choice:1
Enter a value to be pushed:44

Enter the Choice:1
Enter a value to be pushed:55

Enter the Choice:1
Enter a value to be pushed:66

Enter the Choice:2

        The popped elements is 66
Enter the Choice:3

The elements in STACK

55
44
 33
 22
 11
  Press Next Choice
  Enter the Choice:4

          EXIT
O PS C:\Users\tushar\Desktop\DSA File\Code Files>
```

# Program 6

```c
// Array implementation of circular Queue

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

void insertion();
int deletion();
void traversal();
int queue[5], front = 0, rear = 0, i, choice, n = 5;

void main()
{
    do
    {
        printf("\n1.insertion \n2.deletion \n3.traversal \n4.exit");
        printf("\n Enter your choice:");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            insertion();
            break;

        case 2:
            deletion();
            break;

        case 3:
            traversal();
            break;

        case 4:
            exit(0);
            break;

        default:
            printf("Invalid choice\n");
        }
    } while (choice != 4);
}

void insertion()
{
    int item;
    if ((front == 1 && rear == n) || (front == rear + 1))
        printf("overflow");
    else
    {
        printf("Enter the element to be inserted: ");
        scanf("%d", &item);
```

```
        if (front == 0)
            front = rear = 1;
        else if (rear == n)
            rear = 1;
        else
            rear = rear + 1;
        queue[rear] = item;
    }
}

int deletion()
{
    int item;
    if (front == 0)
        printf("\n underflow");
    else
    {
        item = queue[front];
        if ((front == rear) != 0)
            front = rear = 0;
        else if (front == n)
            front = 1;
        else
            front = front + 1;
        printf("\n The deleted item is: %d", item);
    }
    return (item);
}

void traversal()
{
    int f_pos = front, r_pos = rear;
    if (front == 0)
    {
        printf("Queue is empty");
        return;
    }
    if (f_pos <= r_pos)
    {
        for (i = f_pos; i <= r_pos; i++)
            printf("\n %d", queue[i]);
    }

    else
    {
        for (i = f_pos; i <= n; i++)
            printf("\n %d", queue[i]);
        f_pos = 1;
        for (i = f_pos; i <= r_pos; i++)
            printf("\n %d", queue[i]);
    }
}
```

# Output :

```
1.insertion
2.deletion
3.traversal
4.exit
 Enter your choice:1
Enter the element to be inserted: 11

1.insertion
2.deletion
3.traversal
4.exit
 Enter your choice:1
Enter the element to be inserted: 22

1.insertion
2.deletion
3.traversal
4.exit
 Enter your choice:1
Enter the element to be inserted: 33

1.insertion
2.deletion
3.traversal
4.exit
 Enter your choice:1
Enter the element to be inserted: 44

1.insertion
2.deletion
3.traversal
4.exit
 Enter your choice:2

 The deleted item is: 11
1.insertion
2.deletion
3.traversal
4.exit
 Enter your choice:2

 The deleted item is: 22
1.insertion
2.deletion
3.traversal
4.exit
 Enter your choice:3

 33
 44
1.insertion
2.deletion
3.traversal
4.exit
 Enter your choice:4
PS C:\Users\tushar\Desktop\DSA File\Code Files>
```

# Program 7

```c
// Array implementation of  Deque

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

void insert_rear();
void insert_front();
int delete_front();
int delete_rear();
void traversal();
int queue[5], front = 0, rear = 0, i, choice, n = 5;

void main()
{
    do
    {
        printf("\n1.Insert at rear \n2.Insert at front \n3.Delete at fro
\n4.Delete at rear \n5.Traversal  \n6.Exit");
        printf("\n Enter your choice:");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            insert_rear();
            break;
        case 2:
            insert_front();
            break;
        case 3:
            delete_front();
            break;
        case 4:
            delete_rear();
            break;
        case 5:
            traversal();
            break;
        case 6:
            exit(0);
            break;
        default:
            printf("Invalid choice \n");
        }
    } while (choice != 6);
}

void insert_rear()
```

```
{
    int item;
    if ((front == 1 && rear == n) || (front == rear + 1))
        printf("Overflow");
    else
    {
        printf("Enter the element to be inserted:");
        scanf("%d", &item);
        if (front == 0)
            front = rear = 1;
        else if (rear == n)
            rear = 1;
        else
            rear = rear + 1;
        queue[rear] = item;
    }
}

void insert_front()
{
    int item;
    if ((front == 1 && rear == n) || (front == rear + 1))
        printf("Overflow");
    else
    {
        printf("Enter the element to be inserted:");
        scanf("%d", &item);
        if (front == 0)
            front = rear = 1;
        else if (front == 1)
            front = n;
        else
            front = front - 1;
        queue[front] = item;
    }
}

int delete_front()
{
    int item;
    if (front == 0)
        printf("Underflow");
    else
    {
        item = queue[front];
        if ((front == rear) != 0)
            front = rear = 0;
        else if (front == n)
            front = 1;
        else
```

```
            front = front + 1;
        printf("\n The deleted item is:%d", item);
    }
    return (item);
}

int delete_rear()
{
    int item;
    if (front == 0)
        printf("Underflow");
    else
    {
        item = queue[rear];
        if ((front == rear) != 0)
            front = rear = 0;
        else if (rear == 1)
            rear = n;
        else
            rear = rear - 1;
        printf("\n The deleted item is:%d", item);
    }
    return (item);
}

void traversal()
{
    int f_pos = front, r_pos = rear;
    if (front == 0)
    {
        printf("Queue is empty");
        return;
    }

    if (f_pos <= r_pos)
    {
        for (i = f_pos; i <= r_pos; i++)
            printf("\n %d", queue[i]);
    }

    else
    {
        for (i = f_pos; i <= n; i++)
            printf("\n %d", queue[i]);
        f_pos = 1;
        for (i = f_pos; i <= r_pos; i++)
            printf("\n %d", queue[i]);
    }
}
```

# Output :

```
1.Insert at rear
2.Insert at front
3.Delete at front
4.Delete at rear
5.Traversal
6.Exit
 Enter your choice:1
Enter the element to be inserted:11

1.Insert at rear
2.Insert at front
3.Delete at front
4.Delete at rear
5.Traversal
6.Exit
 Enter your choice:1
Enter the element to be inserted:22

1.Insert at rear
2.Insert at front
3.Delete at front
4.Delete at rear
5.Traversal
6.Exit
 Enter your choice:2
Enter the element to be inserted:33

1.Insert at rear
2.Insert at front
3.Delete at front
4.Delete at rear
5.Traversal
6.Exit
 Enter your choice:2
Enter the element to be inserted:44

1.Insert at rear
2.Insert at front
3.Delete at front
4.Delete at rear
5.Traversal
6.Exit
 Enter your choice:3

 The deleted item is:44
1.Insert at rear
2.Insert at front
3.Delete at front
4.Delete at rear
5.Traversal
6.Exit
 Enter your choice:4

 The deleted item is:22
1.Insert at rear
2.Insert at front
3.Delete at front
4.Delete at rear
5.Traversal
6.Exit
 Enter your choice:5

 5
 11
1.Insert at rear
2.Insert at front
3.Delete at front
4.Delete at rear
5.Traversal
6.Exit
 Enter your choice:6
PS C:\Users\tushar\Desktop\DSA File\Code Files>
```

# Program 8

```c
// Different operations on link list

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

struct node
{
    int info;
    struct node *link;
};
struct node *start = NULL;

void insertbeg();
void insertafter(int, int);
void deleteafter();
void deletelist();
int traversal();
int choice, pos, data;

void main()
{
    do
    {
        printf("\n1.Insert at beg \n2.Insert after pos \n3.Delete after pos
\n4.Delete-list \n5.Traversal \n6.Exit");
        printf("\nEnter your choice :");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            insertbeg();
            break;

        case 2:
            insertafter(pos, data);
            break;

        case 3:
            deleteafter();
            break;

        case 4:
            deletelist();
            break;

        case 5:
            traversal();
            break;

```

```
        case 6:
            exit(0);
            break;

        default:
            printf("invalid choice \n");
        }
    } while (choice != 6);
}

void insertbeg()
{
    int data;
    struct node *new;
    printf("Enter the info of node :");
    scanf("%d", &data);
    new = (struct node *)malloc(sizeof(struct node));
    new->info = data;
    new->link = start;
    start = new;
}

void insertafter(int pos, int data)
{
    int i;
    struct node *new;
    struct node *ptr = start;

    printf("Enter the position after which element is inserted :");
    scanf("%d", &pos);
    for (i = 0; i < pos; i++)
    {
        ptr = ptr->link;
        if (ptr == NULL)
        {
            printf("There are less than %d elements", pos);
            return;
        }
    }
    printf("Enter the info of node:");
    scanf("%d", &data);
    new = (struct node *)malloc(sizeof(struct node));
    new->info = data;
    new->link = ptr->link;
    ptr->link = new;
}

void deleteafter()
{
    int i;
    struct node *save = start;
```

```
    struct node *ptr = start->link;
    struct node *locp = NULL;
    struct node *loc = NULL;

    printf("Enter the positio after which element is deleted :");
    scanf("%d", &pos);
    for (i = 0; i < pos; i++)
    {
        save = ptr;
        ptr = ptr->link;
    }
    locp = save;
    loc = ptr;
    if (pos == 0)
    {
        start = start->link;
        return;
    }
    else
        locp->link = loc->link;
    free(loc);
}

void deletelist()
{
    struct node *ptr;
    while (start != NULL)
    {
        ptr = start;
        start = start->link;
        free(ptr);
    }
}

int traversal()
{
    int count = 0;
    struct node *ptr = start;
    printf("\n The linked list is :\n\n");
    printf("->");
    while (ptr)
    {
        printf("%d->", ptr->info);
        ptr = ptr->link;
        count++;
    }
    printf("NULL\n");
    printf("\nNumber of elements in linked list: %d,count");
    return (count);
}
```

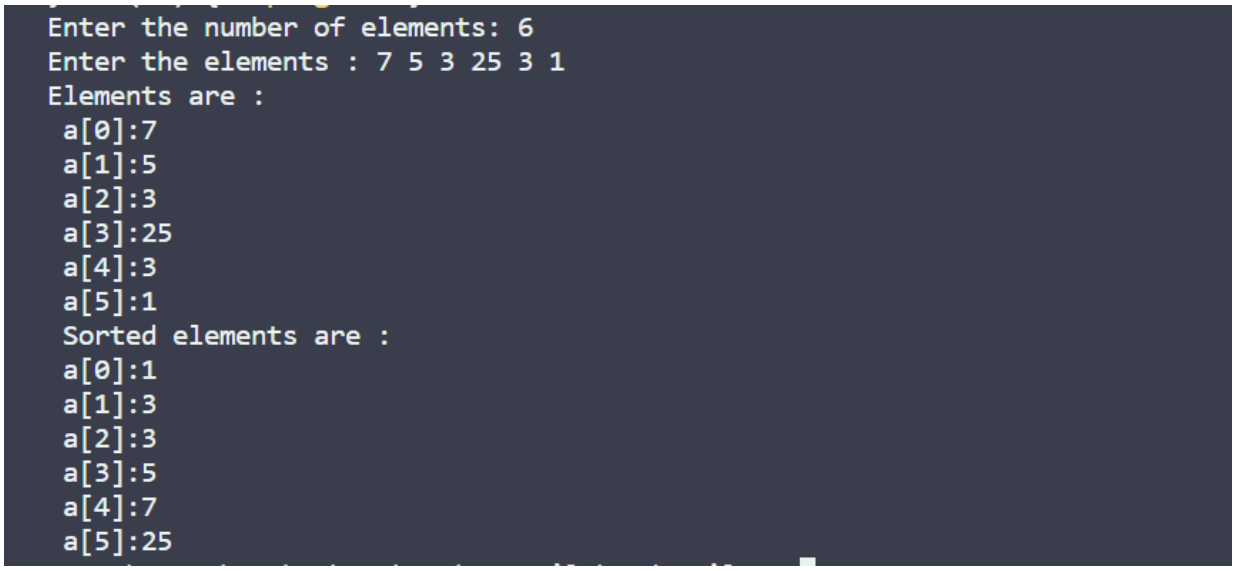
# Output :

```
1.Insert at beg
2.Insert after pos
3.Delete after pos
4.Delete-list
5.Traversal
6.Exit
Enter your choice :1
Enter the info of node :11

1.Insert at beg
2.Insert after pos
3.Delete after pos
4.Delete-list
5.Traversal
6.Exit
Enter your choice :1
Enter the info of node :22

1.Insert at beg
2.Insert after pos
3.Delete after pos
4.Delete-list
5.Traversal
6.Exit
Enter your choice :2
Enter the position after which element is inserted :1
Enter the info of node:33

1.Insert at beg
2.Insert after pos
3.Delete after pos
4.Delete-list
5.Traversal
6.Exit
Enter your choice :3
Enter the positio after which element is deleted :1

1.Insert at beg
2.Insert after pos
3.Delete after pos
4.Delete-list
5.Traversal
6.Exit
Enter your choice :5

 The linked list is :

->22->11->NULL

Number of elements in linked list: 11,count
1.Insert at beg
2.Insert after pos
3.Delete after pos
4.Delete-list
5.Traversal
6.Exit
Enter your choice :6
PS C:\Users\tushar\Desktop\DSA File\Code Files>
```

# Program 9

```c
// Bubble sort

#include <stdio.h>

void bubble_sort(int a[], int);
void main()
{
    int i, n, a[20];

    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Enter the elements : ");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    printf("Elements are :");

    for (i = 0; i < n; i++)
        printf("\n a[%d]:%d", i, a[i]);
    bubble_sort(a, n);
    printf("\n Sorted elements are :");
    for (i = 0; i < n; i++)
        printf("\n a[%d]:%d", i, a[i]);
}

void bubble_sort(int a[], int n)
{
    int pass, j, temp;
    for (pass = 1; pass < n; pass++)
    {
        for (j = 0; j <= n - pass - 1; j++)
        {
            if (a[j] > a[j + 1])
            {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }
}

```

# Output :


Enter the number of elements: 6
Enter the elements : 7 5 3 25 3 1
Elements are :
a[0]:7
a[1]:5
a[2]:3
a[3]:25
a[4]:3
a[5]:1
Sorted elements are :
a[0]:1
a[1]:3
a[2]:3
a[3]:5
a[4]:7
a[5]:25

# Program 10

```c
//  Selection sort

#include <stdio.h>

void Selection_sort(int a[], int);
void main()
{
    int i, n, a[20];

    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Enter the elements : ");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    printf("Elements are :");

    for (i = 0; i < n; i++)
        printf("\n a[%d]:%d", i, a[i]);
    selection_sort(a, n);
    printf("\n Sorted elements are :");
    for (i = 0; i < n; i++)
        printf("\n a[%d]:%d", i, a[i]);
}

void selection_sort(int a[], int n)
{
    int min, loc, temp, k, j;
    min = a[0];
    for (k = 0; k < n - 1; k++)
    {
        min = a[k];
        loc = k;
        for (j = k + 1; j <= n - 1; j++)
        {
            if (min > a[j])
            {
                min = a[j];
                loc = j;
            }
        }
        if (loc != k)
        {
            temp = a[k];
            a[k] = a[loc];
            a[loc] = temp;
        }
    }
}
```

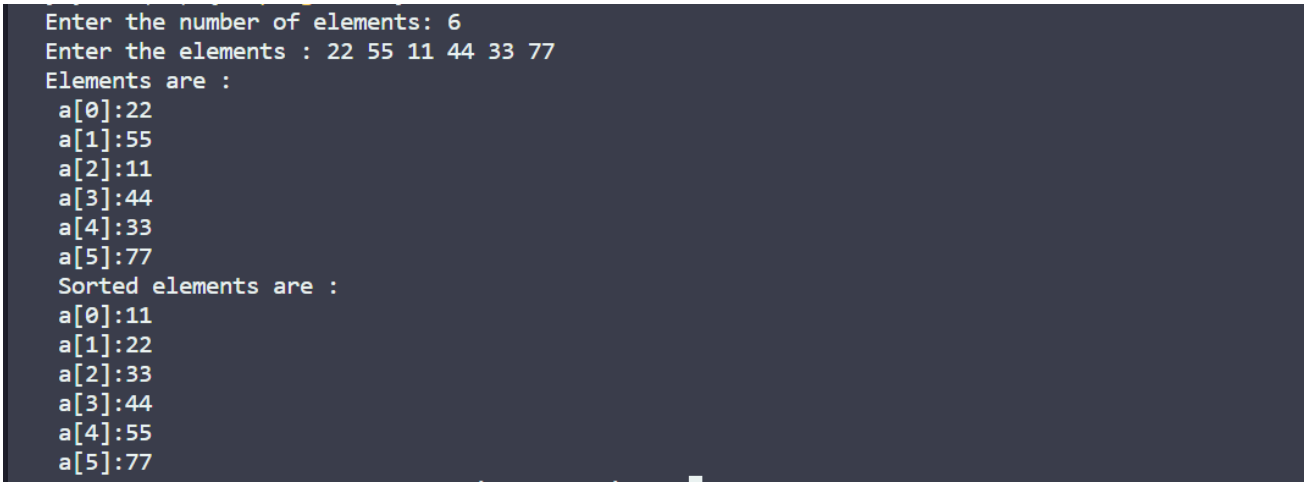
# Output :

```
Enter the number of elements: 6
Enter the elements : 11 44 22 66 77 33
Elements are :
 a[0]:11
 a[1]:44
 a[2]:22
 a[3]:66
 a[4]:77
 a[5]:33
 Sorted elements are :
 a[0]:11
 a[1]:22
 a[2]:33
 a[3]:44
 a[4]:66
 a[5]:77
```

# Program 11

```c
//  Insertion sort

#include <stdio.h>

void insertion_sort(int a[], int);
void main()
{
    int i, n, a[20];

    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Enter the elements : ");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    printf("Elements are :");

    for (i = 0; i < n; i++)
        printf("\n a[%d]:%d", i, a[i]);
    insertion_sort(a, n);
    printf("\n Sorted elements are :");
    for (i = 0; i < n; i++)
        printf("\n a[%d]:%d", i, a[i]);
}

void insertion_sort(int a[], int n)
{
    int temp, i, j, k;
    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;
    }
}
```

# Output :


Enter the number of elements: 6
Enter the elements : 22 55 11 44 33 77
Elements are :
a[0]:22
a[1]:55
a[2]:11
a[3]:44
a[4]:33
a[5]:77
Sorted elements are :
a[0]:11
a[1]:22
a[2]:33
a[3]:44
a[4]:55
a[5]:77