

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

```
1 // Different operations on array
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 // declaration
6
7 void Insertion(), Deletion(), Traversal(), exit();
8 int a[10], n, i, s, p, choice;
9 void main()
10 {
11     printf("Enter the numbers of elements :");
12     scanf("%d", &n);
13
14     printf("Enter the elements");
15     for (i = 0; i < n; i++)
16         scanf("%d", &a[i]);
17     printf("Elements are: ");
18     for (i = 0; i < n; i++)
19         printf("\na[%d]:%d", i, a[i]);
20
21     // calling
22
23     do
24     {
25         printf("\n1.Insertion \n2.Deletion \n3.Traversal \n4.exit");
26         printf("\n Enter your choice: ");
27         scanf("%d", &choice);
28
29         switch (choice)
30         {
31             case 1:
32                 Insertion();
33                 break;
34
35             case 2:
36                 Deletion();
37                 break;
38
39             case 3:
40                 Traversal();
41                 break;
42
43             case 4:
44                 exit(0);
45                 break;
46
47             default:
48                 printf("Invalid choice\n");
49         }
50     }
51 }
```

```
52     } while (choice != 4);
53 }
54
55 // definition
56
57 void Insertion()
58 {
59     printf("Enter the element and position to insert at:");
60     scanf("%d %d", &s, &p);
61
62     for (i = n - 1; i >= p; i--)
63         a[i + 1] = a[i];
64     n = n + 1;
65     a[p] = s;
66
67     printf("Array after Insertion \n");
68     for (i = 0; i < n; i++)
69         printf("\n a[%d]:%d", i, a[i]);
70 }
71
72 void Deletion()
73 {
74     printf("Enter the position number to delete :");
75     scanf("%d", &p);
76
77     s = a[p];
78     for (i = p; i <= n - 2; i++)
79         a[i] = a[i + 1];
80     n = n - 1;
81
82     printf("Array after Deletion \n");
83     for (i = 0; i < n; i++)
84
85         printf("\n a[%d]:%d", i, a[i]);
86 }
87
88 void Traversal()
89 {
90     for (i = 0; i < n; i++)
91         printf("\n a[%d] :%d", i, a[i]);
92 }
93
```

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

Program 2

```
1 // Linear search
2
3 #include <stdio.h>
4
5 void main()
6 {
7     int a[10], i, n, flag = 0, item, loc;
8
9     printf("Enter the number of elements :");
10    scanf("%d", &n);
11    printf("Enter the elements :");
12    for (i = 0; i < n; i++)
13        scanf("%d", &a[i]);
14    for (i = 0; i < n; i++)
15
16        printf("\n a[%d]:%d", i, a[i]);
17    printf("\n Enter the elements to be searched : ");
18    scanf("%d", &item);
19
20    for (i = 0; i < n; i++)
21    {
22        if (a[i] == item)
23
24            {
25                loc = i;
26                flag = flag + 1;
27                printf("Data is found on %d position", loc);
28                break;
29            }
30    }
31    if (flag == 0)
32    {
33        printf("Data is not found");
34    }
35 }
```

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

```
1 // Binary search
2
3 #include <stdio.h>
4
5 void main()
6 {
7     int a[100], i, n, item, loc, beg, end, mid;
8
9     printf("Enter the number of elements: ");
10    scanf("%d", &n);
11
12    printf("Enter the elements :");
13    for (i = 0; i < n; i++)
14        scanf("\n %d", &a[i]);
15    printf("Elements are :");
16    for (i = 0; i < n; i++)
17        printf("\n a[%d]:%d", i, a[i]);
18    printf("\n Enter the elements to be searched :");
19    scanf("%d", &item);
20
21    loc = 0;
22    beg = 0;
23    end = n - 1;
24    mid = ((beg + end) / 2);
25    while ((beg <= end) && (item != a[mid]))
26    {
27        if (item < a[mid])
28            end = mid - 1;
29
30        else
31            beg = mid + 1;
32        mid = ((beg + end) / 2);
33    }
34    if (item == a[mid])
35    {
36        loc = mid;
37        printf("\n Data is found on %d position", loc);
38    }
39    else
40    {
41        printf("Search is unseccesfull");
42    }
43 }
```

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

```
1 // Array Implementation of Stack
2
3 #include <stdio.h>
4
5 int stack[100], choice, n, top, x, i;
6 void push(void);
7 void pop(void);
8 void traversal(void);
9 int main()
10 {
11     top = -1;
12     printf("\n Enter the size of STACK :");
13     scanf("%d", &n);
14     printf("\n\t STACK OPERATIONS USING ARRAY");
15     printf("\n\t-----");
16     printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.Traversal\n\t 4.EXIT");
17     do
18     {
19         printf("\n Enter the Choice:");
20         scanf("%d", &choice);
21         switch (choice)
22         {
23             case 1:
24             {
25                 push();
26                 break;
27             }
28             case 2:
29             {
30                 pop();
31                 break;
32             }
33             case 3:
34             {
35                 traversal();
36                 break;
37             }
38             case 4:
39             {
40                 printf("\n\t EXIT ");
41                 break;
42             }
43             default:
44             {
45                 printf("\n\t Invalid choice ");
46             }
```

```
47     }
48     } while (choice != 4);
49     return 0;
50 }
51 void push()
52 {
53     if (top >= n - 1)
54     {
55         printf("\n\tSTACK is Overflow");
56     }
57     else
58     {
59         printf(" Enter a value to be pushed:");
60         scanf("%d", &x);
61         top++;
62         stack[top] = x;
63     }
64 }
65
66 void pop()
67 {
68     if (top <= -1)
69     {
70         printf("\n\t Stack is Underflow");
71     }
72     else
73     {
74         printf("\n\t The popped elements is %d", stack[top]);
75         top--;
76     }
77 }
78 void traversal()
79 {
80     if (top >= 0)
81     {
82         printf("\n The elements in STACK \n");
83         for (i = top; i >= 0; i--)
84             printf("\n%d", stack[i]);
85         printf("\n Press Next Choice");
86     }
87     else
88     {
89         printf("\n The STACK is empty");
90     }
91 }
92
```

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

```
1 // Array implementation of circular Queue
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 void insertion();
7 int deletion();
8 void traversal();
9 int queue[5], front = 0, rear = 0, i, choice, n = 5;
10
11 void main()
12 {
13     do
14     {
15         printf("\n1.insertion \n2.deletion \n3.traversal \n4.exit");
16         printf("\n Enter your choice:");
17         scanf("%d", &choice);
18
19         switch (choice)
20         {
21             case 1:
22                 insertion();
23                 break;
24
25             case 2:
26                 deletion();
27                 break;
28
29             case 3:
30                 traversal();
31                 break;
32
33             case 4:
34                 exit(0);
35                 break;
36
37             default:
38                 printf("Invalid choice\n");
39         }
40     } while (choice != 4);
41 }
42
43 void insertion()
44 {
45     int item;
46     if ((front == 1 && rear == n) || (front == rear + 1))
47         printf("overflow");
48     else
49     {
50         printf("Enter the element to be inserted: ");
51         scanf("%d", &item);
```

```

52         if (front == 0)
53             front = rear = 1;
54         else if (rear == n)
55             rear = 1;
56         else
57             rear = rear + 1;
58         queue[rear] = item;
59     }
60 }
61
62 int deletion()
63 {
64     int item;
65     if (front == 0)
66         printf("\n underflow");
67     else
68     {
69         item = queue[front];
70         if ((front == rear) != 0)
71             front = rear = 0;
72         else if (front == n)
73             front = 1;
74         else
75             front = front + 1;
76         printf("\n The deleted item is: %d", item);
77     }
78     return (item);
79 }
80
81 void traversal()
82 {
83     int f_pos = front, r_pos = rear;
84     if (front == 0)
85     {
86         printf("Queue is empty");
87         return;
88     }
89     if (f_pos <= r_pos)
90     {
91         for (i = f_pos; i <= r_pos; i++)
92             printf("\n %d", queue[i]);
93     }
94
95     else
96     {
97         for (i = f_pos; i <= n; i++)
98             printf("\n %d", queue[i]);
99         f_pos = 1;
100         for (i = f_pos; i <= r_pos; i++)
101             printf("\n %d", queue[i]);
102     }
103 }

```

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

Program 7

```
1 // Array implementation of Deque
2
3 #include <stdio.h>
4 #include <conio.h>
5 #include <stdlib.h>
6
7 void insert_rear();
8 void insert_front();
9 int delete_front();
10 int delete_rear();
11 void traversal();
12 int queue[5], front = 0, rear = 0, i, choice, n = 5;
13
14 void main()
15 {
16     do
17     {
18         printf("\n1.Insert at rear \n2.Insert at front \n3.Delete at fro
\n4.Delete at rear \n5.Traversal \n6.Exit");
19         printf("\n Enter your choice:");
20         scanf("%d", &choice);
21
22         switch (choice)
23         {
24             case 1:
25                 insert_rear();
26                 break;
27             case 2:
28                 insert_front();
29                 break;
30             case 3:
31                 delete_front();
32                 break;
33             case 4:
34                 delete_rear();
35                 break;
36             case 5:
37                 traversal();
38                 break;
39             case 6:
40                 exit(0);
41                 break;
42             default:
43                 printf("Invalid choice \n");
44         }
45     } while (choice != 6);
46 }
47
48 void insert_rear()
```

```

49 {
50     int item;
51     if ((front == 1 && rear == n) || (front == rear + 1))
52         printf("Overflow");
53     else
54     {
55         printf("Enter the element to be inserted:");
56         scanf("%d", &item);
57         if (front == 0)
58             front = rear = 1;
59         else if (rear == n)
60             rear = 1;
61         else
62             rear = rear + 1;
63         queue[rear] = item;
64     }
65 }
66
67 void insert_front()
68 {
69     int item;
70     if ((front == 1 && rear == n) || (front == rear + 1))
71         printf("Overflow");
72     else
73     {
74         printf("Enter the element to be inserted:");
75         scanf("%d", &item);
76         if (front == 0)
77             front = rear = 1;
78         else if (front == 1)
79             front = n;
80         else
81             front = front - 1;
82         queue[front] = item;
83     }
84 }
85
86 int delete_front()
87 {
88     int item;
89     if (front == 0)
90         printf("Underflow");
91     else
92     {
93         item = queue[front];
94         if ((front == rear) != 0)
95             front = rear = 0;
96         else if (front == n)
97             front = 1;
98         else

```



```

99         front = front + 1;
100     printf("\n The deleted item is:%d", item);
101 }
102 return (item);
103 }
104
105 int delete_rear()
106 {
107     int item;
108     if (front == 0)
109         printf("Underflow");
110     else
111     {
112         item = queue[rear];
113         if ((front == rear) != 0)
114             front = rear = 0;
115         else if (rear == 1)
116             rear = n;
117         else
118             rear = rear - 1;
119         printf("\n The deleted item is:%d", item);
120     }
121     return (item);
122 }
123
124 void traversal()
125 {
126     int f_pos = front, r_pos = rear;
127     if (front == 0)
128     {
129         printf("Queue is empty");
130         return;
131     }
132
133     if (f_pos <= r_pos)
134     {
135         for (i = f_pos; i <= r_pos; i++)
136             printf("\n %d", queue[i]);
137     }
138
139     else
140     {
141         for (i = f_pos; i <= n; i++)
142             printf("\n %d", queue[i]);
143         f_pos = 1;
144         for (i = f_pos; i <= r_pos; i++)
145             printf("\n %d", queue[i]);
146     }
147 }

```

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

Program 8

```
1 // Different operations on link list
2
3 #include <stdio.h>
4 #include <conio.h>
5 #include <stdlib.h>
6
7 struct node
8 {
9     int info;
10    struct node *link;
11 };
12 struct node *start = NULL;
13
14 void insertbeg();
15 void insertafter(int, int);
16 void deleteafter();
17 void deletelist();
18 int traversal();
19 int choice, pos, data;
20
21 void main()
22 {
23     do
24     {
25         printf("\n1.Insert at beg \n2.Insert after pos \n3.Delete after pos
\n4.Delete-list \n5.Traversal \n6.Exit");
26         printf("\nEnter your choice :");
27         scanf("%d", &choice);
28
29         switch (choice)
30         {
31             case 1:
32                 insertbeg();
33                 break;
34
35             case 2:
36                 insertafter(pos, data);
37                 break;
38
39             case 3:
40                 deleteafter();
41                 break;
42
43             case 4:
44                 deletelist();
45                 break;
46
47             case 5:
48                 traversal();
49                 break;
50
```

```

51         case 6:
52             exit(0);
53             break;
54
55         default:
56             printf("invalid choice \n");
57     }
58     } while (choice != 6);
59 }
60
61 void insertbeg()
62 {
63     int data;
64     struct node *new;
65     printf("Enter the info of node :");
66     scanf("%d", &data);
67     new = (struct node *)malloc(sizeof(struct node));
68     new->info = data;
69     new->link = start;
70     start = new;
71 }
72
73 void insertafter(int pos, int data)
74 {
75     int i;
76     struct node *new;
77     struct node *ptr = start;
78
79     printf("Enter the position after which element is inserted :");
80     scanf("%d", &pos);
81     for (i = 0; i < pos; i++)
82     {
83         ptr = ptr->link;
84         if (ptr == NULL)
85         {
86             printf("There are less than %d elements", pos);
87             return;
88         }
89     }
90     printf("Enter the info of node:");
91     scanf("%d", &data);
92     new = (struct node *)malloc(sizeof(struct node));
93     new->info = data;
94     new->link = ptr->link;
95     ptr->link = new;
96 }
97
98 void deleteafter()
99 {
100     int i;
101     struct node *save = start;

```

```

102     struct node *ptr = start->link;
103     struct node *locp = NULL;
104     struct node *loc = NULL;
105
106     printf("Enter the positio after which element is deleted :");
107     scanf("%d", &pos);
108     for (i = 0; i < pos; i++)
109     {
110         save = ptr;
111         ptr = ptr->link;
112     }
113     locp = save;
114     loc = ptr;
115     if (pos == 0)
116     {
117         start = start->link;
118         return;
119     }
120     else
121         locp->link = loc->link;
122     free(loc);
123 }
124
125 void deletelist()
126 {
127     struct node *ptr;
128     while (start != NULL)
129     {
130         ptr = start;
131         start = start->link;
132         free(ptr);
133     }
134 }
135
136 int traversal()
137 {
138     int count = 0;
139     struct node *ptr = start;
140     printf("\n The linked list is :\n\n");
141     printf("->");
142     while (ptr)
143     {
144         printf("%d->", ptr->info);
145         ptr = ptr->link;
146         count++;
147     }
148     printf("NULL\n");
149     printf("\nNumber of elements in linked list: %d,count");
150     return (count);
151 }

```

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

Program 9

```
1 // Bubble sort
2
3 #include <stdio.h>
4
5 void bubble_sort(int a[], int);
6 void main()
7 {
8     int i, n, a[20];
9
10    printf("Enter the number of elements: ");
11    scanf("%d", &n);
12    printf("Enter the elements : ");
13    for (i = 0; i < n; i++)
14        scanf("%d", &a[i]);
15    printf("Elements are :");
16
17    for (i = 0; i < n; i++)
18        printf("\n a[%d]:%d", i, a[i]);
19    bubble_sort(a, n);
20    printf("\n Sorted elements are :");
21    for (i = 0; i < n; i++)
22        printf("\n a[%d]:%d", i, a[i]);
23 }
24
25 void bubble_sort(int a[], int n)
26 {
27     int pass, j, temp;
28     for (pass = 1; pass < n; pass++)
29     {
30         for (j = 0; j <= n - pass - 1; j++)
31         {
32             if (a[j] > a[j + 1])
33             {
34                 temp = a[j];
35                 a[j] = a[j + 1];
36                 a[j + 1] = temp;
37             }
38         }
39     }
40 }
41
```

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

```
1 // Selection sort
2
3 #include <stdio.h>
4
5 void Selection_sort(int a[], int);
6 void main()
7 {
8     int i, n, a[20];
9
10    printf("Enter the number of elements: ");
11    scanf("%d", &n);
12    printf("Enter the elements : ");
13    for (i = 0; i < n; i++)
14        scanf("%d", &a[i]);
15    printf("Elements are :");
16
17    for (i = 0; i < n; i++)
18        printf("\n a[%d]:%d", i, a[i]);
19    selection_sort(a, n);
20    printf("\n Sorted elements are :");
21    for (i = 0; i < n; i++)
22        printf("\n a[%d]:%d", i, a[i]);
23 }
24
25 void selection_sort(int a[], int n)
26 {
27     int min, loc, temp, k, j;
28     min = a[0];
29     for (k = 0; k < n - 1; k++)
30     {
31         min = a[k];
32         loc = k;
33         for (j = k + 1; j <= n - 1; j++)
34         {
35             if (min > a[j])
36             {
37                 min = a[j];
38                 loc = j;
39             }
40         }
41         if (loc != k)
42         {
43             temp = a[k];
44             a[k] = a[loc];
45             a[loc] = temp;
46         }
47     }
48 }
```

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

```
1 // Insertion sort
2
3 #include <stdio.h>
4
5 void insertion_sort(int a[], int);
6 void main()
7 {
8     int i, n, a[20];
9
10    printf("Enter the number of elements: ");
11    scanf("%d", &n);
12    printf("Enter the elements : ");
13    for (i = 0; i < n; i++)
14        scanf("%d", &a[i]);
15    printf("Elements are :");
16
17    for (i = 0; i < n; i++)
18        printf("\n a[%d]:%d", i, a[i]);
19    insertion_sort(a, n);
20    printf("\n Sorted elements are :");
21    for (i = 0; i < n; i++)
22        printf("\n a[%d]:%d", i, a[i]);
23 }
24
25 void insertion_sort(int a[], int n)
26 {
27     int temp, i, j, k;
28     for (k = 1; k <= n-1; k++)
29     {
30         temp = a[k];
31         j = k - 1;
32         while ((temp < a[j]) && (j >= 0))
33         {
34             a[j + 1] = a[j];
35             j = j - 1;
36         }
37         a[j + 1] = temp;
38     }
39 }
```

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
```