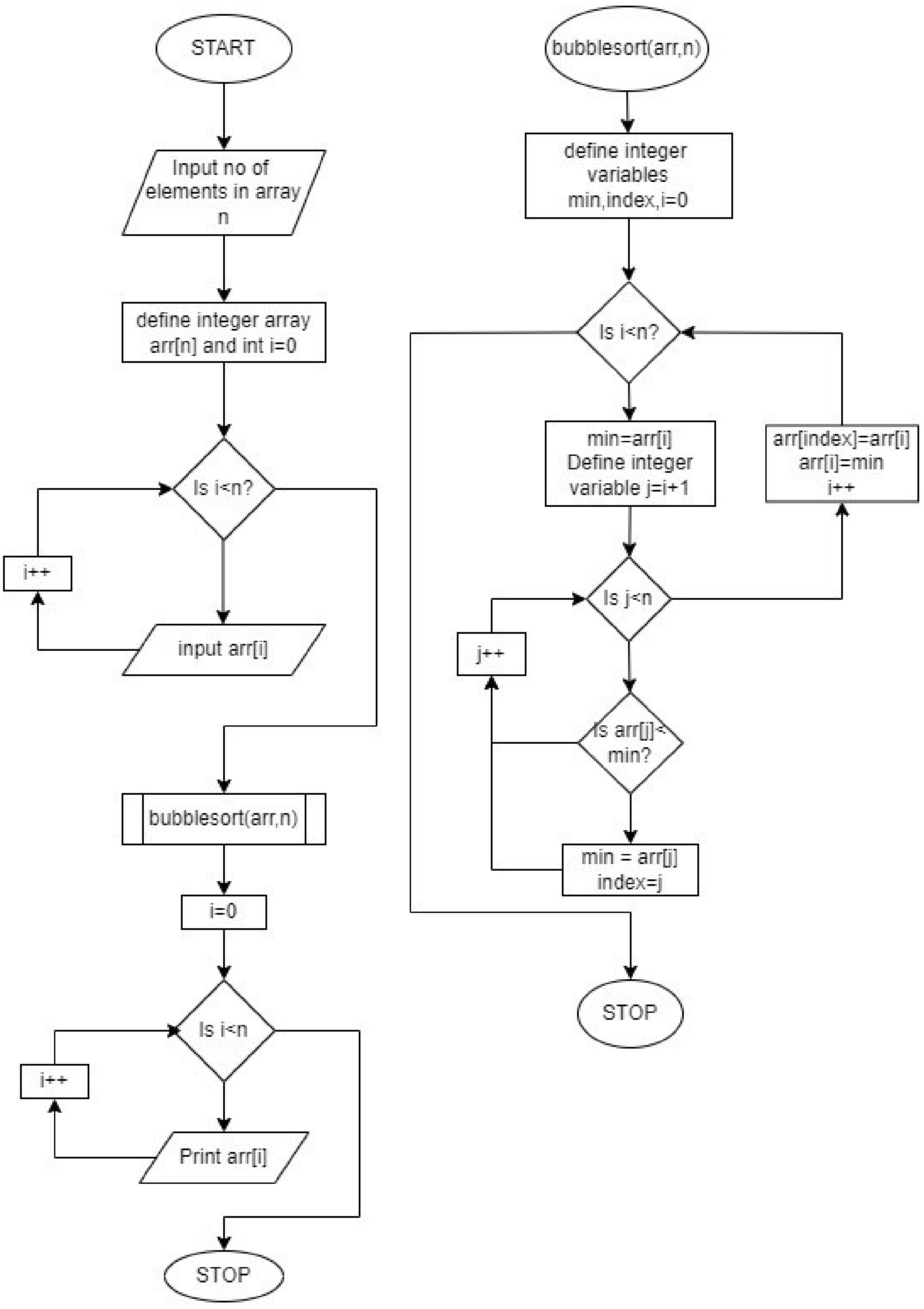
Name	Vineet Parmar
UID no.	2021300092
Experiment No.	5

AIM:	Demonstrate the use of one-dimensional arrays to solve a given problem.
	Program 1

PROBLEM STATEMENT :	The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning. The algorithm maintains two sub-arrays in a given array.
	1) The sub-array which is already sorted.
	2) Remaining sub-array which is unsorted.
	Every iteration of selection sort, the minimum element (considering ascending order) from the unsorted sub-array is picked and moved to the sorted sub-array.
	Following example explains the above steps:
	arr[] = 64 25 12 22 11
	// Find the minimum element in arr[0...4]
	// and place it at beginning
	11 25 12 22 64
	// Find the minimum element in arr[1...4]
	// and place it at beginning of arr[1...4]
	11 12 25 22 64
	// Find the minimum element in arr[2...4]
	// and place it at beginning of arr[2...4]
	11 12 22 25 64
	// Find the minimum element in arr[3...4]
	// and place it at beginning of arr[3...4]
	11 12 22 25 64

ALGORITHM:	1. START 2. Define void function bubblesort with an integer parameter and its size. 3. Define integer variables min and index. 4. I=0

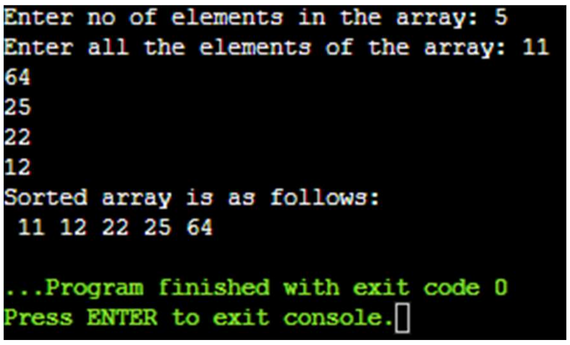
```
5. min=arr[i]
6. j=i+1
7. If(min>arr[j])
     index=j
     min = arr[j]
8. j++
9.  Repeat 7 and 8 till j<n
10. arr[index]=arr[i]
    arr[i]=min
11. Repeat 5,6,7,8,9 and 10 till i<n
12. Define integer function main()
13. Input number of elements in array n
14. Input all elements of array arr[]
15. Call function bubblesort(arr,n)
16. Print the sorted array
17. STOP
```

FLOWCHART:	 START Input no of elements in array n define integer array arr[n] and int i=0 Is i<n? i++ input arr[i] bubblesort(arr,n) i=0 Is i<n i++ Print arr[i] STOP bubblesort(arr,n) define integer variables min,index,i=0 Is i<n? min=arr[i] Define integer variable j=i+1 arr[index]=arr[i] arr[i]=min i++ Is j<n j++ Is arr[j]< min? min = arr[j] index=j STOP

PROGRAM:

```
#include<stdio.h>
void bubblesort(int arr[],int n)
{
    int min,index;
    for(int i=0;i<n;i++)
    {
        index = i;
        for(int j=i+1;j<n;j++)
        {
            if(min>arr[j])
            {
                min=arr[j];
                index = j;
            }
        }
        int temp = arr[index];
        arr[index]=arr[i];
        arr[i]=min;
    }
}
int main()
{
    int n;
    printf("Enter no of elements in the array: ");
    scanf("%d",&n);
    int arr[n];
    printf("Enter all the elements of the array: ");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    bubblesort(arr,n);
    printf("Sorted array is as follows:\n ");
    for(int i=0;i<n;i++)
    {
        printf("%d ",arr[i]);
    }
    return 0;
}
```

RESULT:



```
Enter no of elements in the array: 5
Enter all the elements of the array: 11
64
25
22
12
Sorted array is as follows:
 11 12 22 25 64

...Program finished with exit code 0
Press ENTER to exit console.
```

Program 2

PROBLEM STATEMENT :

Perform search of a particular element on the above array using binary search. Binary Search will search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty. We basically ignore half of the elements just after one comparison.
1. Compare x with the middle element.
2. If x matches with middle element, we return the mid index.
3. Else If x is greater than the mid element, then x can only lie in right half subarray after the
mid element. So we recur for right half.
4. Else (x is smaller) recur for the left half

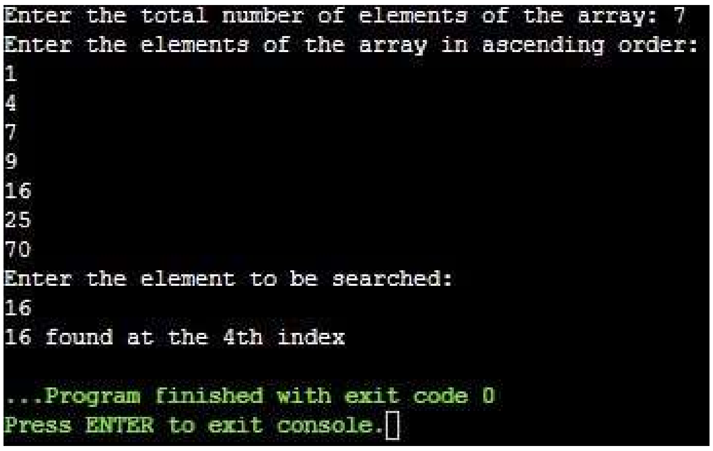
ALGORITHM:

```
1. START
2. Define integer function binarysearch() with an integer array and 3
   integer parameters start, end and element
3. if end is not equal to 0
   {
   mid = (start + end-1)/2
   if(arr[mid] = element)
       return mid
   else if (arr[mid]>element)
       return binarysearch(arr,start,mid-1,element)
   else
       return binarysearch(arr,mid+1,end,element)
   }
   else
       return -1
4. Define integer function main()
```

	5. Input no. of elements of array n 6. Input integer array arr[n] 7. Input element to be searched x 8. Call function binarysearch(arr,0,n-1,x) 9. If binarysearch(arr,0,n-1,x) is not equal to -1 Print binarysearch(arr,0,n-1,x) Else Print “Not Found” 10. STOP.
PROGRAM:	see below

```
#include<stdio.h>
int binarysearch(int arr[],int start,int end,int n)
{
    if(end!=0)
    {
        int mid = start+(end-1)/2;
        if(arr[mid]==n)
        {
            return mid;
        }
        else if(arr[mid]>n)
        {
            return binarysearch(arr,start,mid-1,n);
        }
        else
        {
            return binarysearch(arr,mid+1,end,n);
        }
    }
    else
        return -1;
}
int main()
{
    int x,n;
    printf("Enter the total number of elements of the array: ");
    scanf("%d",&n);
    int arr[n];
    printf("Enter the elements of the array in ascending order:\n");
    for(int i=0;i<n;i++)
    {
       scanf("%d",&arr[i]);
    }
    printf("Enter the element to be searched: \n");
    scanf("%d",&x);
    if(binarysearch(arr,0,n-1,x)!=-1)
    {
       printf("%d found at the %dth index",x,binarysearch(arr,0,n-1,x));
    }
    else
    {
       printf("Element not found");
    }
    return 0;
}
```

RESULT:



```
Enter the total number of elements of the array: 7
Enter the elements of the array in ascending order:
1
4
7
9
16
25
70
Enter the element to be searched:
16
16 found at the 4th index

...Program finished with exit code 0
Press ENTER to exit console.
```

Program 3

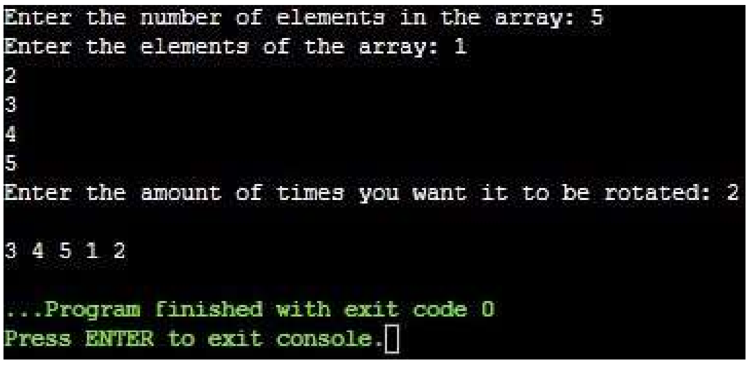
PROBLEM STATEMENT:	Circular Array Rotation means rotating the elements in the array where one rotation operation moves the first element of the array to the last position and shifts all remaining elements to the left. Here, we are given an unsorted array and our task is to perform the circular rotation by n number of rotations where n is a natural number. Initial Array: [1 2 3 4 5] After one rotation : [2 3 4 5 1] After two rotation : [3 4 5 1 2]
ALGORITHM:	1. START 2. Define void function rotation() with integer array arr[] and two integer

```
      parameter m and n
 3. Define integer variables temp, i and j.
 4. i=1
 5. temp = arr[0]
 6. j=0
 7. arr[j]=arr[j+1]
 8. j++
 9. Repeat 7 and 8 till j<n-1
10. Arr[n-1]=temp
11. i++
12. Repeat 5,6,7,8,9,10 and 11 till i<=m
13. Define integer function main()
14. Input no of elements in array n
15. Input no of times array is to be rotated m
16. Input integer array arr[n]
17. Call function rotation(arr,n,m)
18. Print the array
19. STOP
```

PROGRAM:

```c
#include<stdio.h>
void rotation(int arr[],int n,int m)
{
    int temp,i;
    for(i=1;i<=m;i++)
    {
        temp = arr[0];
        for (int j = 0; j < n - 1; j++)
        {
            arr[j] = arr[j + 1];
        }
        arr[n-1]=temp;
    }
}
int main()
{
    int m,n;
    printf("Enter the number of elements in the array: ");
    scanf("%d",&n);
    int arr[n];
    printf("Enter the elements of the array: ");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    printf("Enter the amount of times you want it to be rotated: ");
    scanf("%d",&m);
    rotation(arr,n,m);
    printf("\n");
    for(int i=0;i<n;i++)
    {
        printf("%d ",arr[i]);
    }
    return 0;
}
```

RESULT:



```
Enter the number of elements in the array: 5
Enter the elements of the array: 1
2
3
4
5
Enter the amount of times you want it to be rotated: 2

3 4 5 1 2

...Program finished with exit code 0
Press ENTER to exit console.
```

CONCLUSION: In this experiment we learnt the use of 1-D arrays. We learnt how to declare an integer array and how to call it in a function as a parameter. We also learned how to sort an array using the bubble sort algorithm and how to find an element in a sorted using the binary search algorithm.