Q1.WAP to read an array of integers and search for an element using linear search.

```
#include <stdio.h>
int main()
{
    int array[10], search, c, n;

    printf("Enter 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 a number to search\n");
    scanf("%d", &search);
    for (c = 0; c < n; c++)
    {
        if (array[c] == search)
        {
            printf("%d is present at location %d.\n", search, c + 1);
            break;
        }
    }
    if (c == n)
        printf("%d isn't present in the array.\n", search);
    return 0;
}
```

==========

OUTPUT

```
Enter number of elements in array
5
Enter 5 integer(s)
1
3
5
4
2
Enter a number to search
4
4 is present at location 4.
```

==========

Q2. WAP to read an array of integers and search for an element using binary search

```
#include <stdio.h>
int main()
```

```
{
    int c, first, last, middle, n, search, array[100];

    printf("Enter 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 a number to search\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 isn't present in the list.\n", search);

    return 0;
}
```

==========

OUTPUT

```
Enter number of elements in array
6
Enter 6 integer(s)
1
2
3
4
5
6
Enter a number to search
```

```
        3
3 found at location 3.
```

==========

Q3. Given an array container and integer hunt. WAP to find whether hunt is present in container or not. If present, then triple the value of hunt and search again. Repeat these steps until hunt is not found. Finally return the value of hunt.
Input: container = {1, 2, 3} and hunt = 1 then Output: 9
Explanation: Start with hunt = 1. Since it is present in array, it becomes 3. Now 3 is present in array and hence hunt becomes 9. Since 9 is a not present, program return 9.

```
#include <stdio.h>
int main()
{
    int Container[10], c;
    int hunt, i, j, l, n, Flag = 1;

    printf("Enter number of elements in array\n");
    scanf("%d", &n);
    printf("Enter %d integer(s)\n", n);
    for (c = 0; c < n; c++)
        scanf("%d", &Container[c]);
    printf("Enter a number to search\n");
    scanf("%d", &hunt);
    l = sizeof(Container) / sizeof(Container[0]);

    printf("\nInitial Hunt Value = %d", hunt);
    while (Flag == 1)
    {
        Flag = 0;
        for (i = 0; i < l; i++)
        {
            if (hunt == Container[i])
            {
                Flag = 1;
                hunt = hunt * 3;
            }
        }
    }
    printf("\nFinal Hunt Value = %d", hunt);
    return 0;
}
```

==========

OUTPUT

```
Enter number of elements in array
10
```

```
Enter 10 integer(s)
1
2
3
4
5
6
7
8
9
10
Enter a number to search
1

Initial Hunt Value = 1
Final Hunt Value = 27
```

Q4. Given a sorted array of length n, WAP to find the number in array that appears more than or equal to n/2 times. It can be assumed that such element always exists.
Input: 2 3 3 4 Output: 3
Input: 3 4 5 5 5 Output: 5

```
#include <stdio.h>
int main()
{
    int n;
    printf("Input: ");
    scanf("%d", &n);
    int array[n];
    int max = 0;
    int count;
    int maxelement;
    printf("Output: ");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &array[i]);
    }
    for (int i = 0; i < n; i++)
    {
        count++;
        for (int j = 0; j < n; j++)
        {
            if (array[i] == array[j])
            {
                count++;
            }
```

```
        }
        if (count > max)
        {
            max = count;
            maxelement = array[i];
        }
    }
    printf("%d", maxelement);
    return 0;
}
```

==========

OUTPUT

Input: 2 3 3 3 4 5
Output: 3

==========

Q5. WARP (Write a Recursive Program) to search an element in a dynamic array of n integers using linear search.

```
#include <stdio.h>
int search(int A[], int info, int i, int n)
{
    int pos = 0;
    if (i >= n)
    {
        return 0;
    }
    else if (A[i] == info)
    {
        pos = i + 1;
        return pos;
    }
    else
    {
        return search(A, info, i + 1, n);
    }
    return pos;
}
int main()
{
    int n, info, pos, m = 0, A[10];
    printf("Enter the total elements in the array  ");
    scanf("%d", &n);
    printf("Enter the array elements\n");
    for (int i = 0; i < n; i++)
    {
```

```
        scanf("%d", &A[i]);
    }
    printf("Enter the element to search  ");
    scanf("%d", &info);
    pos = search(A, info, 0, n);
    if (pos != 0)
    {
        printf("Element found at pos %d ", pos);
    }
    else
    {
        printf("Element not found");
    }
    return 0;
}
```

==========

OUTPUT

```
Enter the total elements in the array  6
Enter the array elements
1
2
3
4
5
6
Enter the element to search  4
Element found at pos 4
```

==========

Q6. WARP using recursion to search an element in a dynamic array of n integers using binary search

```
#include <stdio.h>
#include <stdlib.h>
int BinarySearch(int *p, int item, int n);
void input(int *, int);
void main()
{
    int item, n, pos;
    printf("Enter no of elements: ");
    scanf("%d", &n);
    int *p = (int *)calloc(n, sizeof(int));
    printf("Enter %d numbers:", n);
    input(p, n);
    printf("Enter a no to search in an array:");
    scanf("%d", &item);
```

```
    pos = BinarySearch(p, item, n);
    if (pos < 0)
        printf("Number NOT present");
    else
        printf("Item present and its position=%d", pos + 1);
}
void input(int *p, int n)
{
    static int i;
    if (i < n)
    {
        scanf("%d", &(*(p + i)));
        i++;
        input(p, n);
    }
}
int BinarySearch(int *p, int item, int n)
{
    static int first = 0;
    int last = n - 1;
    static int mid;
    if (first <= last)
    {
        mid = (first + last) / 2;
        if (p[mid] == item)
            return mid;
        else if (item > p[mid])
            first = mid + 1;
        else
            last = mid - 1;
        first++;
        BinarySearch(p, item, n);
    }
    return -1;
}
```

==========

OUTPUT

```
Enter no of elements: 7
Enter 7 numbers:7
6
5
4
3
2
```

```
1
Enter a no to search in an array:4
Item present and its position=4
================================================================================
```