Design and Analysis of Algorithms (Week1)
PCS-505

Note - You can find all the problem statements and their solutions on following link - DAA Link

Sample Input Output Format For Each Problem is as follows -

Input format:
The first line contains a number of test cases, T.
For each test case, there will be three input lines.
First line contains n (the size of array).
Second line contains n space-separated integers describing the array.
Third line contains the key element that needs to be searched in the array.

Output format:
The output will have a T number of lines.
For each test case, output will be “**Present**” if the key element is found in the array, otherwise “**Not Present**”.
Also for each test case output the number of **comparisons** required to search the key.

1. Given an array of nonnegative integers, design a linear algorithm and implement it using a program to find whether a given key element is present in the array or not. Also, find the total number of comparisons for each input case.
   (Time Complexity = **O(n)**, where n is the size of input)

   Sample I/O:

Input	**Output**
3	Present 6
8	Present 3
34 35 65 31 25 89 64 30	Not Present 6
89	
5	
977 354 244 546 355	
244	
6	
23 64 13 67 43 56	
63	

2. Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether a given key element is present in the array or not. Also, find the total number of comparisons for each input case.
   (Time Complexity = **O(logn)**, where n is the size of input).

Input	Output
3	Present 3
5	Not Present 4
12 23 36 39 41	Present 3
41	
8	
21 39 40 45 51 54 68 72	
69	
10	
101 246 438 561 796 896 899 4644 7999 8545	
7999	

3. **Jump Search** - Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether a given key element is present in the sorted array or not. For an array arr[] of size **n** and block (to be jumped) size **m**, search at the indexes arr[0], arr[m], arr[2m].....arr[km] and so on. Once the interval **(arr[km] < key < arr[(k+1)m])** is found, perform a linear search operation from the index **km** to find the element **key**.
   (Time Complexity = **O(√n)**, where n is the number of elements need to be scanned for searching):

Input	Output
3	Present 3
5	Not Present 4
12 23 36 39 41	Present 5
41	
8	
21 39 40 45 51 54 68 72	
69	
10	
101 246 438 561 796 896 899 4644 7999 8545	
7999	

4. **Exponential Search** - Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether a given key element is present in the sorted array or not. For an array arr[] of size **n**, search at the indexes **arr[0], arr[1], arr[2], arr[4],.....,arr[2^k]** and so on. Once the interval **(arr[2^k] < key < arr[2^(k+1)])** is found, perform a linear search or binary search operation from the index **2^k** to find the element key. (Complexity = **O(logn)**, where n is the number of elements need to be scanned for searching):

Input	**Output for Linear Search Version**
3	Present 5
5	Not Present 8
12 23 36 39 41	Present 6
41	
8	
21 39 40 45 51 54 68 72	
69	
10	
101 246 438 561 796 896 899 4644 7999 8545	
7999	

Input	**Output for Binary Search Version**
3	Present 5
5	Not Present 7
12 23 36 39 41	Present 6
41	
8	
21 39 40 45 51 54 68 72	
69	
10	
101 246 438 561 796 896 899 4644 7999 8545	
7999	