PRACTICAL - 1

AIM : To perform linear search

PSEUDO CODE :

Linear Search (Array A, Value x)

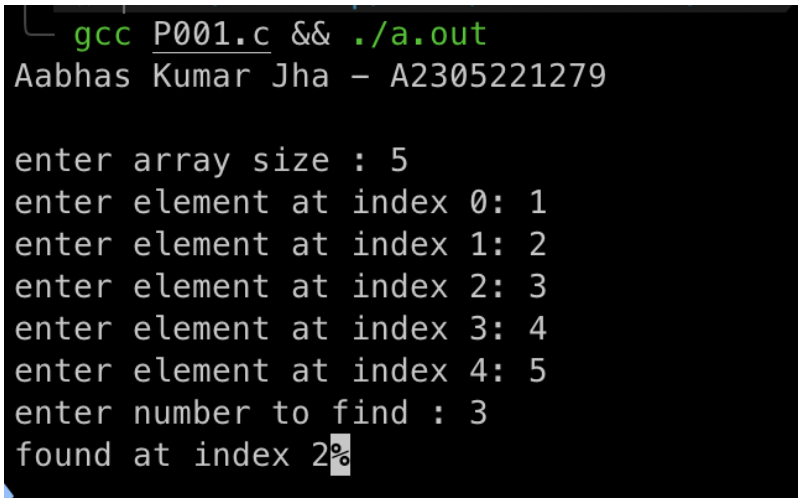
Step 1: Set i to 1
Step 2: if i > n then go to step 7
Step 3: if A[i] = x then go to step 6
Step 4: Set i to i + 1
Step 5: Go to Step 2
Step 6: Print Element x Found at index i and go to step 8
Step 7: Print element not found
Step 8: Exit

CODE :

```c
// linear search

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

int main(int argc, char const *argv[])
{

  printf("Aabhas Kumar Jha - A2305221279\n\n");
  bool found = false;

  int size;
  printf("enter array size : ");
  scanf("%d", &size);
  int arr[size];

  int element;
  for (int i= 0; i < size; i++){
     printf("enter element at index %d: ", i);
     scanf("%d", &element);
     arr[i] = element;
  }

  int num;
```

```
    printf("enter number to find : ");
    scanf("%d", &num);

    for (int i= 0; i < size; i++){
        if (num == arr[i]){
            printf("found at index %d", i);
            found = true;
        }
    }

    if (!found){
        printf("not found");
    }

    return 0;
}
```

OUTPUT :



COMPLEXITY : O(n)

PRACTICAL - 2

AIM : To perform binary search
a) without using recursion
b) using recursion

PSEUDO CODE :

a) without using recursion

Binary Search (Array A, Value x)

Step 1: Set low to 1
Step 2: Set high to n
Step 3: while low <= high, do steps 4-7
Step 4: Set mid to (low + high) / 2
Step 5: if A[mid] = x, then go to step 8
Step 6: if A[mid] < x, then set low to mid + 1
Step 7: if A[mid] > x, then set high to mid - 1
Step 8: Print Element x Found at index mid and go to step 9
Step 9: Exit

b) using recursion

Binary Search (Array A, Value x, low, high)

Step 1: if low > high, then go to step 6
Step 2: Set mid to (low + high) / 2
Step 3: if A[mid] = x, then go to step 4
Step 4: Print Element x Found at index mid and go to
step 5 Step 5: Exit
Step 6: Print Element not found
Step 7: Exit

CODE :

a) binary search without using recursion

```
// binary search without recursion

#include <stdio.h>

int binary_search(int arr[], int n, int target) {
  int left = 0;
  int right = n - 1;
  while (left <= right) {
    int mid = left + (right - left) / 2;

    if (arr[mid] == target) {
      return mid;
    } else if (arr[mid] < target) {
      left = mid + 1;
    } else {
      right = mid - 1;
    }
  }

  return -1;
}

int main() {
  printf("Aabhas Kumar Jha - A2305221279\n\n");
  int n;
  printf("Enter the size of the array: ");
  scanf("%d", &n);
  int arr[n];
  printf("Enter the elements of the sorted array:\n");
  for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }
  int target;
  printf("Enter the element to be found: ");
  scanf("%d", &target);
  int result = binary_search(arr, n, target);
  if (result != -1) {
    printf("Element found at index %d\n", result);
  } else {
    printf("Element not found in the array.\n");
  }
  return 0;
```

}

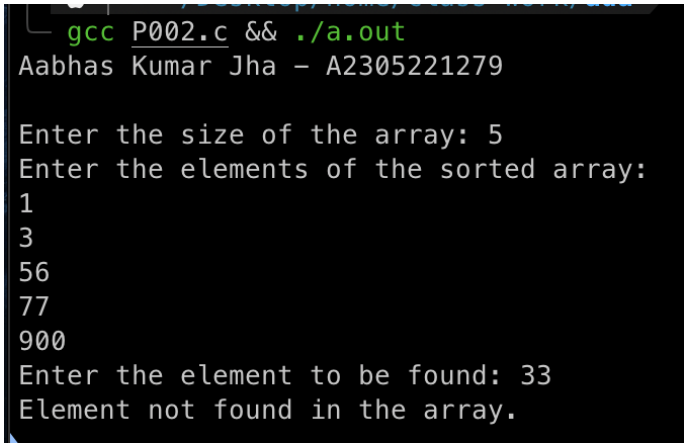
b) binary search using recursion

```
// binary search using recursion

#include <stdio.h>

int binary_search_recursive(int arr[], int left, int right, int target) {
  if (left <= right) {
     int mid = left + (right - left) / 2;

     if (arr[mid] == target) {
        return mid;
     } else if (arr[mid] < target) {
        return binary_search_recursive(arr, mid + 1, right, target);
     } else {
        return binary_search_recursive(arr, left, mid - 1, target);
     }
  }

  return -1;
}

int main() {
  printf("Aabhas Kumar Jha - A2305221279\n\n");
  int n;
  printf("Enter the size of the array: ");
  scanf("%d", &n);

  int arr[n];
  printf("Enter the elements of the sorted array:\n");
  for (int i = 0; i < n; i++) {
     scanf("%d", &arr[i]);
  }

  int target;
  printf("Enter the element to be found: ");
  scanf("%d", &target);

  int result = binary_search_recursive(arr, 0, n - 1, target);

  if (result != -1) {
     printf("Element found at index %d\n", result);
  } else {
     printf("Element not found in the array.\n");
  }
```

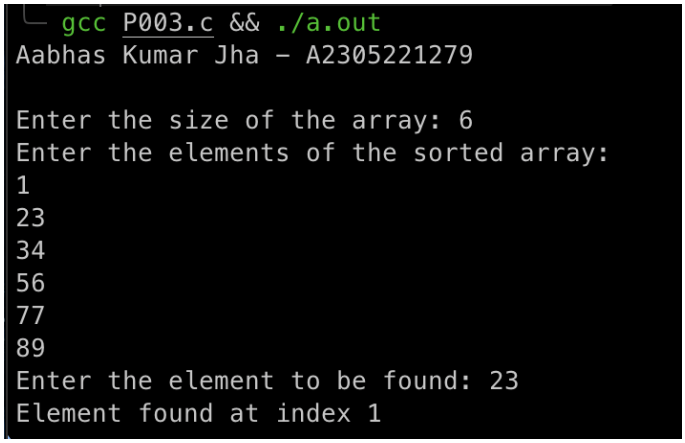
```
  return 0;
}
```

OUTPUT :

a) **Without recursion**



b) **Using recursion**



COMPLEXITY : O(log n)

}

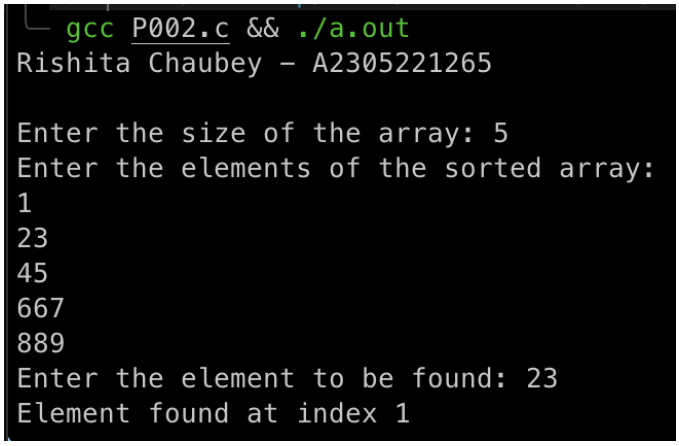
b) binary search using recursion

```
// binary search using recursion

#include <stdio.h>

int binary_search_recursive(int arr[], int left, int right, int target) {
  if (left <= right) {
    int mid = left + (right - left) / 2;

    if (arr[mid] == target) {
      return mid;
    } else if (arr[mid] < target) {
      return binary_search_recursive(arr, mid + 1, right, target);
    } else {
      return binary_search_recursive(arr, left, mid - 1, target);
    }
  }

  return -1;
}

int main() {
  printf("Rishita Chaubey - A2305221265\n\n");
  int n;
  printf("Enter the size of the array: ");
  scanf("%d", &n);

  int arr[n];
  printf("Enter the elements of the sorted array:\n");
  for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }

  int target;
  printf("Enter the element to be found: ");
  scanf("%d", &target);

  int result = binary_search_recursive(arr, 0, n - 1, target);

  if (result != -1) {
    printf("Element found at index %d\n", result);
  } else {
    printf("Element not found in the array.\n");
  }
```

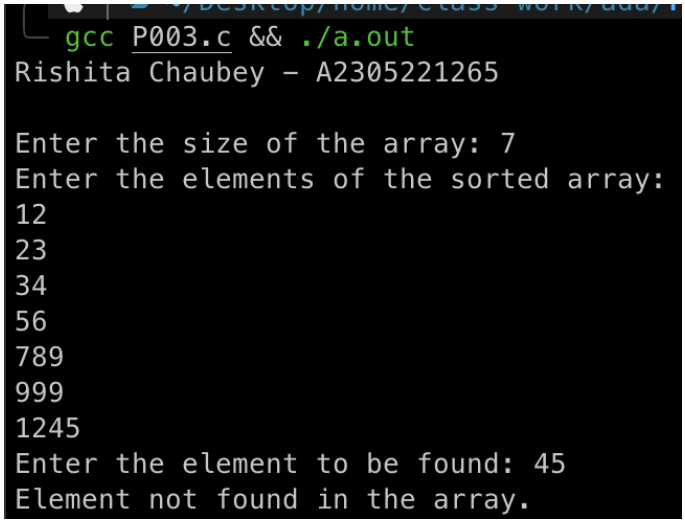
```
    return 0;
}
```

OUTPUT :

a) **Without recursion**



b) **Using recursion**



COMPLEXITY : O(log n)