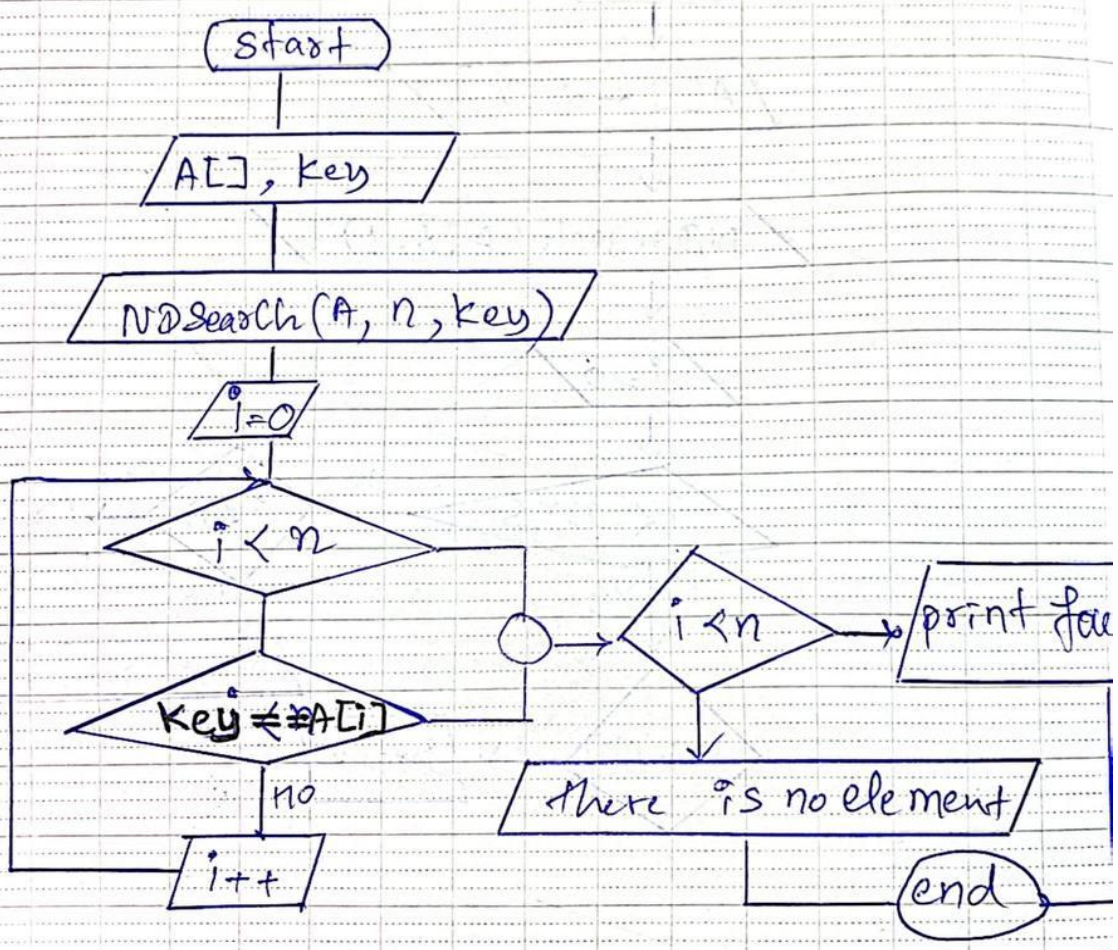
Ex1.cpp | Ex2.cpp | Ex3.cpp

Ex1.cpp > main()

```cpp
#include <iostream>
using namespace std;
int NDsearch(int A[], int n, int key)
{
    int i;
    for (i = 0; i < n; i++)
    {
        if (key == A[i])
        {
            break;
        }
    }
    if (i < n)
    {
        cout << "found";
    }
    else
    {
        cout << "there is no element";
    }
}
int main()
{
    int A[] = {12, 34, 56, 74, 11, 13, 10, 20};
    int key = 11;
    NDsearch(A, 8, key);
}
```

Exercise 1. Non-Duplicate

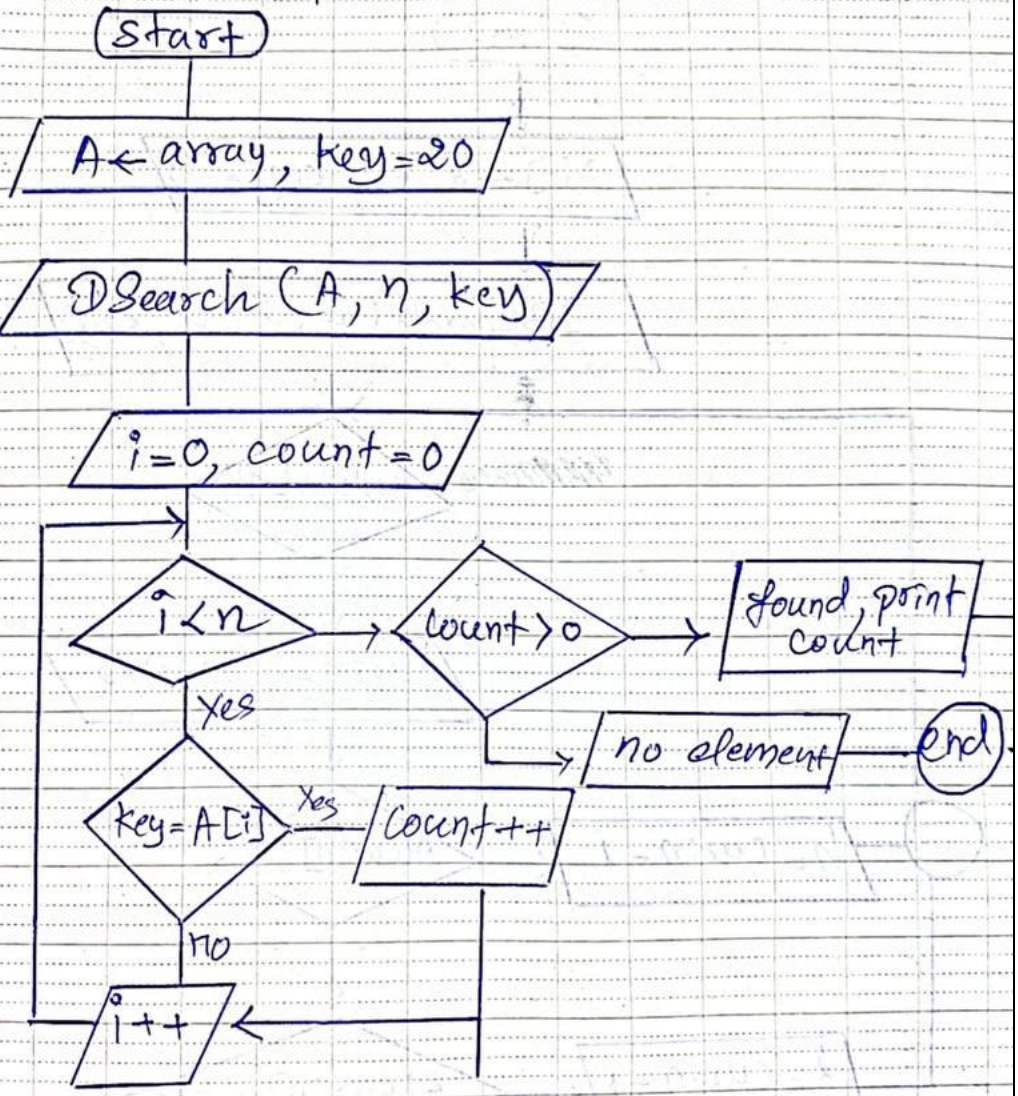


Ex1.cpp | Ex2.cpp ● | Ex3.cpp

Ex2.cpp > Dsearch(int [], int, int)

```
4   {
5       int i;
6       int count = 0;
7       for (i = 0; i < n; i++)
8       {
9           if (key == A[i])
10          {
11              count++;
12          }
13      }
14      if (count > 0)
15      {
16          cout << "found there are " << count << " element in array";
17      }
18      else
19      {
20          cout << "there is no element";
21      }
22  }
23  int main()
24  {
25      int A[] = {12, 34, 56, 74, 11, 13, 10, 20, 20, 20, 20};
26      int key = 20;
27      Dsearch(A, 11, key);
28  }
```

Exercise 2. Duplicate



Ex1.cpp | Ex2.cpp | Ex3.cpp

Ex3.cpp > IBsearch(int [], int, int, int)

```cpp
#include <iostream>
using namespace std;
int RBsearch(int A[], int l, int h, int key)
{
    int curin = (l + h) / 2;
    if (A[curin] < key)
    {
        return RBsearch(A, curin + 1, h, key);
    }
    else if (A[curin] > key)
    {
        return RBsearch(A, l, curin - 1, key);
    }
    return curin;
}
int IBsearch(int A[], int l, int h, int key)
{
    int curin;
    while (l < h)
    {
        curin = (l + h) / 2;
        if (A[curin] > key)
        {
            h = curin - 1;
        }
        else if (A[curin] < key)
        {
            l = curin + 1;
        }
        else
        {
            return curin;
        }
    }
}

int main()
{
    int A[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    int key = 9; // key search
    cout << "the element found  at index " << RBsearch(A, 0, 8, key) << " of array" << endl;
    cout << "the element found  at index " << IBsearch(A, 0, 8, key) << " of array";
}
```

Exercise 3: Binary Search.