Ex. No: 6 **Date: 17.09.24**

Register No.: 230701360 **Name: Tejushree sanjeevikumar**

Competitive Programming

6.a. Finding Duplicates-O(n^2) Time Complexity (1) Space Complexity

Aim: Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

Input Format:

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

Algorithm:

```
void function(int n){
    set c = 0
    read n from input
    create array a of size n

    for i = 0 to n - 1:
        read a[i] from input

    for i = 0 to n - 1:
        for j = 0 to n - 1:
```

```
            if i != j and a[i] == a[j]:
                increment c by a[i]
                break

        if c > 0:
            break

    print c
}
```

Program:

```
#include<stdio.h>
int main()
{
    int n,c=0;
    scanf("%d",&n);
    int a[n];
    for (int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if (i!=j&&a[i]==a[j])
            {
                c=c+a[i];
                break;
```

```
        }
    }
    if(c>0)
    {
        break;
    }
  }
  printf("%d",c);
}
```

Output:

	Input	Expected	Got	
✔	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✔
✔	5 1 2 3 4 4	4	4	✔
✔	5 1 1 2 3 4	1	1	✔

6.b. Finding Duplicates-O(n) Time Complexity (1) Space Complexity

Aim: Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

Input Format:

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

Algorithm:

```
void function(int n){
    read n from input
    create array a of size n
    set ind = 0
    set b = 0

    for i = 0 to n - 1:
        read b from input
        set ind = b % n

        if a[ind] != 0 and a[ind] == b:
            print b
            break

        set a[ind] = b
}
```

Program:

```
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    int ind,b;
    for (int i=0;i<n;i++)
    {
        scanf("%d",&b);
        ind=b%n;

        if(a[ind]!=0 && a[ind]==b) {
            printf("%d",b);
            break;
        }
        a[ind]=b;
    }

}
```

Output:

	Input	Expected	Got	
✔	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✔
✔	5 1 2 3 4 4	4	4	✔
✔	5 1 1 2 3 4	1	1	✔

6.c. Print Intersection of 2 sorted arrays-O(m*n)Time Complexity,O(1) Space Complexity

Aim:

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

· The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

1 6

Algorithm:

void function(int T):

```
read T from input
for each test case from 1 to T:
    read N1 from input
    read N1 elements into array A1

    read N2 from input
    read N2 elements into array A2

    set i = 0
    set j = 0

    while i < N1 and j < N2:
        if A1[i] < A2[j]:
            increment i by 1
        else if A1[i] > A2[j]:
            increment j by 1
        else:
            print A1[i]
            increment i by 1
            increment j by 1
    print a newline after each test case
```

Program:

```
#include <stdio.h>

int main() {
    int T;
    scanf("%d", &T);
```

```
while (T--) {
    int N1, N2;
    scanf("%d", &N1);
    int A1[N1];
    for (int i = 0; i < N1; i++) {
        scanf("%d", &A1[i]);
    }

    scanf("%d", &N2);
    int A2[N2];
    for (int i = 0; i < N2; i++) {
        scanf("%d", &A2[i]);
    }

    int i = 0, j = 0;
    int found = 0;
    while (i < N1 && j < N2) {
        if (A1[i] < A2[j]) {
            i++;
        } else if (A1[i] > A2[j]) {
            j++;
        } else {
            if (found) {
                printf(" ");
            }
            printf("%d", A1[i]);
            found = 1;
            i++;
```

```
            j++;
        }
    }
    printf("\n");
  }
  return 0;
}
```

Output:

	Input	Expected	Got	
✔	1 3 10 17 57 6 2 7 10 15 57 246	10 57	10 57	✔
✔	1 6 1 2 3 4 5 6 2 1 6	1 6	1 6	✔

6.d. Print Intersection of 2 sorted arrays-O(m+n)Time Complexity,O(1) Space Complexity

Aim:

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

· The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

1 6

Algorithm:

void function(int T):

```
read T from input
for each test case from 1 to T:
    read N1 from input
    read N1 elements into array A1

    read N2 from input
    read N2 elements into array A2

    set i = 0
    set j = 0

    while i < N1 and j < N2:
        if A1[i] < A2[j]:
            increment i by 1
        else if A1[i] > A2[j]:
            increment j by 1
        else:
            print A1[i]
            increment i by 1
            increment j by 1
    print a newline after each test case
```

Program:

```
#include <stdio.h>

int main() {
```

```
int T;
scanf("%d", &T);

while (T--) {
    int N1, N2;
    scanf("%d", &N1);
    int A1[N1];
    for (int i = 0; i < N1; i++) {
        scanf("%d", &A1[i]);
    }

    scanf("%d", &N2);
    int A2[N2];
    for (int i = 0; i < N2; i++) {
        scanf("%d", &A2[i]);
    }

    int i = 0, j = 0;
    while (i < N1 && j < N2) {
        if (A1[i] < A2[j]) {
            i++;
        } else if (A1[i] > A2[j]) {
            j++;
        } else {
            printf("%d ", A1[i]);
            i++;
            j++;
        }
```

```
        }
        printf("\n");
    }
    return 0;
}
```

Output:

	Input	Expected	Got	
✔	1 3 10 17 57 6 2 7 10 15 57 246	10 57	10 57	✔
✔	1 6 1 2 3 4 5 6 2 1 6	1 6	1 6	✔

6.e. Difference-O(n^2)Time Complexity,O(1) Space Complexity

Aim:

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

Algorithm:

void function(int n, int k):

read n from input

create array A of size n

for i = 0 to n - 1:

read A[i] from input

read k from input

set i = 0

set j = 1

```
    while i < n and j < n:
        set diff = A[j] - A[i]

        if i != j and diff == k:
            print 1
            return

        else if diff < k:
            increment j by 1

        else:
            increment i by 1

    print 0
```

Program:

```
#include <stdio.h>

int main() {
    int n, k;
    scanf("%d", &n);
    int A[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }
    scanf("%d", &k);

    int i = 0, j = 1;
    while (i < n && j < n) {
```

```
        int diff = A[j] - A[i];
        if (i != j && diff == k) {
            printf("1\n");
            return 0;
        }
        else if (diff < k) {
            j++;
        }
        else {
            i++;
        }
    }

    printf("0\n");
    return 0;
}
```

Output:

	Input	Expected	Got	
✔	3 1 3 5 4	1	1	✔
✔	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✔
✔	10 1 2 3 5 11 14 16 24 28 29 0	0	0	✔
✔	10 0 2 3 7 13 14 15 20 24 25 10	1	1	✔

6.f. Pair with Difference -O(n) Time Complexity,O(1) Space Complexity

Aim: Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

Algorithm:

```
void function(int n, int k){
    read n from input
    create array A of size n

    for i = 0 to n - 1:
        read A[i] from input

    read k from input

    set i = 0
    set j = 1
```

```
    while i < n and j < n:
        set diff = A[j] - A[i]

        if i != j and diff == k:
            print 1
            return

        else if diff < k:
            increment j by 1

        else:
            increment i by 1

    print 0
}
```

Program:

```
#include <stdio.h>

int main() {
    int n, k;
    scanf("%d", &n);
    int A[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }
    scanf("%d", &k);

    int i = 0, j = 1;
```

```
    while (i < n && j < n) {
        int diff = A[j] - A[i];
        if (i != j && diff == k) {
            printf("1\n");
            return 0;
        }
        else if (diff < k) {
            j++;
        }
        else {
            i++;
        }
    }

    printf("0\n");
    return 0;
}
```

Output:

	Input	Expected	Got	
✔	3 1 3 5 4	1	1	✔
✔	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✔
✔	10 1 2 3 5 11 14 16 24 28 29 0	0	0	✔
✔	10 0 2 3 7 13 14 15 20 24 25 10	1	1	✔