

NAME: SREYA G
NO.: 230701334

REGISTER

CLASS : CSE F
31/09/2024

DATE:

EX-6:

COMPETITIVE PROGRAMMING:

PROBLEM 1:

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

For example:

Input	Result
5 1 1 2 3 4	1

ALGORITHM:

1. Input integer n and array arr[] of size n.
2. Set d = -1 to track duplicate.
3. For each element j, compare it with every subsequent element k.
4. If arr[j] == arr[k], set d = arr[j] and break out of the loops.

5. If a duplicate is found ($d \neq -1$), output d ; otherwise, print "No duplicates found".

CODE:

```
#include<stdio.h>

int main()
{
    int n;
    scanf("%d",&n);
    int arr[n];
    int d = -1;
    for(int i = 0;i < n;i++)
    {
        scanf("%d",&arr[i]);
    }
    for(int j = 0;j < n;j++)
    {
        for(int k = j+1;k < n;k++)
        {
            if(arr[j] == arr[k])
            {
                d = arr[j];
                break;
            }
        }
    }
    if(d != -1)
    {
        break;
    }
}
```

```

    }
}
if(d != -1)
{
    printf("%d",d);
}
else
{
    printf("No duplicates found\n");
}
return 0;
}

```

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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 2:

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

For example:

Input	Result
5 1 1 2 3 4	1

ALGORITHM:

1. Input integer n and array a[] of size n.
2. For each element i, input the value and store it in a[i].
3. For each element i starting from 1, compare a[i] with a[j].
4. If a match is found, print a[i] and stop.
5. Otherwise, increment j and continue.

CODE:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int n,c;
```

```
    int j = 0;
```

```
scanf("%d",&n);
int a[n];
for(int i = 0;i <= n;i++)
{
    scanf("%d",&c);
    a[i] = c;
}
for(int i = 1;i <= n;i++)
{
    if(a[j] == a[i])
    {
        printf("%d",a[i]);
        break;
    }
    else
    {
        j++;
    }
}
}
```

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	✓
Passed all tests! ✓				
<div>Correct</div> Marks for this submission: 1.00/1.00.				

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 3:

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

For example:

Input	Result
1	10 57
3 10 17 57	
6	
2 7 10 15 57 246	

ALGORITHM:

1. Input the number of test cases T.
2. For each test case:
 - Input the size N1 and elements of array arr1[].
 - Input the size N2 and elements of array arr2[].
3. Initialize two pointers i = 0 and j = 0 for both arrays.
4. While both pointers are within bounds of their respective arrays:
 - If arr1[i] == arr2[j], print the value and move both pointers forward.
 - If arr1[i] < arr2[j], increment pointer i.
 - Otherwise, increment pointer j.
5. Print the intersection of the two arrays.

CODE:

```
#include<stdio.h>
```

```
void intersection(int arr1[],int N1,int arr2[],int N2){  
    int i = 0,j = 0;  
    int first = 1;
```

```

while(i < N1 && j < N2){
    if(arr1[i] == arr2[j]){
        if(!first) printf(" ");
        printf("%d",arr1[i]);
        first = 0;
        i++;
        j++;
    }else if(arr1[i] < arr2[j]){
        i++;
    }else{
        j++;
    }
}
printf("\n");
}

```

```

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

    while(T--){
        int N1,N2;

        scanf("%d",&N1);
        int arr1[N1];
        int i;
        for(i = 0;i < N1;i++){

```



```

        scanf("%d",&arr1[i]);
    }
    scanf("%d",&N2);
    int arr2[N2];
    for(i = 0;i <N2;i++){
        scanf("%d",&arr2[i]);
    }

    intersection(arr1,N1,arr2,N2);
}
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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 4:

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

For example:

Input	Result
1 3 10 17 57 6 2 7 10 15 57 246	10 57

ALGORITHM:

1. Input number of test cases T.

2. For each test case:
 - Input the size N1 and elements of array arr1[].
 - Input the size N2 and elements of array arr2[].
3. Initialize two pointers i = 0 and j = 0 to traverse both arrays.
4. While both pointers are within bounds:
 - If arr1[i] == arr2[j], print the element and move both pointers forward.
 - If arr1[i] < arr2[j], increment pointer i.
 - Otherwise, increment pointer j.
5. After processing all test cases, print the intersection of the two arrays for each test case.

CODE:

```
#include<stdio.h>
```

```
void intersection(int arr1[],int N1,int arr2[],int N2){
```

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

```
    int first = 1;
```

```
    while(i < N1 && j < N2){
```

```
        if(arr1[i] == arr2[j]){
```

```
            if(!first){
```

```
                printf(" ");
```

```
            }
```

```
            printf("%d",arr1[i]);
```

```
            first = 0;
```

```
            i++;
```

```
            j++;
```

```
    }else if(arr1[i] < arr2[j]){  
        i++;  
    }else{  
        j++;  
    }  
}  
printf("\n");  
}
```

```
int main(){  
    int T;  
    scanf("%d",&T);  
  
    while(T--){  
        int N1,N2;  
  
        scanf("%d",&N1);  
        int arr1[N1];  
        int i;  
        for(i = 0;i < N1;i++){  
            scanf("%d",&arr1[i]);  
        }  
  
        scanf("%d",&N2);  
        int arr2[N2];  
        for(i = 0;i < N2;i++){  
            scanf("%d",&arr2[i]);  
        }  
    }  
}
```

```

    }
    intersection(arr1,N1,arr2,N2);
}
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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 5:

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 $A[j] - A[i] = k$, $i \neq j$.

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.

For example:

Input	Result
3	1
1 3 5	
4	

ALGORITHM:

1. Input the size of the array n and the difference k.
2. Input the array elements arr[].
3. For each pair of elements arr[i] and arr[j] where $i < j$:
 - If the absolute difference between arr[j] and arr[i] equals k, return 1 (pair found).
 - If the difference is greater than k, break the inner loop (optimization).
4. If no such pair is found, return 0.
5. Output the result of the function findPairsWithDiff.

CODE:

```
#include<stdio.h>
```

```

int findPairsWithDiff(int arr[],int n,int k){
    int i,j;
    for(i = 0;i < n - 1;i++){
        for(j = i + 1;j < n;j++){
            if(arr[j] - arr[i] == k){
                return 1;
            }else if(arr[j] - arr[i] > k){
                break;
            }
        }
    }
    return 0;
}

```

```

int main(){
    int n,k;
    scanf("%d",&n);
    int arr[n];
    int i;
    for(i = 0;i < n;i++){
        scanf("%d",&arr[i]);
    }
    scanf("%d",&k);
    printf("%d\n",findPairsWithDiff(arr,n,k));
    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	✓
Passed all tests! ✓				

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 6:

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 $A[j] - A[i] = k$, $i \neq j$.

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.

For example:

Input	Result
3 1 3 5 4	1

ALGORITHM:

1. Input the size n of the array and the difference k.
2. Input the array elements arr[].
3. Initialize two pointers $i = 0$ and $j = 1$.
4. While $j < n$:
 - If $arr[j] - arr[i] == k$, return 1 (pair found).
 - If the difference is less than k, increment j to check the next element.
 - If the difference is greater than k, increment i and ensure $i < j$ by adjusting j if needed.
5. If no pair is found, return 0.
6. Output the result from findPairsWithDiff.

CODE:

```
#include<stdio.h>
```

```
int findPairsWithDiff(int arr[],int n,int k){
```

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

```
    while(j < n){
```

```
        if(arr[j] - arr[i] == k){
```

```
            return 1;
```

```
        }else if(arr[j] - arr[i] < k){
```

```
            j++;
```

```
        }else{
```

```
            i++;
```

```
            if(i == j){
```

```
                j++;
```

```
            }
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

```
int main(){
```

```
    int n,k;
```

```
    scanf("%d",&n);
```

```
    int arr[n];
```

```
    int i;
```

```
    for(i = 0;i < n;i++){
```

```

        scanf("%d",&arr[i]);
    }
    scanf("%d",&k);
    printf("%d\n",findPairsWithDiff(arr,n,k));
    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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Thus the code is executed successfully and gives the expected output.