

```
1 #include <stdio.h>
2 int main() {
3     int n;
4     scanf("%d", &n);
5     int A[n];
6     for (int i = 0; i < n; i++) {
7         scanf("%d", &A[i]);
8     }
9     int k;
10    scanf("%d", &k);
11    int i = 0, j = 1;
12    int found = 0;
13    while (i < n && j < n) {
14        if (i != j) {
15            int diff = A[j] - A[i];
16            if (diff == k) {
17                found = 1;
18                break;
19            } else if (diff < k) {
20                j++;
21            } else {
22                i++;
23            }
24        } else {
25            j++;
26        }
27    }
28    printf("%d\n", found);
29    return 0;
30 }
```

```
1 #include <stdio.h>
2 int main() {
3     int n;
4     scanf("%d", &n); // number of elements
5     int A[n];
6     for (int i = 0; i < n; i++) {
7         scanf("%d", &A[i]);
8     }
9     int k;
10    scanf("%d", &k); // non-negative integer k
11    int i = 0, j = 1; // two pointers
12    int found = 0;
13    while (i < n && j < n) {
14        if (i != j && A[j] - A[i] == k) {
15            found = 1;
16            break;
17        } else if (A[j] - A[i] < k) {
18            j++;
19        } else {
20            i++;
21        }
22    }
23    printf("%d\n", found);
24    return 0;
25 }
26 }
```

	Input	Expected	Got	
✓	3 1 3 5 4	1	1	✓

```
1 #include <stdio.h>
2
3 int main() {
4     int T;
5     scanf("%d", &T); // number of test cases
6
7     while (T--) {
8         int N1, N2;
9         scanf("%d", &N1);
10        int arr1[N1];
11        for (int i = 0; i < N1; i++)
12            scanf("%d", &arr1[i]);
13
14        scanf("%d", &N2);
15        int arr2[N2];
16        for (int i = 0; i < N2; i++)
17            scanf("%d", &arr2[i]);
18
19        int i = 0, j = 0;
20        int printed = 0;
21
22        // Two-pointer approach
23        while (i < N1 && j < N2) {
24            if (arr1[i] == arr2[j]) {
25                if (printed) printf(" ");
26                printf("%d", arr1[i]);
27                printed = 1;
28                i++;
29                j++;
30            } else if (arr1[i] < arr2[j]) {
31                i++;
32            } else {
33                j++;
34            }
35        }
36        printf("\n");
37    }
}
```

```
1 #include <stdio.h>
2 int main() {
3     int T;
4     scanf("%d", &T);
5     while (T--) {
6         int N1, N2;
7         scanf("%d", &N1);
8         int arr1[N1];
9         for (int i = 0; i < N1; i++) {
10            scanf("%d", &arr1[i]);
11        }
12        scanf("%d", &N2);
13        int arr2[N2];
14        for (int i = 0; i < N2; i++) {
15            scanf("%d", &arr2[i]);
16        }
17        int i = 0, j = 0;
18        int first = 1;
19
20        while (i < N1 && j < N2) {
21            if (arr1[i] == arr2[j]) {
22                if (!first) printf(" ");
23                printf("%d", arr1[i]);
24                first = 0;
25                i++;
26                j++;
27            }
28            else if (arr1[i] < arr2[j]) {
29                i++;
30            } else {
31                j++;
32            }
33        }
34        printf("\n");
35    }
36    return 0;
37 }
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int n;
6     scanf("%d", &n);
7
8     int *arr = (int *)malloc(n * sizeof(int));
9     for (int i = 0; i < n; i++) {
10         scanf("%d", &arr[i]);
11     }
12
13     int *visited = (int *)calloc(n + 1, sizeof(int)); // elements between 1 and n
14
15     int duplicate = -1;
16     for (int i = 0; i < n; i++) {
17         if (visited[arr[i]] == 1) {
18             duplicate = arr[i];
19             break;
20         }
21         visited[arr[i]] = 1;
22     }
23
24     printf("%d\n", duplicate);
25
26     free(arr);
27     free(visited);
28     return 0;
29 }
```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 int main() {
4     int n;
5     scanf("%d", &n);
6     int *arr = (int *)malloc(n * sizeof(int));
7     for (int i = 0; i < n; i++) {
8         scanf("%d", &arr[i]);
9     }
10    int *visited = (int *)calloc(n + 1, sizeof(int)); // n+1 because elements are 1 to n
11    int duplicate = -1;
12    for (int i = 0; i < n; i++) {
13        if (visited[arr[i]] == 1) {
14            duplicate = arr[i];
15            break;
16        }
17        visited[arr[i]] = 1;
18    }
19    printf("%d\n", duplicate);
20    free(arr);
21    free(visited);
22    return 0;
23 }
24

```

	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	✓