

Name: Sahil

UID: 22bCS10928

Section : 901_KPIT(B)

1. Searching a Number

Given an integer k and array arr . Your task is to return the position of the first occurrence of k in the given array and if element k is not present in the array then return -1.

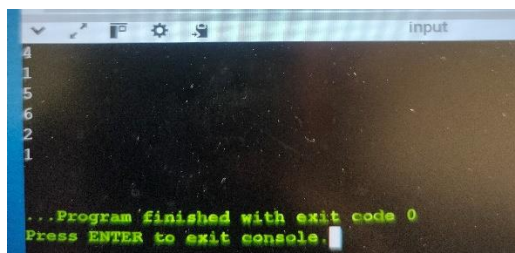
Note: 1-based indexing is followed here.

Answer:

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n, k, x;
    cin >> n >> k;
    vector<int> arr;
    for (int i = 0; i < n; i++) {
        cin >> x;
        arr.push_back(x);
    }
    for (int i = 0; i < n; i++) {
        if (arr[i] == k) {
            cout << i + 1;
            return 0;
        }
    }
    cout << -1;
    return 0;
}
```

OUTPUT:



2. Create Sorted Array through Instructions.

Given an integer array instructions, you are asked to create a sorted array from the elements in instructions. You start with an empty container nums. For each element from left to right in instructions, insert it into nums. The cost of each insertion is the minimum of the following:

The number of elements currently in nums that are strictly less than instructions[i].

The number of elements currently in nums that are strictly greater than instructions[i].

For example, if inserting element 3 into nums = [1,2,3,5], the cost of insertion is min(2, 1) (elements 1 and 2 are less than 3, element 5 is greater than 3) and nums will become [1,2,3,3,5].

Return the total cost to insert all elements from instructions into nums. Since the answer may be large, return it modulo $10^9 + 7$.

Answer:

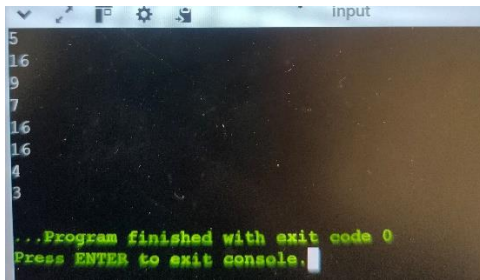
```
#include <iostream>
#include <vector>
using namespace std;

const int MOD = 1e9 + 7;
class FenwickTree {
    vector<int> tree;
public:
    FenwickTree(int size) : tree(size + 1, 0) {}

    void update(int idx, int value) {
        while (idx < tree.size()) {
            tree[idx] += value;
            idx += idx & -idx;
        }
    }

    int query(int idx) {
        int sum = 0;
        while (idx > 0) {
            sum += tree[idx];
            idx -= idx & -idx;
        }
        return sum;
    }
};
```

OUTPUT:

A screenshot of a terminal window with a dark background. The title bar at the top says 'input'. The terminal shows a list of numbers: 5, 16, 9, 7, 16, 16, 4, 3. Below this, it says '...Program finished with exit code 0' and 'Press ENTER to exit console.' with a cursor at the end of the line.

```
int createSortedArray(vector<int>& instructions) {
    int maxVal = 100000;
    FenwickTree fenwick(maxVal);
    long long cost = 0;

    for (int i = 0; i < instructions.size(); i++) {
        int num = instructions[i];
        int less = fenwick.query(num - 1);
        int greater = i - fenwick.query(num);
        cost = (cost + min(less, greater)) % MOD;
        fenwick.update(num, 1);
    }

    return cost;
}

int main() {
    int n;
    cin >> n;
    vector<int> instructions(n);
    for (int i = 0; i < n; i++) {
        cin >> instructions[i];
    }
    cout << createSortedArray(instructions) << endl;
    return 0;
}
```

3. Pair Sum Closet to 0.

Given an integer array of N elements. You need to find the maximum sum of two elements such that sum is closest to zero.

Answer:

```
#include <iostream>
#include <vector>
#include <algorithm>
```

```

#include <climits>
using namespace std;

int closestPairSum(vector<int>& arr) {
    sort(arr.begin(), arr.end());
    int left = 0, right = arr.size() - 1;
    int closestSum = INT_MAX;

    while (left < right) {
        int sum = arr[left] + arr[right];

        if (abs(sum) < abs(closestSum)) {
            closestSum = sum;
        } else if (abs(sum) == abs(closestSum) && sum > closestSum) {
            closestSum = sum; // If same absolute value, take the larger sum
        }

        if (sum < 0) {
            left++;
        } else {
            right--;
        }
    }

    return closestSum;
}

int main() {
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;

    vector<int> arr(n);
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    cout << "Sum closest to zero: " << closestPairSum(arr) << endl;
    return 0;
}

```

OUTPUT:

```
input
Enter the number of elements: 3
Enter the elements of the array: -8
-66
-60
Sum closest to zero: -68

...Program finished with exit code 0
Press ENTER to exit console.
```

4. Smallest Positive Missing Number.

You are given an integer array `arr[]`. Your task is to find the smallest positive number missing from the array.

Note: Positive number starts from 1. The array can have negative integers too.

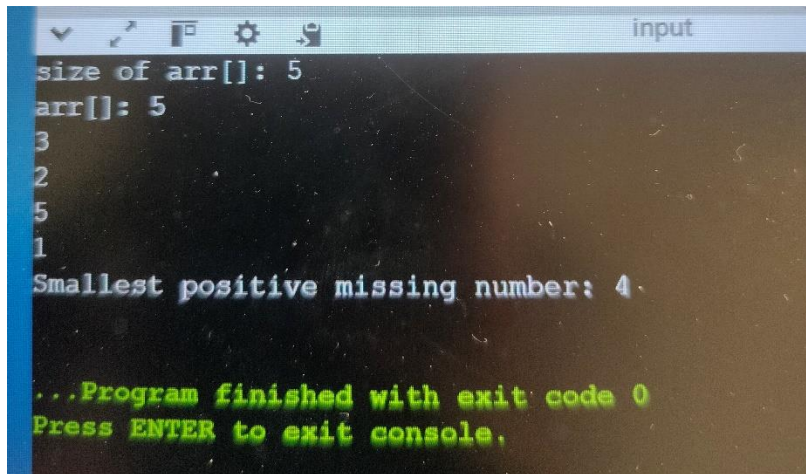
Answer:

```
#include <iostream>
#include <vector>
using namespace std;
```

```
int smallestMissingPositive(vector<int>& arr) {
    int n = arr.size();
    for (int i = 0; i < n; i++) {
        while (arr[i] > 0 && arr[i] <= n && arr[arr[i] - 1] != arr[i]) {
            swap(arr[i], arr[arr[i] - 1]);
        }
    }
    for (int i = 0; i < n; i++) {
        if (arr[i] != i + 1) {
            return i + 1;
        }
    }
    return n + 1;
}
```

```
int main() {
    int n;
    cout << "size of arr[]: ";
    cin >> n;
    vector<int> arr(n);
    cout << "arr[]: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    cout << "Smallest positive missing number: " << smallestMissingPositive(arr) << endl;
    return 0;
}
```

oUTpUT:

A screenshot of a console window titled 'input'. The output shows the size of an array as 5, followed by the array elements 3, 2, 5, 1. It then states 'Smallest positive missing number: 4'. At the bottom, it says '...Program finished with exit code 0' and 'Press ENTER to exit console.' in green text.

```
input
size of arr[]: 5
arr[]: 5
3
2
5
1
Smallest positive missing number: 4
...Program finished with exit code 0
Press ENTER to exit console.
```

5. Find Minimum in Rotated Sorted Array.

Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array `nums = [0,1,2,4,5,6,7]` might become:

`[4,5,6,7,0,1,2]` if it was rotated 4 times.

`[0,1,2,4,5,6,7]` if it was rotated 7 times.

Answer:

```
#include <iostream>
#include <vector>
using namespace std;

int findMin(vector<int>& nums) {
    int left = 0, right = nums.size() - 1;
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (nums[mid] > nums[right]) {
            left = mid + 1;
        } else {
            right = mid;
        }
    }
    return nums[left];
}

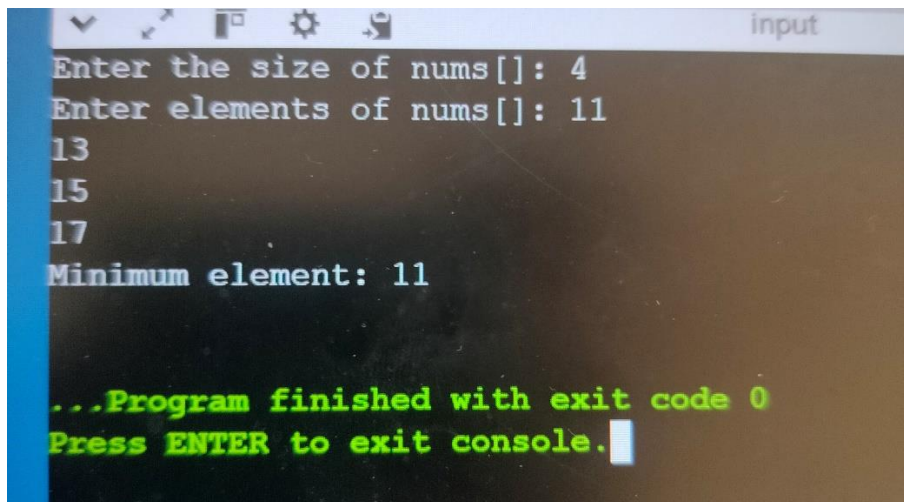
int main() {
    int n;
```

```
cout << "Enter the size of nums[]: ";
cin >> n;

vector<int> nums(n);
cout << "Enter elements of nums[]: ";
for (int i = 0; i < n; i++) {
    cin >> nums[i];
}

cout << "Minimum element: " << findMin(nums) << endl;
return 0;
}
```

OUTput:

A screenshot of a console window titled 'input'. The window shows the execution of a C++ program. The user has entered '4' for the size of the array and '11', '13', '15', '17' for the elements. The program outputs 'Minimum element: 11'. At the bottom, it says '...Program finished with exit code 0' and 'Press ENTER to exit console.' with a cursor. The console has a dark background with light blue text for prompts and output, and green text for the final status message.

```
input
Enter the size of nums[]: 4
Enter elements of nums[]: 11
13
15
17
Minimum element: 11
...Program finished with exit code 0
Press ENTER to exit console.
```