

```

1 #include <stdio.h>
2 void swap(int *a, int *b) {
3     int temp = *a;
4     *a = *b;
5     *b = temp;
6 }
7 int partition(int arr[], int low, int high) {
8     int pivot = arr[high]; // choosing last element as pivot
9     int i = (low - 1);
10    for (int j = low; j < high; j++) {
11        if (arr[j] <= pivot) {
12            i++;
13            swap(&arr[i], &arr[j]);
14        }
15    }
16    swap(&arr[i + 1], &arr[high]);
17    return (i + 1);
18 }
19 void quickSort(int arr[], int low, int high) {
20    if (low < high) {
21        int pi = partition(arr, low, high);
22        quickSort(arr, low, pi - 1);
23        quickSort(arr, pi + 1, high);
24    }
25 }
26 int main() {
27     int n;
28     scanf("%d", &n);
29     int arr[n];
30    for (int i = 0; i < n; i++) {
31        scanf("%d", &arr[i]);
32    }
33    quickSort(arr, 0, n - 1);
34    for (int i = 0; i < n; i++) {
35        printf("%d ", arr[i]);
36    }
37    return 0;

```

```
1 #include <stdio.h>
2 void findPair(int arr[], int left, int right, int x) {
3     if (left >= right) {
4         printf("No\n");
5         return;
6     }
7     int sum = arr[left] + arr[right];
8     if (sum == x) {
9         printf("%d\n", arr[left]);
10        printf("%d\n", arr[right]);
11        return;
12    }
13    if (sum > x) {
14        findPair(arr, left, right - 1, x);
15    }
16    else {
17        findPair(arr, left + 1, right, x);
18    }
19 }
20 int main() {
21     int n, x;
22     scanf("%d", &n);
23     int arr[n];
24     for (int i = 0; i < n; i++) {
25         scanf("%d", &arr[i]);
26     }
27     scanf("%d", &x);
28     findPair(arr, 0, n - 1, x);
29     return 0;
30 }
31
```

```
1 #include <stdio.h>
2 int findFloor(int arr[], int low, int high, int x) {
3     if (low > high)
4         return -1;
5     int mid = low + (high - low) / 2;
6     if (arr[mid] == x)
7         return arr[mid];
8     if (arr[mid] > x) {
9         return findFloor(arr, low, mid - 1, x);
10    }
11    int floorRight = findFloor(arr, mid + 1, high, x);
12    if (floorRight == -1)
13        return arr[mid];
14    else
15        return floorRight;
16 }
17 int main() {
18     int n, x;
19     scanf("%d", &n);
20     int arr[n];
21     for (int i = 0; i < n; i++) {
22         scanf("%d", &arr[i]);
23     }
24     scanf("%d", &x);
25     int result = findFloor(arr, 0, n - 1, x);
26     if (result == -1)
27         printf("No floor exists\n");
28     else
29         printf("%d\n", result);
30     return 0;
31 }
32
```

```

1 #include <stdio.h>
2 int majorityElement(int* nums, int n) {
3     int count = 0, candidate = 0;
4     for (int i = 0; i < n; i++) {
5         if (count == 0) {
6             candidate = nums[i];
7         }
8         if (nums[i] == candidate)
9             count++;
10        else
11            count--;
12    }
13    return candidate;
14 }
15 int main() {
16     int n;
17     scanf("%d", &n);
18     int nums[n];
19     for (int i = 0; i < n; i++) {
20         scanf("%d", &nums[i]);
21     }
22     int result = majorityElement(nums, n);
23     printf("%d\n", result);
24     return 0;
25 }
26

```

	Input	Expected	Got	
✓	3	3	3	✓
	3 2 3			

```

1 #include <stdio.h>
2 int firstZero(int arr[], int low, int high) {
3     if (high >= low) {
4         int mid = low + (high - low) / 2;
5         if ((mid == 0 || arr[mid - 1] == 1) && arr[mid] == 0)
6             return mid;
7         if (arr[mid] == 1)
8             return firstZero(arr, mid + 1, high);
9         return firstZero(arr, low, mid - 1);
10    }
11    return -1;
12 }
13 int main() {
14     int m;
15     scanf("%d", &m);
16     int arr[m];
17     for (int i = 0; i < m; i++) {
18         scanf("%d", &arr[i]);
19     }
20     int index = firstZero(arr, 0, m - 1);
21     if (index == -1) {
22         printf("0\n");
23     } else {
24         printf("%d\n", m - index);
25     }
26     return 0;
27 }
28

```

	Input	Expected	Got	
✓	5	2	2	✓
	1			