CS2.201: Computer Systems Organization
Spring 2025

International Institute of Information Technology, Hyderabad

Assignment 2
Mid submission deadline: 23:59, 31/03/2025
End submission deadline: 23:59, 09/04/2025

Instructions

- Writing complete code with successful execution guarantees full marks. Ensure that all edge cases are handled. Hard coded solutions will get a straight zero.
- Strict plagiarism checks will be performed on all submissions. Any and all forms of plagiarism will result in zero marks for this assignment.
- Comment your code properly explaining why you are doing what you are doing.
- Total marks for assignment is 50 (30 marks for working code submission, 20 marks for viva).
- C files should be used only to take input, call the function, and give output. The complete logic of the function itself should be in an Assembly file; failing this will result in zero marks for that question.
- For questions requiring an array as output, create an array in C and pass its pointer as one of the function arguments.
- Problems 1, 2, and 3 are due at mid submission, while problems 4 and 5 are due at end submission.
- Ensure that your submission works on a Linux-based OS, as that is what will be used to test your submission.

Submission format for the mid submission

- For the Mid submission, maintain the following directory structure:

```
<roll_number>
|-- q1
|   |-- q1.c
|   `-- q1.s
|-- q2
|   |-- q2.c
|   `-- q2.s
|-- q3
    |-- q3.c
    `-- q3.s
```

- Zip the `<roll_number>` folder, rename the zipfile to `<roll_number>_assign2_mid.zip` and submit.
- Ensure that upon extraction of the ZIP, the above tree is obtained as it is (with `<roll_number>` folder being there).

Submission format for the end submission

- For the End submission, maintain the following directory structure:

```
<roll_number>
|-- q4
|   |-- q4.c
|   `-- q4.s
`-- q5
    |-- q5.c
    `-- q5.s
```

- Zip the `<roll_number>` folder, rename the zipfile to `<roll_number>_assign2_end.zip` and submit.
- Ensure that upon extraction of the ZIP, the above tree is obtained as it is (with `<roll_number>` folder being there).

Late day Policy

- You can use 3 late days for the mid submission, with a 20% penalty that applies on that part only (i.e problem 1, 2 and 3 will have a 20% penalty)
- Similarly, you can use 3 late days for the end submission, with a 20% penalty that applies on that part only (i.e problem 4 and 5 will have a 20% penalty)

Problem 1 (6 marks)

Given an array of integers, determine the number of subarrays whose sum equals zero. The solution should be implemented in $O(n^2)$ time complexity or better.

Constraints

- The array contains both positive and negative integers.
- The length of the array N is at most 10^4.
- The solution must run in $O(n^2)$ time complexity or better.
- The maximum possible sum of any subarray is 1000.
- The minimum possible sum of any subarray is -1000.

Input

N
a1 a2 a3.....an

Output

number of subarrays with sum 0

Example Test Cases

Example 1

Input:

```
10
1 2 -3 3 1 -4 2 2 -2 3
```

Output:

```
6
```

Explanation:

The subarrays that sum to zero are:

- $\{1, 2, -3\}$
- $\{1, 2, -3, 3, 1, -4\}$
- $\{3, 1, -4\}$
- $\{2, -2\}$
- $\{-3, 3\}$
- $\{-4, 2, 2\}$

Example 2

Input:

```
10
4 -2 -2 2 3 -3 1 -1 2 -2
```

Output:

```
12
```

Explanation:

The subarrays that sum to zero are:

- $\{4, -2, -2\}$
- $\{-2, 2\}$
- $\{-2, 2, 3, -3\}$
- $\{3, -3\}$
- $\{-2, 2, 3, -3, 1, -1\}$
- $\{3, -3, 1, -1\}$
- $\{1, -1\}$
- $\{-2, 2, 3, -3, 1, -1, 2, -2\}$
- $\{3, -3, 1, -1, 2, -2\}$
- $\{1, -1, 2, -2\}$
- $\{2, -2\}$
- $\{-2, -2, 2, 3, -3, 1, -1, 2\}$

Problem 2 (6 marks)

The Proprietary Code Development Committee (PCDC) planned to host an "Introduction to Get" session and diligently submitted a proposal to the Coalition of Committees (CoC) weeks in advance. Despite multiple follow-ups, the CoC delayed approval until just three days before the event. As a result of this last-minute decision, the coordinators were unable to secure a suitably large venue, as all the larger rooms had already been booked. Instead, they were assigned a smaller space than required. A total of n participants have registered for the event, but there's only r seats available.

Unfortunately, they were unable to fit the remaining participants anywhere else. You have been brought in to solve the issue: pick a total of $n - r$ participants to kick out of the event. You can pick them however you want to, and the coordinators would like to know how many ways there are to do so. It is important to note that the coordinators love efficiency, assembly code and recursion (they're weirdos, but then again, aren't we all?).

So, **write a program to print the number of ways in which you can pick $n - r$ participants to kick out of the event, out of n total participants, using recursion in O(r) time complexity.**

Input Format

- n r

Output Format

An integer denoting the number of ways to pick $n - r$ participants to evict, out of n total participants.

Constraints

- $1 \leq n \leq 10$
- $0 \leq r \leq n$

Sample Input

```
4 2
```

Sample Output

```
6
```

Problem 3 (6 marks)

At IIIT, the Computer Systems Organization (CSO) course follows an unconventional marking scheme. Instead of assigning fixed marks to each question, the marks for a question depend on the next question with higher marks in the exam paper.

The Teaching Assistants (TAs) want to automate this process to avoid manual effort. Given an array where each element represents the marks of a question in sequential order, determine the new marks assigned to each question based on the next question with a greater value. If no such question exists, assign -1.

To ensure efficiency, implement the solution using stack operations in O(n) time complexity.

Input

- The first line contains a single integer N, the number of questions. $(1 \leq N \leq 10^5)$.
- The second line contains N space-separated integers $a_1, a_2, \ldots, a_N$, representing the marks of the questions. $(-10^9 \leq a_i < 10^9)$.

Output

- Print a single line containing N space-separated integers, where the i^{th} value is the Next Higher Marks Question or -1 if no such question exists.

Example

Sample Input

```
6
3 7 1 8 2 5
```

Sample Output

```
7 8 8 -1 5 -1
```

Problem 4 (6 marks)

Given an array nums of size N, return the **majority element**. The **majority element** is defined as the element that appears more than $\lfloor N/2 \rfloor$ times. It is guaranteed that such an element always exists in the array.
Time Complexity Required - **O(n)**
Hint - Look for Boyer-Moore Majority Voting Algorithm

Constraints

- $N = \text{len(nums)}$
- $1 \leq n \leq 5 \times 10^4$
- $0 \leq \text{nums}[i] \leq 10^9$

Input

N
a1 a2 a3 ..... an

Output

majority element

Examples

Example 1

Input

```
3
3 2 3
```

Output

```
3
```

Example 2

Input

```
7
2 2 1 1 1 2 2
```

Output

```
2
```

Problem 5 (6 marks)

It is the day of the deadline for your assignment. You glance at the clock, and your heart skips a beat - it's 11:58 PM. The assignment is due at 11:59 PM. A wave of relief washes over you as you remember that, for once, you had planned ahead. You had started working on the assignment on time, and it was already finished.

But then, a terrible realization strikes. You haven't submitted it on Moodle yet. Panic surges through your veins as you hurriedly open your laptop, your fingers flying across the trackpad to open Moodle. The page loads... or rather, it tries to. The spinning wheel of doom stares back at you, taunting you with its slow, deliberate rotations. Your mobile hotspot is dragging its feet at the worst possible time.

It is now too late to ask for an extension, so there is only one thing you can do now. You have N rooms in a sequence along the corridor your room is at. You can get into any room and check the signal value (assume that you get a 16-bit signed integer). Your goal is to find the signal **value at the peak**, so that you can submit your assignment on time. A room is said to have **peak** signal if it does not have a predecessor or successor room that has greater signal than it in value.

Remember, you absolutely do not have time to check all rooms one by one, so you have to do something smarter. Because of how the network towers are laid out, you realize that the sequence is guaranteed to have **unique peak value**, i.e, if there are multiple peaks in the sequence, they will all have the same value.

Constraints

- $1 \leq N \leq 10^6$
- L_i is a 16-bit signed integer.
- The algorithm must run in $\mathcal{O}(\log N)$ time (a linear scan naive approach has $\mathcal{O}(N)$ complexity and is therefore not allowed).
- L_i has a **unique peak value**, i.e, if there are two or more peaks, they will all have the same value.

Input Format

- N
- $L_0 \ldots L_{N-1}$

The size of the sequence N followed by N space separated values of the sequence.

Output Format

- P

The **value** P of the peak in L.

Example Input

```
5
-1 6 10 9 -3
```

Example Output

```
10
```

All the Best!