Offline Divide-and-Conquer and Greedy Algorithms

Problem 1 : Task Allocation among Employees

You are given a list of tasks, each having a certain amount of workload. The goal is to allocate all the tasks to a given number of employees. Each task must be assigned to exactly one employee, and the tasks assigned to an employee must form a contiguous segment of the original list.

Each employee must be assigned at least one task.

The objective is to minimize the maximum total workload assigned to any single employee.

Input:

- The first line contains two integers n and k — the number of tasks and the number of employees, respectively. ($1 \leq k, n \leq 10^5$)
- The second line contains n integers, where the i-th integer denotes the workload of the i-th task. ($1 \leq$ workload $\leq 10^9$)

Output:

- Output a single integer — the minimum possible value of the maximum workload assigned to any employee.
- If it is not possible to allocate all tasks to the given number of employees according to the rules, output -1.

Constraints:

- Each employee must get at least one task.
- Tasks assigned to an employee must be contiguous in the original list.
- The order of tasks cannot be changed.
- You must use a divide and conquer based algorithm. Otherwise, your solution will not be accepted.

Examples:

Example 1:

- Input:

```
5 2
10 20 30 40 50
```

- Output:

```
90
```

- Explanation:

  One optimal way is to allocate:

  - Employee 1: Tasks [10, 20, 30] — total workload = 60
  - Employee 2: Tasks [40, 50] — total workload = 90

  The maximum workload among all employees is 90, which is minimized.

  Other splits like [10, 20] and [30, 40, 50] would lead to a maximum workload of 120, which is worse.

Example 2:

- Input:

```
4 5
5 10 15 20
```

- Output:

```
-1
```

- Explanation:

  There are only 4 tasks but 5 employees. It is impossible to allocate at least one task to each employee.

Example 3:

- Input:

```
6 3
5 10 15 20 25 30
```

- Output:

```
45
```

- Explanation:

  One possible allocation:

  - Employee 1: [5, 10, 15] — total = 30
  - Employee 2: [20, 25] — total = 45
  - Employee 3: [30] — total = 30

  Maximum workload is 45, which is minimized.

Problem 2 : Bench-Based Student Seating

You are managing seating arrangements for an exam. The exam hall has a row of n seats, divided evenly into b benches. Each bench contains exactly $\frac{n}{b}$ seats. Some seats are already occupied by students of two types, each having specific restrictions. Additionally, you are tasked with seating new students of a third type.

You are given:

- An integer n — total number of seats.
- An integer b — number of benches.
- An integer array `seats` of size n, where:
  - `seats[i] = 1` means the seat is occupied by a type 1 student, who blocks the seat to their **right within the same bench**.
  - `seats[i] = 2` means the seat is occupied by a type 2 student, who blocks the seats to their **left and right within the same bench**.
  - `seats[i] = 0` means the seat is empty.
- An integer m — number of new students of type 3 that you need to seat.

Type 3 students have no personal restrictions. However, they **cannot be seated in any position that violates the blocking rules of type 1 or type 2 students**.

Return `true` if it is possible to seat all m type 3 students following the restrictions. Otherwise, return `false`.

Now, you are required to place m new students of **type 3** in the empty seats, such that:

- Type 3 students can be seated in any empty seat that is not blocked by restrictions of neighboring type 1 or type 2 students **within the same bench**.
- There is no restriction that prevents students from sitting adjacent across bench boundaries. Restrictions apply only **within individual benches**.

Your task is to determine whether it is possible to place all m new students following the rules above.

Input

- An integer n — total number of seats.
- An integer b — number of benches in the classroom. ($n \bmod b = 0$)
- An array `seats` of length n — the current seating configuration.
- An integer m — the number of new students to be seated.

Output

`true` if it is possible to place all m new students according to the given rules; otherwise, `false`.

Examples

<table>
<tr><th>Example 1</th><th>Example 2</th></tr>
<tr><td>Input:
$n = 8$, $b = 2$
<code>seats = [1, 0, 0, 0, 2, 0, 0, 0]</code>
$m = 2$</td><td>Input:
$n = 12$, $b = 3$
<code>seats = [1, 0, 2, 0, 0, 0, 1, 0, 2, 0, 1, 0]</code>
$m = 3$</td></tr>
<tr><td>Output:
<code>true</code>
Explanation:
There are 2 benches of 4 seats each.
Bench 0: [1, 0, 0, 0]
Seat 1 is blocked by type 1 at seat 0.
Seats 2 and 3 are free and not adjacent to any blockers or to each other.Therefore, place students at seats 2 and 3.
Bench 1: [2, 0, 0, 0]
Seat 5 is blocked by type 2 at seat 4.
Only 2 students need to be seated and we already placed them in bench 0. So it's possible.</td><td>Output:
<code>false</code>
Explanation:
There are 3 benches of 4 seats each.
Bench 0: [1, 0, 2, 0]
Seat 1 is blocked by type 1 (seat 0) and also by type 2 (seat 2).
Seat 3 is also blocked by type 2 (seat 2).
No placements are possible in this bench.
Bench 1: [0, 0, 1, 0]
Seat 6 is type 1, so seat 7 is blocked.
Seats 4 and 5 are free, so two students can be placed here.
Bench 2: [2, 0, 1, 0]
Seat 8 is type 2 — blocks seat 9.
Seat 10 is type 1 — blocks seat 11.
So, no seat is available here.
Total seats available: at most 2.
Required: 3 students.
So, not possible.</td></tr>
</table>

Submission Guidelines

1. Create a new folder named with your 7-digit student ID.

2. Copy all your source files into the folder created in Step 1.
3. Compress (zip) the folder. Ensure the compressed file has a ".zip" extension.
4. Upload the zip file to the designated submission link on Moodle.

Important Notes:

- Please carefully follow the submission guidelines. Failure to comply may result in a penalty of up to 10% of the total marks.
- Ensure you upload the correct and uncorrupted zip file. Submitting an incorrect or corrupted file may affect your grade.
- Do not submit solutions from previous assignments or unrelated files.
- Copying code from other students or external sources is strictly prohibited and may result in a penalty of up to −**100%** of the total marks. We expect honesty and integrity from all students.