Lab – 08

17/03/2025

Tree ADT – Binary Tree

Note:

1. Use only Visual Studio code type your program and run your code.
2. Always follow industry coding best practices.

A. Utilize C++ STL to solve the following **(K5)**,

There are n block towers, numbered from 1 to n. The i-th tower consists of a_i blocks.

In one move, you can move one block from tower i to tower j, but only if $a_i > a_j$. That move increases a_j by 1 and decreases a_i by 1. You can perform as many moves as you would like (possibly, zero).

What's the largest amount of blocks you can have on the tower 1 after the moves?

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of testcases.

The first line of each testcase contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of towers.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — the number of blocks on each tower.

The sum of n over all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase, print the largest amount of blocks you can have on the tower 1 after you make any number of moves (possibly, zero).

Example

input

```
4
3
1 2 3
3
1 2 2
2
1 1000000000
10
3 8 6 7 4 1 2 4 10 1
```

output

```
3
2
500000001
9
```

Note

In the first testcase, you can move a block from tower 2 to tower 1, making the block counts $[2, 1, 3]$. Then move a block from tower 3 to tower 1, making the block counts $[3, 1, 2]$. Tower 1 has 3 blocks in it, and you can't obtain a larger amount.

In the second testcase, you can move a block from any of towers 2 or 3 to tower 1, so that it has 2 blocks in it.

In the third testcase, you can 500000000 times move a block from tower 2 to tower 1. After that the block countes will be $[500000001, 500000000]$.

B. Write a separate C++ menu-driven program to implement Tree ADT using a character binary tree. Maintain proper boundary conditions and follow good coding practices. The Tree ADT has the following operations,

1. Insert
2. Preorder
3. Inorder
4. Postorder
5. Search
6. Exit

What is the time complexity of each of the operations? **(K4)**