Question - 1

Merge Strings

Complete the *mergeStrings* function in your editor. It has *2* parameters:

1. A string, *a*.
2. A string, *b*.

Your function must *merge* strings *a* and *b*, and then return a single merged string. A *merge* operation on two strings is described as follows:

- Append alternating characters from *a* and *b*, respectively, to some new string, *mergedString*.
- Once all of the characters in one of the strings have been merged, append the remaining characters in the other string to *mergedString*.

Input Format

The locked stub code in your editor reads two strings, *a* and *b*, from stdin and passes them to your function.

Constraints

- $1 \leq |a|, |b| \leq 25000$

Output Format

Your function must return the *merged* string. This will be printed to stdout by the locked stub code in your editor.

Sample Input 1

```
abc
def
```

Sample Output 1

```
adbecf
```

Sample Input 2

```
ab
zsd
```

Sample Output 2

```
azbsd
```

Explanation

Sample Case 1

a = abc

b = def

Taking alternate characters from both the strings , we

a = ab
b = *zsd*
Taking alternate characters from both the strings , we get *azbsd*

Question - 2

HackLand Election

There are *n* citizens voting in this year's *HackLand election*. Each voter writes the name of their chosen candidate on a ballot and places it in a ballot box.
The candidate with the highest number of votes wins the election;
if two or more candidates have the same number of votes, then the tied candidates' names are ordered alphabetically and the *last* name wins.

Complete the *electionWinner* function in your editor. It has *1* parameter: an array of strings, *votes*, describing the votes in the ballot box. This function must review these votes and return a string representing the name of the winning candidate.

Input Format
The locked stub code in your editor reads the following input from stdin and passes it to your function:
The first line contains an integer, *n*, denoting the size of the *votes* array.
Each line *i* of the *n* subsequent lines (where $0 \leq i < n$) of strings contains a citizen's vote in the form of a candidate's name.

Constraints
- $1 \leq n \leq 10^4$

Output Format
Your function must return a *string* denoting the name of the *winner*. This is printed to stdout by the locked stub code in your editor.

Sample Input 1

```
10
Alex
Michael
Harry
Dave
Michael
Victor
Harry
Alex
Mary
Mary
```

Sample Output 1

```
Michael
```

Explanation 1
votes = {"Alex", "Michael", "Harry", "Dave", "Michael",

"Victor", "Harry", "Alex", "Mary", "Mary"}
Alex, Harry, Michael, and Mary are all tied for the highest number of votes. Because Michael is alphabetically last, we return his name as the winner.

Sample Input 2

```
10
Victor
Veronica
Ryan
Dave
Maria
Maria
Farah
Farah
Ryan
Veronica
```

Sample Output 2

```
Veronica
```

Explanation 2
votes = {"Victor", "Veronica", "Ryan", "Dave", "Maria", "Maria", "Farah", "Farah", "Ryan", "Veronica"}
Veronica, Ryan, Maria, and Farah are all tied for the highest number of votes. Because Veronica is alphabetically last, we return her name as the winner.

Question - 3
Minimum Amount

Alex is shopping at a flea market and stops at a stand displaying a row of N items numbered from 0 to $N-1$ where the i^{th} ($0 \leq i \leq N-1$) item has a cost, c_i.

Noticing Alex's interest, the stand owner makes the following offer: if Alex agrees to purchase all N items, the owner will discount each item i by the amount of the cheapest item to the left of i^{th} item.

In other words, discount d_i for item i is the minimum c_k where $0 \leq k < i$; if discount d_i is greater than cost c_i, Alex gets item i for free. The very first item ($i = 0$) must be purchased at full price (without a discount).

Complete the *calculateAmount* function (which takes the array of costs, *arr*, as a parameter) so that it calculates the total amount Alex must pay to buy all N items and then returns that result as a *long*.

Input Format
The *calculateAmount* function has a parameter, *arr*, which is the array of integers denoting the *cost* of each item.

The locked code in the editor handles reading the following input from stdin, assembling it into an array of integers (*prices*), and calling *calculateAmount(prices)*.

The first line of input contains N, the number of items for sale. Each line i of the N subsequent lines describes the i^{th} item cost as an integer, c_i.

Constraints

- $1 \leq N \leq 2 \times 10^6$
- $1 \leq c_i \leq 2 \times 10^6$, where $1 \leq i \leq N$

Output Format

Your *calculateAmount* function should return a *long* denoting the amount paid by Alex to purchase each of the N items. The locked stub code in the editor will then print this result.

Sample Input 1

```
4
4
9
2
3
```

Sample Output 1

```
10
```

Sample Input 2

```
4
1
2
3
4
```

Sample Output 2

```
7
```

Explanation

Sample Case 1:
$N = 4, prices = \{4, 9, 2, 3\}$
$c_0 = 4$; $d_0 = 0$; $price_0 = 4$ *(the first item is not discounted)*
$c_1 = 9$; $d_1 = 4$; $price_1 = 9 - 4 = 5$
$c_2 = 2$; $d_2 = min(4,9) = 4$; $price_2 = 0$ because $d_2 > c_2$
$c_3 = 3$; $d_3 = min(4,9,2) = 2$; $price_3 = 3 - 2 = 1$

The total *cost (result)* returned by our *calculate_amount* function is $4 + 5 + 0 + 1 = 10$.

Sample Case 2
$N = 4, prices = \{1, 2, 3, 4\}$
$c_0 = 1$; $d_0 = 0$; $price_0 = 1$ *(the first item is not discounted)*
$c_1 = 2$; $d_1 = 1$; $price_1 = 2 - 1 = 1$
$c_2 = 3$ $d_2 = min(1, 2) = 1$; $price_2 = 3 - 1 = 2$
$c_3 = 4$; $d_3 = min(1, 2, 3) = 1$; $price_3 = 4 - 1 = 3$

The total *cost (result)* returned by our *calculate_amount* function is $1 + 1 + 2 + 3 = 7$.

Question - 4

ExcludeAndSum

Given 3 int values, a b c, return their sum. However, if any of the values is a teen -- in the range 13..19 inclusive -- then that value counts as 0, except 15 and 16 that do not count as a teens.

Examples:
noTeenSum(1, 2, 3) → 6
noTeenSum(2, 13, 1) → 3
noTeenSum(2, 1, 15) → 18

Question - 5
Armstrong Numbers

A positive integer is called an Armstrong number if the sum of cubes of individual digit is equal to that number itself. For example:
153 = 1*1*1 + 5*5*5 + 3*3*3 // 153 is an Armstrong number.
12 is not equal to 1*1*1+2*2*2 // 12 is not an Armstrong number.
Write a Program that checks whether a given number is a Armstrong number and prints "Armstrong Number" or "Not a Armstrong Number"

Sample Input : 153
Sample Output : Armstrong Number

Sample Input:12
Sample Output : Not an Armstrong Number