Salad Spree

Background

Meet Tom, software developer by profession, but only because noone has figured out how to make a living out of his favourite hobbies, consuming salad and being lazy.
Tom will go to the greatest lengths to not move a muscle in his body, rumour has it that he went a month without eating because the dishes weren't clean.
While he wears his laziness like a badge of honour, he has recently discovered that it's possible to be lazy and healthy at the same time.
Tom has discovered that the miracle food - 'Salad' is the panacea to all his problems. He can consume nearly infinite amounts of it while putting on no weight.

Problem

Alas, there still is a thorn in his plan, he needs to walk to stores to buy the amounts of salads he requires, and due to recent food shortages, each store that sells salads is only selling one salad per customer.
Tom is also incredibly miser and wants to make sure that he spends the least amount of money possible procuring his salads.
He has crafted an intelligent plan and decided that he will only buy salad from stores that are next to each other, that way he minimises the walking that he needs to do.
However, all this thinking has rendered Tom too tired and therefore he has hired you to write a program which will tell him the least amount of money he would need to spend to buy the number of salads he needs from consecutive stores.
Tom is too lazy and if there aren't enough consecutive stores selling the number of salads he needs, he will not buy any salads, therefore your solution should return 0 in that case.

Endpoint

Provide a `POST` endpoint `/salad-spree` that given 1 set of input will return 1 set of output

Input

The `HTTP POST` request will come with a body of `Content-Type: application/json` .

```
{
    "number_of_salads" : n,
    "salad_prices_street_map" : S (S is an array of arrays where each array indicates the price of salads in each store on a certain street. A price of "X" indicates that the store doesn't sell salads)
}
```

Output

The expected `HTTP` response will come with a body of `Content-Type: application/json` containing

```
{
    result : (The minimum amount of money that Tom needs to spend to buy n salads from n consecutive stores that sell salad)
}
```

Constraints

- $1 \leq n \leq$ *size of each array inside* $S \leq 100$
- HTTP Request Timeout: 500 milliseconds

i.e. You should be expecting the number of required salads to be smaller or equal to the number of shops per street. Both of these numbers should not exceed 100.

Examples

Sample Input 1

```
{
    "number_of_salads" : 3,
    "salad_prices_street_map" : [["12", "12", "3", "X", "3"], ["23", "X", "X", "X", "3"], ["33", "21", "X", "X", "X"], ["9", "12", "3", "X", "X"], ["X", "X", "X", "4", "5"]]
}
```

Sample Output 1

```
{
    "result": 24
}
```

Explanation 1

The solution in this case is on the 4th street, with 9, 12 and 3 being the lowest possible combination of three consecutive shops to buy salad from.

Sample Input 2

```
{
    "number_of_salads" : 3,
    "salad_prices_street_map" : [["X", "X", "2"], ["2", "3", "X"], ["X", "3", "2"]]
}
```

Sample Output 2

```
{
    "result": 0
}
```

Explanation 2

Your solution should return 0, since no street has three consecutive stores that sell salads.

Sample Input 3

```
{
    "number_of_salads" : 2,
    "salad_prices_street_map" : [["2", "3", "X", "2"], ["4", "X", "X", "4"], ["3", "2", "X", "X"], ["X", "X", "X", "5"]]
}
```

Sample Output 3

```
{
    "result": 5
}
```

Explanation 3

Even though there is more than one solution, the correct answer is still 5.