Problem I
Überwatch

The lectures are over, the assignments complete and even those pesky teaching assistants have nothing left to criticize about your coding project. Time to play some video games! As always, your procrastinating self has perfect timing: Cold Weather Entertainment just released *Überwatch*, a competitive first person video game!

Sadly, you aren't very good at these kind of games. However, Überwatch offers more than just skill based gameplay. In Überwatch you can defeat all opponents in view with a single button press using your ultimate attack. The drawback of this attack is that it has to charge over time before it is ready to use. When it is fully charged you can use it at any time of your choosing. After its use it immediately begins to charge again.

With this knowledge you quickly decide on a strategy:

- Hide from your opponents and wait for your ultimate attack to charge.
- Wait for the right moment.
- Defeat all opponents in view with your ultimate attack.
- Repeat.

After the game your teammates congratulate you on your substantial contribution. But you wonder: How many opponents could you have defeated with optimal timing?

The game is observed over n time slices. The ultimate attack is initially not charged and requires m time slices to charge. This first possible use of the ultimate attack is therefore in the $(m+1)$-th time slice. If the ultimate attack is used in the i-th time slice, it immediately begins charging again and is ready to be fired in the $(i+m)$-th time slice.

Input

The input consists of:

- one line with two integers n and m, where
  - n ($1 \leq n \leq 300\,000$) is the game duration;
  - m ($1 \leq m \leq 10$) is the time needed to charge the ultimate attack in time slices.
- one line with n integers x_i ($0 \leq x_i \leq 32$) describing the number of opponents in view during a time slice in order.

Output

Output the maximum number of opponents you can defeat.

Sample Input 1

```
4 2
1 1 1 1
```

Sample Output 1

```
1
```

Sample Input 2

```
9 3
1 1 2 2 3 2 3 2 1
```

Sample Output 2

```
5
```