Memories

In life, there are many things that are valuable to you. Some people believe that the most valuable thing that you have is your memory. One must always cherish their memory, no matter how bitter, how sweet, how happy, or how sad it is. Each memory is a treasure that is valuable to you throughout your life.

Now, imagine that you are given the chance to walk down your memory lane. The memory lane is a straight path of size **1 x N**. However, walking in the memory lane takes a lot of your energy. You can only afford walking for **M** hours in the memory lane. Fortunately, you are given the chance to pick the position of entry into the memory lane. Once inside, you **will spend** the whole of your **M hours** walking.

The memory lane is composed of N blocks, each containing a precious memory of your life. You will spend 1 hour at each block to reflect and cherish that particular memory. You might feel sad remembering that memory, whether it is a beautiful one or not. You might shed a few tears when it's a beautiful memory, especially if it involves your loved ones and knowing that you can never go back and experience it ever again for the rest of your life.

Knowing that some memories are more precious than others, you want to be able to select starting point such that **you will get the best experience**. The best experience is defined by **the most precious memory** that you encounter while walking down the memory lane. It does not matter how precious the other memories are as you will only cherish the one with the most powerful impact on your soul.

In order to do so, you roughly recall how precious your memories are in each block and you want to know, for every possible point of entry, what is the value of the most precious memory that you will encounter while walking for M hours inside the memory lane when you enter at that cell. That is, after walking **M blocks** from each point of entry, you want to know what is the value of the most precious memory that you see.

Input

The first line of input consists of two space-separated integers, **N and M (1 <= M <= N <= 100 000)**, the length of the memory lane and the number of hours you can walk inside the memory lane respectively.

The next line consists of N space-separated (**positive**) integers at most 1000, each denoting the value of the memory located at each block inside the memory lane, starting from the beginning to end.

Output

Print, in a single line, the value of the most precious memory for each point of entry that is possible. Your output must contain a newline character. Note that there is no whitespace after the last number.

Sample Input	Sample Output
5 2 1 4 3 2 5	4 4 3 5

Explanation

You can walk for at most two hours. If you enter at the first point of entry, you will get [1, 4]. Therefore, the maximum is 4. The next point of entry will give you [4, 3]. The maximum is still 4. The third point of entry will give you [3, 2]. This time, the maximum is 3, so you should print 3. The fourth **and last** point of entry will give you [3,5], hence you need to print 5, the most valuable memory inside there.

Skeleton

You are given the skeleton file `Memories.java`. You should see a non-empty file when you open the skeleton, otherwise you might be in the wrong directory.

```
/**
 * Name          :
 * Matric No.    :
 * PLab Acct.    :
 */

public class Memories {

    private void run() {
        // treat this as your "main" method
    }

    public static void main(String[] args) {
        Memories myMemory = new Memories();
        myMemory.run();
    }
}
```

Notes

1. You should develop your program in the subdirectory **ex1** and use the skeleton file provided.
2. You are free to use anything to solve this problem.
3. This problem is worth 30% of the total PE marks.
4. You will get at most:
   - **80%** of the total marks you receive if your algorithm runs worse than **O(MN).**
   - **90%** of the total marks you receive if your algorithm runs in **O(MN).**
   - **100%** of the total marks you receive if your algorithm runs in **O(N).**
5. Please be reminded that the marking scheme is:

   Input : 10%
   Output : 10%
   Correctness : 50%
   Programming Style : 30% (awarded if you score **at least 20% from the above**):
   - Meaningful comments (pre- and post- conditions, comments inside the code): 10%
   - Modularity (modular programming, proper modifiers [public / private]): 10%
   - Proper Indentation: 5%
   - Meaningful Identifiers (for both method and variable names): 5%

   Compilation Error : Deduction of 50% of the total marks obtained.