Product Distribution

locked

A company has requested to streamline their product allocation strategy, and given n products, each of which has an associated value, you are required to arrange these products into segments for processing. There are infinite segments indexed as 1, 2, 3 and so on.

However, there are two constraints:

- You can assign a product to a segment with index i if and only if $i = 1$ or the segment with index $i - 1$ has at least m products.
- Any segment must contain either no products or at least m products.

The score for a segment is defined as the index of the segment multiplied by the sum of values of the products it contains. The score of an arrangement of products is the sum of scores of segments. Your task is to compute the maximum score of an arrangement.

Consider, for example, $n = 11$ products and $m = 3$. One optimal way to assign is -

1. Assign the first three products to segment 1.
2. Assign the next three products to segment 2.
3. Assign the next five products to segment 3.

Note that we can not assign 2 products to segment 4 as the second constraint would be violated. The score of the above arrangement is -

1 * (1 + 2 + 3) + 2 * (4 + 5 + 6) + 3 * (7 + 8 + 9 + 10 + 11) = 6 + 30 + 135 = 171.

Since the arrangement score can be very large, print it modulo $10^9 + 7$.

Input Format

In the first line, there are two space-separated integers n and m.

In the second line, there are n space-separated integers $a_0, a_1, \ldots, a_{n-1}$ denoting the values associated with the products.

Constraints

- $1 \leq n \leq 10^6$
- $1 \leq m \leq n$
- $1 \leq a_i \leq 10^9$

Output Format

In a single line, print a single integer denoting the maximum score of the arrangement modulo $10^9 + 7$.

Sample Input 0

```
5 2
1 5 4 2 3
```

Sample Output 0

```
27
```

Explanation 0

The array is $a = [1, 5, 4, 2, 3]$ and $m = 2$. It is optimal to put the first and fourth products into the first segment and the remaining products to the second segment. Doing that, we get the arrangement score $(1 + 2) \cdot 1 + (3 + 4 + 5) \cdot 2 = 27$ which is the greatest score that can be obtained. Finally, the answer is 27 modulo $10^9 + 7$ which is 27.

Sample Input 1

```
4 4
4 1 9 7
```

Sample Output 1

```
21
```

Explanation 1

All the four products must be placed in the first segment. The score in this case will be 1 * (4 + 1 + 9 + 7) = 21.

Submissions: 4189
Max Score: 10
Difficulty: Easy

Rate This Challenge:
☆☆☆☆☆

More

Current Buffer (saved locally, editable)

Java 8

```java
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;

class Result {

    /*
     * Complete the 'maxScore' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts following parameters:
     *  1. INTEGER_ARRAY a
     *  2. INTEGER m
     */

    public static int maxScore(List<Integer> a, int m) {
    // Write your code here

    }

}

public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT_PATH")));

        String[] firstMultipleInput = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

        int n = Integer.parseInt(firstMultipleInput[0]);

        int m = Integer.parseInt(firstMultipleInput[1]);

        List<Integer> a = Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))
            .map(Integer::parseInt)
            .collect(toList());

        int ans = Result.maxScore(a, m);

        bufferedWriter.write(String.valueOf(ans));
        bufferedWriter.newLine();

        bufferedReader.close();
        bufferedWriter.close();
    }
}

```

Line: 1 Col: 1

Upload Code as File □ Test against custom input Run Code Submit Code