#include <assert.h>

#include <ctype.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

char\* ltrim(char\*);

char\* rtrim(char\*);

char\*\* split\_string(char\*);

int parse\_int(char\*);

/\*

\* 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

\*/

//////////////

void swap(int\* a, int\* b)

{

long long int t = \*a;

\*a = \*b;

\*b = t;

}

int partition (int a[], int low, int high)

{

long long int pivot = a[high]; // pivot

long long int i = (low - 1); // Index of smaller element

for (long long int j = low; j <= high- 1; j++)

{

// If current element is smaller than the pivot

if (a[j] < pivot)

{

i++; // increment index of smaller element

swap(&a[i], &a[j]);

}

}

swap(&a[i + 1], &a[high]);

return (i + 1);

}

void quickSort(int a[], int low, int high)

{

if (low < high)

{

long long int pi = partition(a, low, high);

quickSort(a, low, pi - 1);

quickSort(a, pi + 1, high);

}

}

////////////////////

int maxScore(int a\_count, int\* a, int m) {

//first sort the array

quickSort(a, 0, a\_count-1);

long long int i;

//list is successfully sorted

long long int sets;

sets = a\_count / m;

long long int sum =0;

int remainder ;

remainder = a\_count - sets\*m;

long long int multiplier;

multiplier = 1;

for(i=0;i<sets\*m;i++)

{

if(i!=0 && i%m==0)

{

multiplier++;

}

sum = sum + a[i] \* multiplier;

}

if(remainder > 0)

{

for(i=sets\*m;i<sets\*m+remainder;i++)

{

sum = sum + a[i] \* multiplier;

}

}

return sum % ((int)pow(10,9)+7);

}

int main()

{

FILE\* fptr = fopen(getenv("OUTPUT\_PATH"), "w");

char\*\* first\_multiple\_input = split\_string(rtrim(readline()));

int n = parse\_int(\*(first\_multiple\_input + 0));

int m = parse\_int(\*(first\_multiple\_input + 1));

char\*\* a\_temp = split\_string(rtrim(readline()));

int\* a = malloc(n \* sizeof(int));

for (int i = 0; i < n; i++) {

int a\_item = parse\_int(\*(a\_temp + i));

\*(a + i) = a\_item;

}

int ans = maxScore(n, a, m);

fprintf(fptr, "%d\n", ans);

fclose(fptr);

return 0;

}

char\* readline() {

size\_t alloc\_length = 1024;

size\_t data\_length = 0;

char\* data = malloc(alloc\_length);

while (true) {

char\* cursor = data + data\_length;

char\* line = fgets(cursor, alloc\_length - data\_length, stdin);

if (!line) {

break;

}

data\_length += strlen(cursor);

if (data\_length < alloc\_length - 1 || data[data\_length - 1] == '\n') {

break;

}

alloc\_length <<= 1;

data = realloc(data, alloc\_length);

if (!data) {

data = '\0';

break;

}

}

if (data[data\_length - 1] == '\n') {

data[data\_length - 1] = '\0';

data = realloc(data, data\_length);

if (!data) {

data = '\0';

}

} else {

data = realloc(data, data\_length + 1);

if (!data) {

data = '\0';

} else {

data[data\_length] = '\0';

}

}

return data;

}

char\* ltrim(char\* str) {

if (!str) {

return '\0';

}

if (!\*str) {

return str;

}

while (\*str != '\0' && isspace(\*str)) {

str++;

}

return str;

}

char\* rtrim(char\* str) {

if (!str) {

return '\0';

}

if (!\*str) {

return str;

}

char\* end = str + strlen(str) - 1;

while (end >= str && isspace(\*end)) {

end--;

}

\*(end + 1) = '\0';

return str;

}

char\*\* split\_string(char\* str) {

char\*\* splits = NULL;

char\* token = strtok(str, " ");

int spaces = 0;

while (token) {

splits = realloc(splits, sizeof(char\*) \* ++spaces);

if (!splits) {

return splits;

}

splits[spaces - 1] = token;

token = strtok(NULL, " ");

}

return splits;

}

int parse\_int(char\* str) {

char\* endptr;

int value = strtol(str, &endptr, 10);

if (endptr == str || \*endptr != '\0') {

exit(EXIT\_FAILURE);

}

return value;

}