1.
Fibonacci of n (1 <= n <= 10**18)
As the answer can be very big, you have to print answer % 1000000007

Input:
First line contains integer n
Output:
Single line containing the answer

Example:
Input: 10
Output: 55

2.
You are in the business of buying and selling shares. You are given the price of a share for N consecutive days. You can buy and sell a share on any day at the price for that day. Your only constraints are that you can not engage in multiple transactions at the same time(ie, you must sell the share before you buy again) and also you can not buy and sell on the same day.

Print the maximum profit you can achieve in these N days.

Input format
First line contains N.
2nd line contains N elements, Ai which are the prices of stocks for those N days.

Output format
A single number which is the maximum profit.

Input
7
100 180 260 310 40 535 695

Output
865(buy on day0, sell on 3. Buy again on day4 and sell on last day ie 310-100 + 695-40)

Constraints
2 <= N <= 10^6
0 < Ai <= 10^3

3.
Given n (n<=10) pairs of parentheses, write a function to generate all combinations of well-formed parentheses of length 2*n.
For example, given n = 3, a solution set is:
"((()))", "(()())", "(())()", "()(())", "()()()"
Make sure the returned list of strings are sorted.
Input
3
Output
((()))
(()())
(())()
()(())
()()()

4.
A sequence of *integers* is called good if each number divides (without a remainder) by the next number in the sequence.
Given *n* and *k* find the number of good sequences of length *k such that each number in sequence is between 1 & n*.
As the answer can be rather large print it modulo ($10^9 + 7$).
Input
The first line of input contains two space-separated integers *n*, *k* ($1 \le n, k \le 1000$).
Output
Output a single integer — the number of good sequences of length *k* modulo 1000000007 ($10^9 + 7$).
Examples
Input
3 2
Output
5
([1, 1], [2, 2], [3, 3], [1, 2], [1, 3])

Input
6 4
Output
39

5.
You are given a 2D matrix of nxm . Each entry of the matrix is initially a '.'
You will be given queries of 2 types .
1) Add x y > make the cell (x, y) a '#'
2) Find x y > find the nearest '#' from (x, y) (if it does not exist print -1)
Distance between two cells (x, y) & (a, b) is given by :
abs(x - a)+abs(y - b)
For each query of type2 print this minimum distance or else -1

Constraints:
N,M <= 500
Q <= 100000
Query of type 1 <= 50

Example
4 4
5
Find 1 1
Add 4 4
Find 3 3
Add 2 2
Find 3 3

Output:
-1
2
2