09. Recursion 1

Recursion is designed for a method calling itself.

Base case
recursion
stack of activation records
Comparison of recursion and loop

Exercise 1: Product of digits

Exercise 2: Tournament matches

Exercise 3: Adding parentheses

Exercise 4: Group sum

Exercise 5: Calculator

Exercise 6: Just stalling (Jan 2021)

Product of digits

Take a positive integer and recursively report the product of non-zero digits until the answer is less than 10. For examples,

1379 → 189 → 72 → 14 → 4
999999 → 531441 → 240 → 8

Tournament matches

You are given an integer N, the number of teams in a tournament that has strange rules:

- If the current number of teams is even, each team gets paired with another team. A total of $N/2$ matches are played, and $N/2$ teams advance to the next round.
- If the current number of teams is odd, one team randomly advances in the tournament, and the rest gets paired. A total of $(N-1)/2$ matches are played, and $(N-1)/2+1$ teams advance to the next round.

Return the number of matches played in the tournament until a winner is decided.

Input (from terminal / stdin)
- The only line contains integer N, $1 \leq N \leq 1e9$.

Output (to terminal / stdout)
- Report the number of matches.

Sample input
7

Sample output
6

1st Round: Teams = 7, Matches = 3, and 4 teams advance.
2nd Round: Teams = 4, Matches = 2, and 2 teams advance.
3rd Round: Teams = 2, Matches = 1, and 1 team is declared the winner.
Total number of matches = 3 + 2 + 1 = 6.

Adding parentheses

Given a string S of numbers and operators, return all possible results from computing all the different possible ways of grouping numbers and operators.

The string S consists of digits and the operator '+', '-', and '*'. The first character is a digit. The number of operators of S is at most 10. All the integer values in S are in the range [0, 20].

Input (from terminal / stdin)

- The only line contains the expression.

Output (to terminal / stdout)

- Report all the possible distinct integers values achievable by adding parentheses. Display the values in ascending order.

Sample input 1
2-1-1

Sample output 1
0 2

Sample input 2
2*3-4*5

Sample output 2
-34 -14 -10 10

```
(2*(3-(4*5))) = -34
((2*3)-(4*5)) = -14
((2*(3-4))*5) = -10
(2*((3-4)*5)) = -10
(((2*3)-4)*5) = 10
```

Group sum

Given an array A of N non-negative integers, report whether it is possible to pick some integers from A so that the sum is equal to the given target value T.

Input (from terminal / stdin)

- The first line contains Q, the number of queries, $1 \leq Q \leq 100$.
- For each query, the first line contains integers N and T, $2 \leq N \leq 20$, and $0 \leq T \leq 1000$.
- The next line contains the N integers, all in the range [0, 100].

Output (from terminal / stdout)

- For each query, report "YES" or "NO" on a separate line.

Sample input
```
3
3 10
2 4 8
3 14
2 4 8
3 9
2 4 8
```

Sample output
```
YES
YES
NO
```

Calculator

Given a string S representing a valid expression, implement a basic calculator to evaluate it, and return the result of the evaluation.

Note: Don't use any built-in function which evaluates strings as mathematical expressions, such as eval().

Input (from terminal / stdin)

- The only line contains the string S. The length of S is at most 1000.
- S consists of digits 0-9, '+', '-', '(', and ')'.
- The operator '+' is binary, i.e., there will always exists a value before and after each '+'.
- '-' could be used as a unary operation (i.e., "-1" and "-(2 + 3)" is valid).
- There will be no two consecutive operators in the input.
- All integers appearing in S are in the range [0, 1000].

Output (to terminal / stdout)

- Report the result of the evaluation.

Sample input

```
-1-(2+3-20)
```

Sample output

```
14
```

Just stalling (Jan 2021)

Farmer John has N cows $(1 \leq N \leq 20)$ of heights $a_1, \ldots, a_N$. His barn has N stalls with max height limits $b_1, \ldots\ b_N$ (so for example, if $b_5 = 17$, then a cow of height at most 17 can reside in stall 5). In how many distinct ways can Farmer John arrange his cows so that each cow is in a different stall, and so that the height limit is satisfied for every stall?

INPUT FORMAT (input arrives from the terminal / stdin):

- The first line contains N.
- The second line contains N space-separated integers $a_1, a_2, \ldots, a_N$.
- The third line contains N space-separated integers $b_1, b_2, \ldots, b_N$. All heights and limits are in the range $[1, 10^9]$.

OUTPUT FORMAT (print output to the terminal / stdout):

- The number of ways Farmer John can place each cow into a different stall such that the height limit is satisfied for every stall. Note that the large size of the output might require the use of a 64-bit integer, like a "long long" in C++.

SAMPLE INPUT:

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

SAMPLE OUTPUT:

```
8
```

In this example, we cannot place the third cow into the first stall since $3 = a_3 > b_1 = 2$. Similarly, we cannot place the fourth cow into the first or third stalls. One way to satisfy the height limits is to assign cow 1 to stall 1, cow 2 to stall 2, cow 3 to stall 3, and cow 4 to stall 4.

SCORING:
Test cases 1-5 satisfy $N \leq 8$
Test cases 6-12 satisfy no additional constraints.