2023 NORTHERN INDIANA HIGH SCHOOL CODING COMPETITION

Round One

COMPUTER SCIENCE AND INFORMATICS
COLLEGE OF LIBERAL ARTS AND SCIENCES
INDIANA UNIVERSITY SOUTH BEND

Problem 1. **String Mining**

Given an alphanumeric string made up of digits (0-9) and English letters (a/A-z/Z), find the sum of all the digit characters in the string.

The input is a string of length in the range of [1, 100]. The output is a number, which is the sum of all the digit characters in that string.

Sample input and output:

Input: 0
Output: 0

Input: aByMR
Output: 0

Input: 1234
Output: 10

Input: Hello2there3friend
Output: 5

Input: CS123welcome45IUSB
Output: 15

Problem 2. Round to Even

The round-to-even (**rte**) method is used in engineering and finance to reduce bias when you use rounded numbers to estimate sums and averages. The round-to-even method works like this:

- If the difference between the number and the nearest integer is less than 0.5, round to the nearest integer. For example **rte**(1.48)→1, **rte**(1.501)→2, and **rte**(3.0)→3.
- If the difference between the number and the nearest integer is exactly 0.5, look at the integer part of the number. If the integer part is even, round towards the integer. If the integer part of the number is odd, round away from the integer. In either case, the rounded number is an even integer. Examples are given in the following table, where 0.5 and 2.5 both have even integer parts and the rounded numbers are 0 and 2, respectively; while 1.5 and 3.5 both have odd integer parts and the rounded numbers are 2 and 4, respectively.

	Traditional Round	Round to Even
0.00	0	0
0.25	0	0
0.50	1	0
0.75	1	1
1.00	1	1
1.25	1	1
1.50	2	2

	Traditional Round	Round to Even
2.00	2	2
2.25	2	2
2.50	3	2
2.75	3	3
3.00	3	3
3.25	3	3
3.50	4	4

Write a program that accepts a non-negative number as input and outputs the result of round to even operation.

Sample input and output:

Input: 0
Output: 0

Input: 2.83
Output: 3

Input: 3.27
Output: 3

Input: 2.5
Output: 2

Input: 11.5
Output: 12

Problem 3. Gold Prices

Of all the precious metals, gold is the most popular as an investment. In the past 100 years, gold prices were fluctuating as shown below, but with an increasing trend in general.

In this program, you are given some years' gold prices and asked to identify the two years with the largest annual price change. The Annual Price Change (APC) is defined as the following:

APC = (P2 – P1) / P1

Where P1 and P2 are two consecutive end-year gold prices.

The input to the program has two lines. The first line has two numbers: **y n** separated by a space where **y** is a four-digit number representing the starting year and **n** is a whole number greater than or equal to 2 representing the number of years, considered consecutively. The second line contains the gold price data of these **n** years, with decimal numbers separated by spaces. The output line consists of the two years with the largest APC. Assume that no multiple years have the same largest APC.

Sample input and output:

input: 1969 10
input: 35.21 37.38 43.50 64.70 112.25 187.50 140.25 134.55 165.60 224.50
output: 1972 - 1973

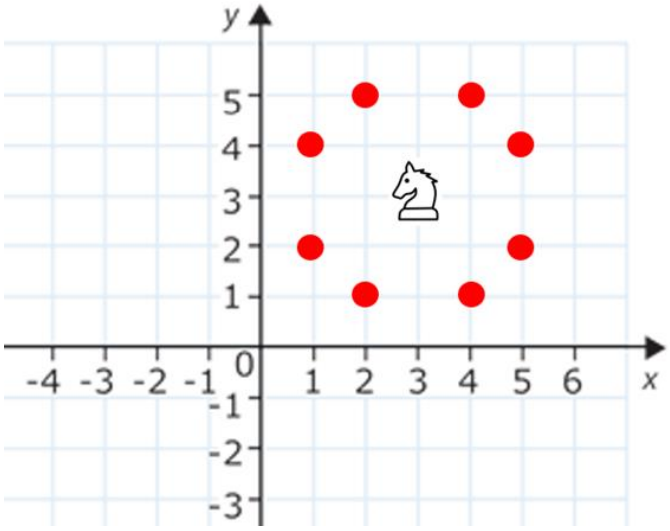
input: 1980 7
input: 589.50 400.00 448.00 381.50 309.00 327.00 390.90
output: 1985 – 1986

input: 2017 6
input: 1296.50 1281.65 1523.00 1895.10 1828.60 1824.32
output: 2019 – 2020

Problem 4. Knight Movement

In chess, knights move in an "L shape" at any step—that is, they can move two squares in any direction vertically followed by one square horizontally, or two squares in any direction horizontally followed by one square vertically.

Suppose we have an extended version of chessboard with unlimited two dimensions. Both the current position and the target position of a knight are represented by a pair of (x, y) coordinates, where both x and y are integers (could be negative). In the following example, a knight is at position (3, 3) and in one step, it can move to any of the eight positions (red dots).



Write a program that accepts inputs of four integers: x_1 y_1 x_2 y_2 separated by white spaces, where (x_1, y_1) represents the knight's current position and (x_2, y_2) represents its target position. The output is the minimum steps the knight needs to move from its current position to the target position. If the minimum steps are greater than 2 you just display "**>2**".

Sample input and output:

Input: 0 0 0 0
Output: 0 # 0 step

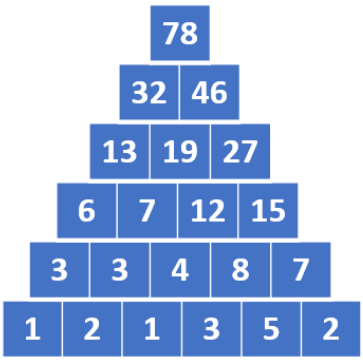
Input: 3 3 1 4
Output: 1 # 1 step

Input: 3 3 0 0
Output: 2 #2 steps: (3, 3) → (1, 2) → (0, 0)

Input: 3 3 2 3
Output: >2 # more than 2 steps: (3, 3) → (2, 1) → (4, 2) →(2, 3)

Problem 5. Summation Steps

Summation Steps can be formed by starting with a set of numbers placed at the first level (bottom). From there, at each level above it, the number of elements is one less than the previous level and each element is the sum of the two elements below it in the previous level. An example is given below. For example, the number 8 on the second level from the bottom is the sum of 3 and 5 from the bottom level.



In this program, you are given a list of numbers (separated by white spaces) making up the bottom level. You are asked to print the entire Summation Steps pyramid. At each level of the output, the numbers should be separated by a comma and a space.

The input has two lines: the first line is the bottom list size (greater than 0), and the second line is the bottom list itself. The output is the entire summation steps.

Sample input and output:

Input:
```
4
1 2 1 1
```

Output:
```
11
6, 5
3, 3, 2
1, 2, 1, 1
```

Input:
```
5
1 2 3 4 5
```

Output:
```
48
20, 28
8, 12, 16
3, 5, 7, 9
1, 2, 3, 4, 5
```

Problem 6. Alphabetical Order

A kid is given ten different English letters and asked to arrange them in alphabetical order. Sometimes he gets the order right and sometimes he gets it wrong. If the order is incorrect, you are asked to remove the minimum number of letters from the list to make the rest of the letters in alphabetical order.

Write a program to perform this task. The input line contains 10 different letters all in upper cases NOT separated by spaces. The output line contains the minimum number of letters that should be removed to make the rest of the letters in alphabetical order.

Sample input and output:

Input: ABCEFGDLMN
Output: 1

Explanation: D should be removed.

Input: ABCEFGMNOZ
Output: 0

Explanation: The letters are in order.

Input: ABCEFGLXND
Output: 2

Explanation: X and D should be removed.