SIENA COMPUTER SCIENCE

38th Annual High School Programming Contest

sponsored by transfinder

April 17, 2026

Gold Problem #1: The Saffir-Simpson Scale

Background Information: From Earth Science class, you might remember that the Saffir-Simpson scale is a measurement for the intensity of major storms and hurricanes. These weather phenomena are classified using the following chart.

<table>
<tr><th colspan="6">NHC Classification & Saffir-Simpson Hurricane Scale (SSHS)</th></tr>
<tr><th colspan="2">Cat.</th><th>Pressure at center</th><th>Wind (km/h)</th><th>Surge (m)</th><th>Damage</th></tr>
<tr><td colspan="2">D</td><td rowspan="3">above 980 hPa</td><td>below 62</td><td rowspan="2">below 1.2m</td><td>None</td></tr>
<tr><td colspan="2">S</td><td>62 to 117</td><td>Light</td></tr>
<tr><td rowspan="2">H</td><td>1</td><td>118 to 153</td><td>[1.2m, 1.8m)</td><td>Minimal</td></tr>
<tr><td>2</td><td>966 hPa to 980 hPa</td><td>154 to 177</td><td>[1.8m, 2.7m)</td><td>Moderate</td></tr>
<tr><td rowspan="3">M</td><td>3</td><td>946 hPa to 965 hPa</td><td>178 to 209</td><td>[2.7m, 4.0m)</td><td>Extensive</td></tr>
<tr><td>4</td><td>920 hPa to 945 hPa</td><td>210 to 249</td><td>[4.0m, 5.5m]</td><td>Extreme</td></tr>
<tr><td>5</td><td>below 920 hPa</td><td>above 249</td><td>above 5.5m</td><td>Catastrophic</td></tr>
</table>

The National Hurricane Center considers anything below category 1 to be either a *tropical depression* (D) or a *tropical storm* (S). Categories 1 and 2 are *hurricanes* (H) above the Beaufort *force 12* threshold. Categories 3, 4 and 5 are *major hurricanes* (M). There's no need for a category 6.

Programming Problem:

Your program will read in a string representing the unit and a non-negative real or integer. The units will be either "**hPa**", "**km/h**" or "**m**", as shown above in the table. The number could either be a non-negative integer (for pressure or wind) or real number (for surge), depending on the unit. Your program will output either "**Tropical Depression**", "**Tropical Storm**", "**Category-1 Hurricane**", "**Category-2 Hurricane**", "**Category-3 Hurricane**", "**Category-4 Hurricane**", or "**Category-5 Hurricane**". These are the only possible outputs, and they are listed in order of increasing severity. If an input, such as **1.0 m** is given which can apply to more than one category, output the more severe classification.

Input: A string unit and a number, on separate lines.
Output: The corresponding classification

Example 1: Input:
```
km/h
180
```
Output: `Category-3 Hurricane`

Example 2: Input:
```
hPa
1005
```
Output: `Category-1 Hurricane`

Example 3: Input:
```
m
0.8
```
Output: `Tropical Storm`

Example 4: Input:
```
km/h
18
```
Output: `Tropical Depression`

Gold Problem #2: Squareish Numbers

Background Information: A squareish number is an integer that is divisible by a perfect square integer > 1. For example, 12 is a squareish number, since 12 is divisible by 4. It is easy to show that there are infinite squareish numbers.
A K-squarerish sequence is a sequence of consecutive integers where each number in the sequence is a squareish number and the numbers immediately before and after the numbers in the sequence are not squareish numbers.

This problem will ask you to find an instance of K consecutive squarish integers, starting at or above a given starting value S.

Programming Problem:
Write a program that will find the first K-squareish sequence that starts at or after S.

Input: Positive Integers S <= 1000 and K <= 8 on separate input lines

Output: The smallest integer N that is the first number in the first K-squareish sequence where N , such that N, N+1, …, N+(K-1) are all squareish numbers.

Example 1: Input:
```
1
1
```
Output:
```
4
```

Example 2: Input:
```
1
2
```
Output:
```
8
```

Example 3: Input:
```
5
4
```
Output:
```
242
```

Example 4: Input:
```
10
2
```
Output:
```
24
```

Gold Problem #3: 2026 is Not Prime and 2026 has an Iterative Collatz Value of 112

Background Information: Eighty-nine years ago, in 1937, the German mathematician, Lothar Collatz, stated what sounded like a simple question but turned out to be very difficult. His question became known as the Collatz Conjecture. It is sometimes called the 3N + 1 problem. Collatz said that if you start with any positive integer N, then repeatedly apply the following rules- divide N by 2 if it is even, or replace N by 3N + 1 if it is odd - and continue doing this then you will eventually get to 1. Many great mathematicians have worked on this problem and none have been able to prove the conjecture or find a counter example.

Example: if you start with N = 7 then

7 → 22 → 11 → 34 → 17 → 52 → 26 → 13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1 .

Note that it takes 16 steps for 7 to reach 1. We say that 7 has an Iterative Collatz Value or ICV of 16.

The integer 2026 has an ICV of 112.

For values less than a billion, the one with the largest ICV is 670,617,279 which has an ICV of 986.

Programming Problem:

Input: A positive integer N ≤ 100,000.

Output: The number N followed by a space followed by the ICV(N).

Example 1:	Input:	`2026`
	Output:	`2026 112`
Example 2:	Input:	`7`
	Output:	`7 16`

Gold Problem #4: An AI Bot's Look And Say

Background Information: Starting with a seed number *s*, a Look-and-Say sequence LS(s) is a sequence of integers generated by an AI bot from a starting seed number. The AI bot produces the sequence by recording in text exactly what it states using its voice-enabled functionality when reading off the digits. The AI bot is talented and avoids repetition. For example, when it produces part of a sequence when the seed number has repeating digits the bot just says the number of those digits. For example when the bot sees 333333 it will say "six threes" which then becomes 63 in the sequence.

Here are three easy examples:

1 has "one 1" in it, so LS(1) = 11.
11 has "two 1s" in it, so LS(11) = 21.
21 has "one 2 and one 1" in it, so LS(21) = 1211.

Write a program that will input the seed value *s,* and a non-negative integer *n*. Your output will be the result of applying the Look-and-Say function LS(x) *n* times, first on *s*, and each subsequent time on the result of the previous function application.

For example, if you input 1 for *s* and 4 for *n* your program will compute:

LS(1) = 11 and then LS(11) = 21 and then LS(21) = 1211 and then LS(1211) = 111221

Another example, if you input 37 for *s* and 5 for *n* your program will compute:
LS(37) = 1317 and then LS(1317) = 11131117 and then LS(11131117) = 31133117 and then LS(31133117) = 1321232117 and finally LS(1321232117) = 111312111213122117

Programming Problem:

Input: The seed number *s* and the number of function applications *n*, on separate lines.
Output: The result of applying the function *n* times, starting on *s*.

Example 1: Input:
```
1
4
```
Output:
```
111221
```

Example 2: Input:
```
22
10
```
Output:
```
22
```

Example 3: Input:
```
5
5
```
Output:
```
1113122115
```

Example 4: Input:
```
3
2
```
Output:
```
1113
```

Example 5: Input:
```
37
5
```
Output:
```
111312111213122117
```

Gold Problem #5: Sock It To Me! Sock It To Me! Sock It To Me!

Background Information: You are setting out socks from your sock drawer in sequences of length n >= 3. The socks are either black or white. Since you are familiar with mathematics and especially the binary representation of positive integers, you know there are 2^n different sequences of length n that can be made with black and white socks. How many of those sequences do not have three consecutive white socks?

Your program must find the number of sequences made from all sequences of input n number of socks that do not have three white socks in a row.

Programming Problem:

Input: positive integer $3 \leq n \leq 70$
Output: The number of possible sequences of socks of length n, that do not have three consecutive white socks.

Example 1: Input:
```
3
```
Output:
```
7
```

Example 2: Input:
```
4
```
Output:
```
13
```

Example 3: Input:
```
6
```
Output:
```
44
```

Example 4: Input:
```
60
```
Output:
```
1383410902447554
```

Gold Problem #6: Best Rectangle

Background Information: You are given an *n* by *m* grid of integer numbers. Your program is to find the subrectangle within the grid that has the largest sum. For example, the grid

0	-2	-7	0
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

has as its largest subrectangle

9	2
-4	1
-1	8

with a sum of 15.

Your program will be given the dimensions of an *n* x *m* grid and then *n* rows of *m* numbers each. You will then print out the sum of the values of the largest subrectangle.

Programming Problem:

Input: Grid dimensions *n* and *m* on one line, separated by a space, and each a positive integer less than or equal to 75, followed by *n* lines of *m* numbers each, separated by spaces, which are the values in the rows of the grid. These values are integers in the range [-9,9].
Output: The sum of the largest subrectangle.

Example 1: Input:
```
4 4
0 -2 -7 0
9 2 -6 2
-4 1 -4 1
-1 8 0 -2
```

Output:
```
15
```

Example 2: Input:
```
2 3
1 2 3
1 2 3
```

Output:
```
12
```

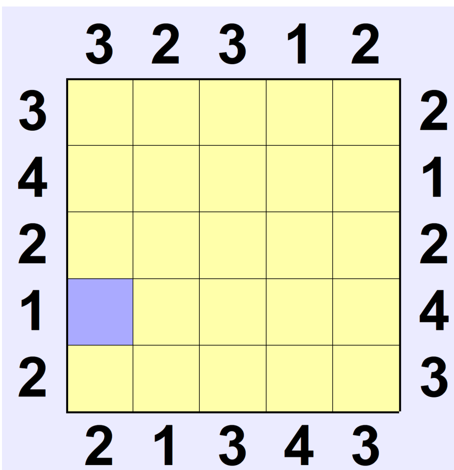
38th Annual High School Programming Contest

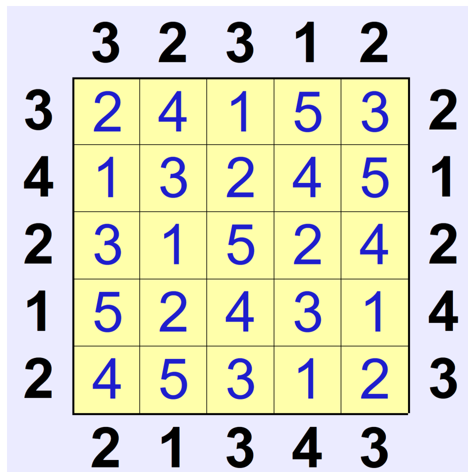
sponsored by transfinder

April 17, 2026

Gold Problem #7: Skyscrapers

Background Information: The skyscrapers problem is a logic problem based upon a 5x5 grid of integers. Each row and column in the completed puzzle will contain one instance of each of the numbers 1, 2, 3, 4, and 5. Below are the puzzle starting configuration, and its completed solution.





Each integer of the grid that you place represents the height of a skyscraper. The bolded numbers along the sides represent the number of skyscraper peaks that you can see when you look into the grid from the number. For example, for the top row, there is a 3 on the left, and a 2 on the right. The 3 means that you can see three skyscraper peaks (in this case the 2, 4, and 5) gazing into the grid row from the left. But the 2 means that you can see two skyscraper peaks (in this case, the 3 and the 5) gazing into the grid row from the right. You cannot see a smaller skyscraper behind a taller one, obviously because it would be blocked from sight.

So, given the 25 bolded numbers indicating the number of skyscrapers seen from each row/column orientation, your program must find the heights of all skyscrapers in the grid.

Programming Problem:

Input: Four rows of 5 numbers, each on their own line, representing the top, bottom, left, and right bolded numbers.

Output: 5 lines of 5 numbers representing the grid solution

Example 1:

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

Output:
```
2 4 1 5 3
1 3 2 4 5
3 1 5 2 4
5 2 4 3 1
4 5 3 1 2
```