TigerHacks

November 15th 2025

Henry Santangelo, Charlie Moloney, Zachry Bostock
Oliver Servedio, Katie Benincasa, Asher Rosen
Julian Toub, Mary Chickering, Chris Suy, Irhan Iftikar

Welcome To The Farm!

Question 1 : Farmer Hall’s Market

Quirky Farmer Hall goes to the farmer’s market only for days that end with “th”. Calendar days are written as January 1st, January 2nd, etc.. Your job is to convert from a cardinal number representation of the date to a boolean if Farmer Hall will be at the market.

Rules for Date Suffixes

When converting a number to its ordinal form (like 1st, 2nd, 3rd, 4th), follow these rules:

- If a number ends in 1:
  - For 1, 21, or 31: append “st” (Example: 1st, 21st, 31st)
  - For 11: append “th” (Example: 11th)
- If a number ends in 2:
  - For 2 or 22: append “nd” (Example: 2nd, 22nd)
  - For 12: append “th” (Example: 12th)
- If a number ends in 3:
  - For 3 or 23: append “rd” (Example: 3rd, 23rd)
  - For 13: append “th” (Example: 13th)
- If a number ends in 4, 5, 6, 7, 8, 9, or 0:
  - Always append “th” (Example: 4th, 5th, 6th, 7th, 8th, 9th, 10th)

Input Format

- Your program will receive two positive integers on two lines
- First line: month (1-12)
- Second line: day (1-31, depending on the month)
- Numbers will always represent valid calendar dates

Output Format

- Print “Yes” if Farmer Hall will be at the market (if the day ends in “th”)
- Print “No” if Farmer Hall will not be at the market (if the day ends in “st”, “nd”, or “rd”)

Examples

- Input : Two positive integers on two lines (month on the first line, day on the second line).
- Output : “Yes” or “No”, depending on whether Farmer Hall will be at the farmer’s market.

Example 1

- Input:

```
1
2
```

- Output : `No`
- Explanation: January 2nd ends with “nd”, not “th”, so Farmer Hall won’t be there

Example 2

- Input:

```
10
8
```

- Output : `Yes`
- Explanation: October 8th ends with “th”, so Farmer Hall will be there

Example 3

- Input:

```
4
7
```

- Output : `Yes`
- Explanation: April 7th ends with “th”, so Farmer Hall will be there

Example 4

- Input:

```
3
11
```

- Output : `Yes`
- Explanation: March 11th is a special case that ends with “th” (not “st”), so Farmer Hall will be there

Example 5

- Input:

```
12
23
```

- Output : `No`

- Explanation: December 23rd ends with “rd”, not “th”, so Farmer Hall won’t be there

Question 2: Farmer Moloney’s Eggs

Eccentric Farmer Moloney has a peculiar rule at his egg stand - he refuses to sell eggs to anyone whose name is a palindrome (reads the same forwards and backwards). A name is considered a palindrome if it reads exactly the same forwards and backwards when considering only letters (ignoring case, spaces, and any other characters).

For example: - “Hannah” is a palindrome because “hannah” reads the same forwards and backwards - “John Doe” is not a palindrome - “A man, a plan, a canal: Panama” would be a palindrome because “amanaplanacanalpanama” reads the same forwards and backwards

Your job is to write a program that determines whether Farmer Moloney will sell eggs to a customer based on their name.

Input

- Input consists of a single line containing the customer’s name (may include spaces and punctuation)
- Names will not be empty and will be at most 100 characters long
- Names may contain letters, spaces, and punctuation marks

Output

- Output “No eggs for you!” if the name is a palindrome (Farmer Moloney won’t sell)
- Output “Here are your eggs!” if the name is not a palindrome (Farmer Moloney will sell)

Examples

Example 1

- Input: `Bob`
- Output: `No eggs for you!`
- Explanation: “bob” is the same forwards and backwards

Example 2

- Input: `James Smith`
- Output: `Here are your eggs!`
- Explanation: “jamessmith” is not the same forwards and backwards

Example 3

- Input: `A Santa at NASA`
- Output: `No eggs for you!`

- Explanation: “asantaatnasa” is the same forwards and backwards

Example 4

- Input: `Mary-Jane!`
- Output: `Here are your eggs!`
- Explanation: “maryjane” is not the same forwards and backwards

Question 3: Farmer Iversen's Magic Seeds

Farmer Iversen keeps her magic seeds in small jars, and she has a strange rule for when she'll plant them.

She will only plant seeds on days where: - The sum of the month and day numbers (not the digits) is a prime number - The month name (like "January", "February", etc.) has an odd number of letters

Your job is to decide whether Farmer Iversen will plant her seeds on a given date.

Input Format

- Your program will receive two positive integers on a single line
- First number represents the month (1-12)
- Second number represents the day (1-31, depending on the month)
- Numbers will always represent valid calendar dates

Output Format

- Print "Planting day!" if both rules are met
- Print "No planting today." otherwise

Examples

Example 1

- Input: `3 24`
- Output: `No planting today.`
- Explanation: March (5 letters, odd) and 3+24=27 (not prime)

Example 2

- Input: `3 5`
- Output: `No planting today.`
- Explanation: March (5 letters, odd) and 3+5=8 (not prime)

Example 3

- Input: `5 2`
- Output: `Planting day!`
- Explanation: May (3 letters, odd) and 5+2=7 (prime)

Example 4

- Input: `5 13`

- Output: `No planting today.`
- Explanation: May (3 letters, odd) and 5+13=18 (not prime)

Question 4: Farmer Benincasa’s Wheat Tax

Farmer Benincasa farms wheat and is subject to a marginal tax system based on her annual yield. The tax brackets are always the same, and only the total wheat yield for the year is provided as input.

Your job is to help Farmer Benincasa calculate the total amount of tax she owes for this year’s wheat yield, rounded to the nearest dollar.

Tax Brackets (constant for all problems)

- 0% for the first 1000 units
- 10% for the next 4000 units (1001 to 5000)
- 15% for the next 5000 units (5001 to 10000)
- 20% for any units above 10000

Input Format

- A single integer `y`, the total wheat yield for the year (0 ≤ y ≤ 1,000,000).

Output Format

- Print a single integer: the total tax Farmer Benincasa owes, rounded to the nearest dollar.

Rules

- For each bracket, only the portion of the yield within that bracket is taxed at the bracket’s rate.
- The first bracket always starts at 0 units.
- Tax is calculated for each bracket as: (amount in bracket) × (rate/100).
- Round the final total tax to the nearest dollar.

Examples

Example 1

- Input:

```
4200
```

- Output: `320`

- Explanation:

  - First 1000 units: 0% tax → $0
  - Next 3200 units (1001 to 4200): 10% tax → $320
  - No units in higher brackets

Example 2

- Input:

  ```
  6000
  ```

- Output: `550`

- Explanation:

  - First 1000 units: 0% tax → $0
  - Next 4000 units (1001 to 5000): 10% tax → $400
  - Next 1000 units (5001 to 6000): 15% tax → $150
  - No units in higher brackets

Example 3

- Input:

  ```
  12000
  ```

- Output: `1550`

- Explanation:

  - First 1000 units: 0% tax → $0
  - Next 4000 units: 10% tax → $400
  - Next 5000 units: 15% tax → $750
  - Next 2000 units: 20% tax → $400

Question 5: Farmer Bostock's Magic Tablet

Farmer Bostock owns a mysterious tablet that predicts the start of the rainy season. The tablet displays a Boolean expression using the following operators: NOT, AND, OR, XOR, NAND, and parentheses for grouping.

Each variable in the expression is either True or False. The tablet will declare that the rainy season will start tomorrow if the expression evaluates to True.

Your job is to help Farmer Bostock determine if the rainy season will start, by evaluating the Boolean expression.

Input Format

- The first line contains a Boolean expression using variables (single uppercase letters), parentheses, and the operators: NOT, AND, OR, XOR, NAND.
- The second line contains space-separated assignments for each variable, in the format `A=True B=False ...`.
- Operators and variables are case-sensitive. Parentheses may be nested.
- There will be no syntax errors in the input.

Output Format

- Print `Rainy season starts!` if the expression evaluates to True.
- Print `No rain tomorrow.` if the expression evaluates to False.

Operators

- NOT X: logical negation
- X AND Y: logical and
- X OR Y: logical or
- X XOR Y: logical exclusive or
- X NAND Y: logical nand (not and)

Examples

Example 1

- Input:

```
 (A AND B) OR (NOT C)
A=True B=False C=False
```

- Output: `Rainy season starts!`
- Explanation: (True AND False) OR (NOT False) → False OR True → True

Example 2

- Input:

```
 NOT (A NAND B)
A=True B=True
```

- Output: `Rainy season starts!`
- Explanation: NOT (NOT (True AND True)) → NOT (NOT True) → True

Example 3

- Input:

```
 (A XOR B) AND (C OR D)
A=True B=True C=False D=True
```

- Output: `No rain tomorrow.`
- Explanation: (True XOR True) AND (False OR True) → False AND True → False

---

Question 6: Farmer Santangelo’s Pasture

Farmer Santangelo owns a large rectangular pasture divided into square plots. Each plot is either Grass (G), Water (W), or Rock (R).

He wants to know how many cattle his land can support. Each connected stretch of grass (connected up, down, left, or right, but not diagonally) can support as many cattle as there are grass plots in that stretch.

Unfortunately, the cattle can’t cross water or rocks — so each separate area of connected grass counts as its own grazing zone.

Your job is to help Farmer Santangelo find the largest grazing zone — that is, the biggest connected group of grass plots on the pasture.

Example Grid Format

Suppose the input grid line is:

```
GGWWRRRG;GWWWGRRG;GGGGGWRG;WRWWGWRG;WWWWGGGG;RRGGGGGG
```

This represents the following 2D array:

```
 GWWWGRRG
GGGGGWRG
WRWWGWRG
WWWWGGGG
RRGGGGGG
```

Input Format

- Two lines:
  - First line: two positive integers separated by a space:
    - r — number of rows (1 ≤ r ≤ 100)
    - c — number of columns (1 ≤ c ≤ 100)
  - Second line: the grid, with each row of c characters separated by a semicolon (`;`).
    - Each character is one of:
      - G for grass
      - W for water
      - R for rock

Output Format

- Print a single integer — the size of the largest connected grazing zone (the maximum number of connected G plots).

Rules

- Two grass plots are connected if they share a side (north, south, east, or west).
- Diagonal adjacency does not count.
- If there is no grass at all, output 0.

Examples

Example 1

- Input:

```
 6 8
GGWWRRRG;GWWWGRRG;GGGGGWRG;WRWWGWRG;WWWWGGGG;RRGGGGGG
```

- Output: `24`

- Explanation: The largest connected patch of grass contains 24 plots.

Example 2

- Input:

```
 5 5
WWWWW;WGGGW;WGWGW;WGGGW;WWWWW
```

- Output: `8`

- Explanation: The inner ring of grass (forming a square) has 8 connected plots.

Example 3

- Input:

```
 4 4
WWWW;WRRW;WRRW;WWWW
```

- Output: `0`

- Explanation: There is no grass at all, so the largest grazing zone supports 0 cattle.

Question 7: Farmer Campbell's Pig Latin

Extremely eccentric Farmer Campbell has discovered that his pigs respond best to Pig Latin. To communicate with them, he needs to convert his sentences into Pig Latin.

Your job is to help Farmer Campbell translate a sentence into its Pig Latin equivalent.

Pig Latin Rules

- For each word in the sentence:
  - If the word begins with a vowel (a, e, i, o, u), add "way" to the end of the word.
  - If the word begins with a consonant, move all the consonants before the first vowel to the end of the word, then add "ay".
  - Words are separated by spaces. Preserve the original word order.
  - Treat 'y' as a consonant at the start of a word.
  - Ignore punctuation (assume input contains only lowercase letters and spaces).

Input Format

- A single line containing a sentence (lowercase letters and spaces only).

Output Format

- Print the Pig Latin translation of the sentence, with words separated by spaces.

Examples

Example 1

- Input:

  ```
  i love pigs
  ```

- Output: `iway ovelay igspay`

- Explanation:

  - "i" → "iway"
  - "love" → "ovelay"
  - "pigs" → "igspay"

Example 2

- Input:

```
apples are tasty
```

- Output: `applesway areway astytay`

- Explanation:

    - “apples” → “applesway”
    - “are” → “areway”
    - “tasty” → “astytay”

Example 3

- Input:

```
oink oink
```

- Output: `oinkway oinkway`

- Explanation:

    - “oink” → “oinkway”

Question 8: Farmer Servedio's Corn Maze

Farmer Servedio is in a hurry—he left his flute at the end of the corn maze and wants to know the shortest path to retrieve it!

The corn maze is a rectangular grid made of walls and open passages. Farmer Servedio can move up, down, left, or right (but never diagonally) and cannot move through walls. Your job is to help Farmer Servedio find the shortest path from the start (S) to the exit (F), where his flute is waiting.

There may be more than one path in the maze to the finish line, there also may be dead end paths.

Input Format

- Two lines:
  - First line: two integers R and C, the number of rows and columns in the maze.
  - Second line: the maze grid, with each row of C characters separated by a semicolon (`;`).
    - `#` represents a wall
    - `.` represents an open space
    - `S` represents the starting point
    - `F` represents the exit (where the flute is)

Output Format

- Print a single integer: the minimum number of moves required to reach the exit.
- If there is no path, print `-1`.

Examples

Example 1

- Input:

```
5 7
#######;#S#...#;#.#.#.#;#...#.F;#######
```

- Output: `11`

- Explanation: Farmer Servedio must weave through open corridors to reach his flute in 11 moves. The maze looks like:

```
#######
#S#...#
#.#.#.#
#...#.F
#######
```

Example 2

- Input:

```
4 6
######;#S#..F;#...##;######
```

- Output: `6`
- Explanation: Farmer Servedio can reach his flute in 6 moves. The maze looks like:

```
######
#S#..F
#...##
######
```

Question 9: Farmer Davis’ and Farmer Carr’s Crazy Tractors

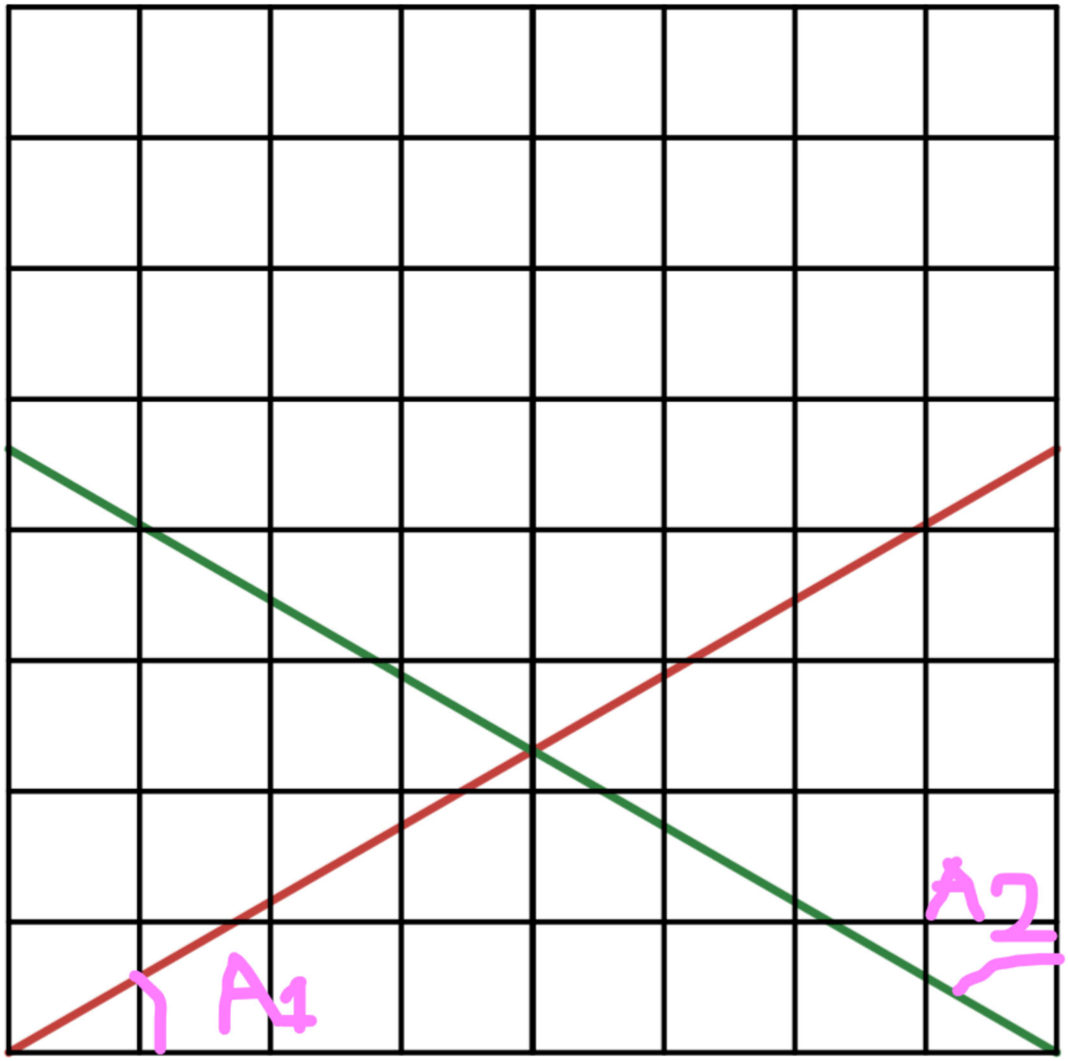
Farmer Davis and Farmer Carr are planting crops in a square field. Each tractor drives in a straight line across the field from its starting corner, and the farmers want to know which 1×1 plots are visited by each tractor and where both tractors visit the same plot.

Field and motion model: - The field is an 8×8 grid; each cell is 1 meter by 1 meter. - Farmer Davis starts at the bottom-left corner (coordinate (0, 0)). - Farmer Carr starts at the bottom-right corner (coordinate (8, 0)). - You are given two angles, A1 and A2 (in degrees), with 0 < A < 90. - A1 is measured from the positive x-axis (due east) toward the positive y-axis (due north). - A2 is measured from the right vertical edge (the positive y-axis at x = 8) toward the left; i.e., it is the angle between Carr’s path and the upward vertical direction on the right boundary. - Davis drives along the ray from (0, 0) at angle A1 until he exits the field. - Carr drives along the ray from (8, 0) at angle A2 (deviation from vertical), heading up and to the left, until he exits the field.

Visitation rule: - A cell is considered visited by a farmer if the farmer’s straight-line path segment intersects that cell. - Touching edges or corners counts as a visit. In particular, if the path passes exactly through a grid intersection (corner), all four adjacent cells are considered visited.

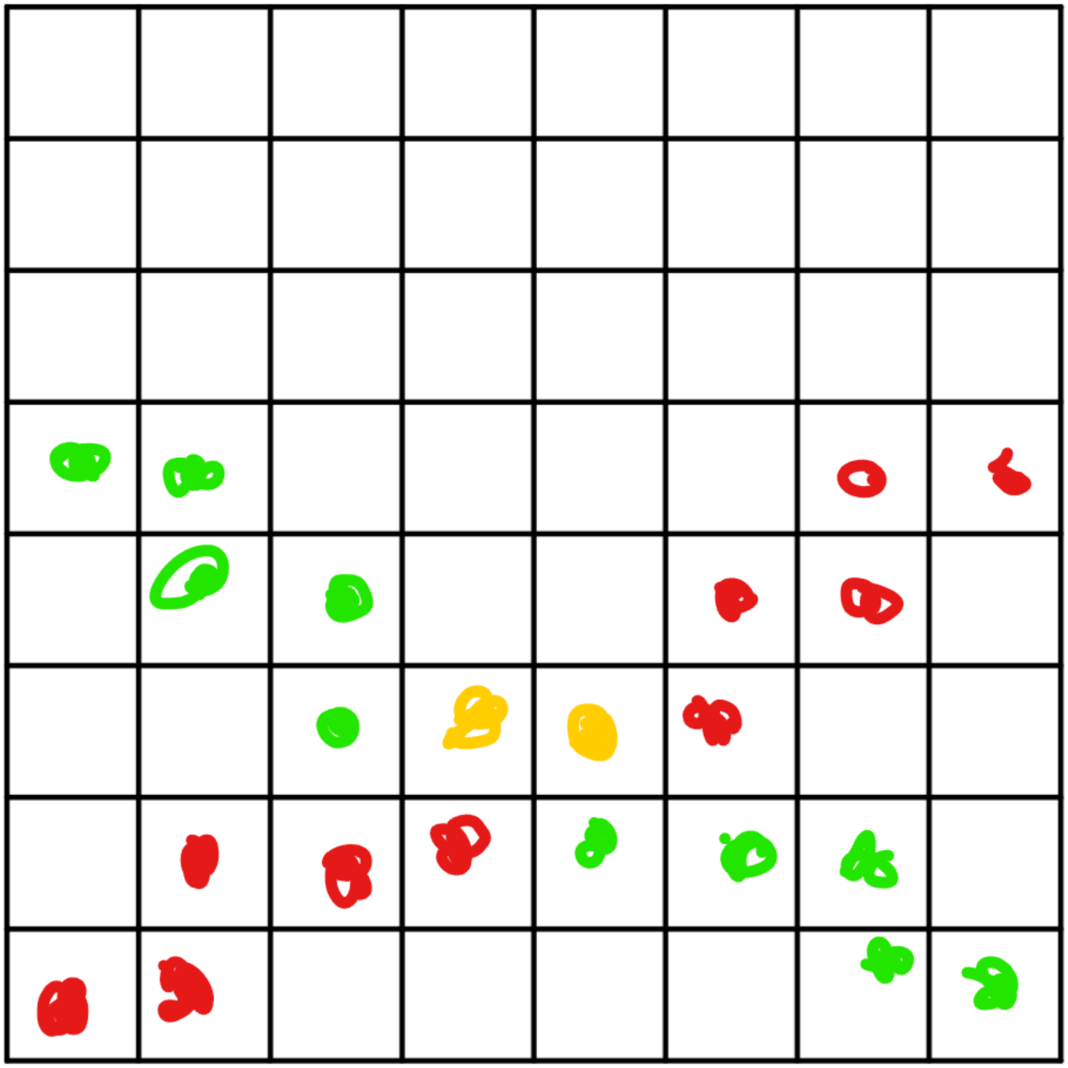
Legend: - Red dots (D): cells visited by Farmer Davis - Green dots (C): cells visited by Farmer Carr - Yellow dots (X): cells visited by both farmers

Illustration of the paths for example angles:



- A1 = 30 degrees
- A2 = 60 degrees

The traced paths look like:



Output encoding: - Use D for Davis, C for Carr, X for both, and . for empty. - Output 8 rows from top to bottom, joined by semicolons `;` .

Input Format

- One line containing two integers separated by a space: A1 and A2
- Both integers are between 0 and 90 (exclusive)

Output Format

- Print a single string representing the 8×8 grid, with rows separated by `;`

Examples

Example 1

- Input:

```
30 60
```

- Output: `........;........;........;CC....DD;.CC..DD.;..CXXD..;.DDDCCC.;DD....CC`

Example 2

- Input:

```
45 45
```

- Output: `CC....DD;CCC..DDD;.CCCDDD.;..CXXD..;..DXXC..;.DDDCCC.;DDD..CCC;DD....CC`
