## Fifteenth Annual
## University of Central Florida

# High School Programming Tournament: Online Edition

## *Problems – Division 2*

| Problem Name | Filename |
|---|---|
| Super Cheetah Speedway | cheetah |
| Clout Chasing | clout |
| DNA Sequences | dna |
| The Ducks of Danger Island | ducks |
| Optical Illusion | illusion |
| Mechanized Mob | mob |
| Palindromic Passwords | password |
| Pizza Party | pizza |
| The Royal Wedding | wedding |

Call your program file:
*filename*.cpp, *filename*.java, or *filename*.py

For example, if you are solving Super Cheetah Speedway,
Call your program file:
cheetah.cpp, cheetah.java, or cheetah.py
Call your Java class: cheetah

# Super Cheetah Speedway

*Filename:* `cheetah`

Construction has just finished on the Super Cheetah Speedway! To commemorate this milestone, we are holding a running race for all the local cheetahs. Racers, take your positions! BANG! And they're off!

Just look at those cheetahs flying by! As fun as it is to watch, we must also make sure to enforce certain safety measures on the racing circuit. For instance, if a cheetah is running too fast, he is at risk of getting tired out, or worse, tripping and getting hurt. As such, there is a maximum speed limit imposed on the racers. However, if a cheetah is running too slow, it could confuse the racers behind him. Therefore, there is also a minimum speed limit imposed on the racers, which is equal to half of the maximum speed limit.

To run faster than the maximum speed limit or slower than the minimum speed limit is a violation of the race guidelines. Can you help us determine which cheetahs are breaking the rules?

**The Problem:**

Given the maximum speed limit and a cheetah's speed, determine whether he is running too fast, too slow, or at an acceptable speed.

**The Input:**

The input consists of two space-separated integers, $m$ and $s$ ($0 < m, s \leq 1000$), representing the maximum speed limit and the cheetah's speed.

**The Output:**

Output a single line. If the cheetah is running too fast, output "he's cheetah-ing!" If the cheetah is running too slow, output "slowpoke spotted!" If the cheetah is running at an acceptable speed, output "born to run!"

| **Sample Input 1:** | **Sample Output 1:** |
|---|---|
| `100 101` | `he's cheetah-ing!` |

| **Sample Input 2:** | **Sample Output 2:** |
|---|---|
| `51 25` | `slowpoke spotted!` |

| **Sample Input 3:** | **Sample Output 3:** |
|---|---|
| `50 25` | `born to run!` |

# Clout Chasing

*Filename:* `clout`

At the UCF High School Programming Tournament (HSPT), while all the students were contemplating solutions to programming problems, a judge named Sachin Sivakumar was contemplating something of a different manner: clout. There was an aspect of Sachin that set him apart from the other judges: he was a clout chaser.

Sachin harbored a peculiar belief that one's clout in the world of programming tournaments was directly correlated with the uniqueness and grandeur of their name. This conviction led him to devise a name-based metric to measure his clout against the other judges. Fortunately for him, he held an advantage that few could boast: alliteration. He decided then and there that clout was determined by the delightful dance of letters in one's name—does it have words starting with the same letter? Armed with this unique metric, Sachin set out on a quest to secure his position as the reigning clout champion among the HSPT judges.

However, there was one rival for his clout supremacy, a judge by the name of Cameron Custer, whose name rivaled his in alliteration. With a touch of competitive spirit, Sachin introduced a second parameter to his clout metric—the total number of letters in the judge's name. Longer names have additional grandeur and clout. You may have objections to these metrics of clout, including that these metrics seem incredibly contrived to help Sachin specifically or that clout is determined not by one's name but by one's actions. Sachin would have you keep your objections to yourself.

Sachin went to work, scoring the clout of himself and his fellow judges to determine the name of the clout champion. Determine the judge with the most clout according to Sachin's definition.

**The Problem:**

Given a list of HSPT judge names, determine which judge has the most clout as measured by whether their name is alliterative and breaking ties by picking the name with more letters. As a reminder, two words are alliterative if they start with the same letter.

**The Input:**

The first line consists of one positive integer, $n$ ($1 \leq n \leq 10^4$), representing the number of HSPT judges. The next $n$ lines each contain a first and last name separated by a single space, representing the name of an HSPT judge. The first and last names will only be lowercase English letters and will each be at most 20 characters. No two judges will have the same name. It is guaranteed that the input will produce a single name with the most clout after both rules are applied (no ties for first place).

**The Output:**

Print out the first and last name (separated by a single space) in lowercase letters of the HSPT judge with the most clout.

**Sample Input 1:**

```
3
sathvik kuthuru
sachin sivakumar
cameron custer
```

**Sample Output 1:**

```
sachin sivakumar
```

**Sample Input 2:**

```
5
natalie longtin
lior barak
tyler marks
glenn martin
chris gittings
```

**Sample Output 2:**

```
natalie longtin
```

**Sample Input 3:**

```
2
gennady korotkevich
brian barak
```

**Sample Output 3:**

```
brian barak
```

# DNA Sequences

*Filename:* `dna`

Thomas has just finished getting his degree in computer science! Since going into the real world is scary, he has decided to pursue a master's degree in bioinformatics, which combines the fields of biology and computer science. As part of his thesis, Thomas is constructing a database of known genome sequences in avocados. Who said research couldn't be interesting?

His database currently has $n$ entries, where each entry is an exactly $k$-character long string made of the characters A, C, G and T, representing the four nucleotides found in DNA: adenine, cytosine, guanine and thymine. Thomas has decided to search for more avocado genome sequences to add to his database, but he doesn't want to waste time looking for a sequence he has already cataloged. Since you are a good friend of Thomas's, he has asked you to find him any $k$-character long genome sequence that isn't already in his database. He's not picky, so he will accept any such sequence!

**The Problem:**

Given a list of genome sequences, find a genome sequence of length $k$ that is not in the list.

**The Input:**

The first line of input will contain 2 integers, $n$ ($1 \leq n \leq 50{,}000$) and $k$ ($8 \leq k \leq 50{,}000$), representing the number of genome sequences and the length of each genome sequence, respectively. It is also guaranteed that the product of $n$ and $k$ does not exceed 400,000. The following $n$ lines will each contain exactly $k$ characters, each of which will be either A,C,G,T. Additionally, it's guaranteed all DNA sequences given will be distinct.

**The Output:**

Output a single line containing a string that does not occur in the genome sequence database. Only strings that are exactly $k$ characters long consisting of the characters A, C, G, T will be considered correct.

**Sample Input 1:**

```
4 9
AAATTTGGG
TAGCATACT
ACACACACA
ACCCCCCCC
```

**Sample Output 1:**

```
CATCATCAT
```

**Sample Input 2:**

```
1 8
CATCATCA
```

**Sample Output 2:**

```
TAGTAGTA
```

# The Ducks of Danger Island

*Filename:* `ducks`

A mysterious magic took hold of Danger Island one night, leaving all of its quacking inhabitants with the remarkable ability to shoot bolts of lightning from their feet. The ducks were a peaceful population, and seldom felt the need to use their electrifying powers. But one fateful day, the clear sky turned stormy, and a flying pirate ship descended from the gathering gray clouds. It was an ambush by the Marauding Macaws!

The ducks of Danger Island hatched a daring plan. On a pedestal in the center of the island sat the Webbed Water Gem, a magical artifact with the ability to enhance the range and power of the ducks' lightning bolts. As told in the ancient scriptures of Danger Island, a duck must balance the gem on his foot in order to receive its magic. The ducks decided to line up and face this unstable undertaking one at a time.

Some of the ducks are more surefooted than others. Specifically, each duck has a certain probability of dropping the gem while balancing it. If a duck drops the gem, he will still receive the gem's magic, but the gem will shatter and become unusable for all the ducks after him in line.

The battle cries of the Marauding Macaws drew steadily closer, and a critical question hung in the air: How could the ducks of Danger Island ensure their victory?

**The Problem:**

Given the probability that each duck will drop the Webbed Water Gem, print the expected number of ducks that will have their lightning powers amplified if they line up in the best way.

**The Input:**

The first line of input contains an integer, $n$ ($1 \leq n \leq 10^5$), representing the number of ducks on Danger Island. The second line of input contains $n$ integers, where the $i^{\text{th}}$ integer $p_i$ ($0 \leq p_i \leq 100$), represents the percent chance that the $i^{\text{th}}$ duck will drop the gem while balancing it.

**The Output:**

Output the expected number of ducks that will have their lightning powers amplified. Your answer will be considered correct if its absolute or relative error does not exceed $10^{-3}$.

**Sample Input 1:**

```
5
0 100 0 100 0
```

**Sample Output 1:**

```
4
```

**Sample Input 2:**

```
2
50 50
```

**Sample Output 2:**

```
1.5
```

**Sample Input 3:**

```
10
50 79 48 39 41 53 98 17 32 50
```

**Sample Output 3:**

```
3.14168
```

# Optical Illusion

*Filename:* `illusion`

Anders loves optics. As he was exploring a funhouse, he entered his favorite room: the mirror maze! This maze isn't composed of typical mirrors, though. Instead, these mirrors only reflect portions of an image and leave the rest of the image the same.

This completely broke everything Anders knew about optics, and he started exploring what different things looked like in these mirrors. He held up an array of items and the mirrors reflected a certain range of these items. He sent you a picture of this peculiar reflection along with the range of items that were reflected and challenged you to determine what the original array of items looked like.

**The Problem:**

Given an array of items, represented by integers, and a range that was reflected, determine what the original array looked like. A reflection will be represented by the corresponding subarray being reversed.

**The Input:**

The first line of input will consist of three integers: $n$, $L$, and $R$ ($1 \leq L \leq R \leq n \leq 10^5$) where $n$ is the size of the array, and $L$ and $R$ represent the range of indices (inclusive) that was reflected in the array. The second line of input will consist of $n$ integers between 0 and 100, inclusive, representing the elements at each position in the reflected array.

**The Output:**

Output a single line consisting of $n$ integers: the original array that Anders had.

**Sample Input 1:**

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

**Sample Output 1:**

```
1 2 3 4
```

**Sample Input 2:**

```
6 4 6
100 5 51 51 92 39
```

**Sample Output 2:**

```
100 5 51 39 92 51
```

# Mechanized Mob

*Filename:* `mob`

The year is 4202. Everything is exactly the same as in 2024, or that is what our robot overlords tell us at least. Aren't these robots great!? They make our lives so much easier: they cook for us, clean for us, and tell us exactly what to think and believe!

Well, they aren't all great. Those vacuum "robots" (I think our ancestors called them Roombas?) are quite dumb compared to our superior robot overlords and will often form mobs and get themselves stuck. When this happens they shut down and reboot, which only allows them to move in one direction. My robot overlord has told me that I think this is quite annoying! Can you help me resolve this issue?

To be more specific, my robot overlord has given me a map of the room, which is modeled as a rectangular grid. There will be five different symbols in this grid:

- '.' that represents an empty cell
- '<' that represents a robot that can only move to the left
- '>' that represents a robot that can only move to the right
- '^' that represents a robot that can only move upwards
- 'v' that represents a robot that can only move downwards

To resolve the traffic, can you help me find an order in which the robots can be removed from the grid? A robot can only be removed if it can reach the outside of the grid without crashing into another robot, and once a robot starts moving, it cannot stop moving until it leaves the room. Because the intelligence of these Roombas is questionable, the next Roomba will only begin moving if and when the current Roomba successfully exits the grid.

**The Problem:**

Given the grid containing the robots, determine an order in which the robots can successfully be removed from the grid.

**The Input:**

The first line of input consists of two integers, $n$ and $m$ ($1 \leq n, m \leq 200$) representing the number of rows and columns in the grid, respectively. Following will be $n$ lines, each consisting of $m$ characters. Each character will be one of the five characters listed above. It is guaranteed that there is at least one robot on the grid.

**The Output:**

If it is possible to remove all the robots, output $k$ lines, where $k$ is the number of robots in the grid. The $i^{th}$ line should contain two integers $r$ ($1 \leq r \leq n$) and $c$ ($1 \leq c \leq m$) indicating that the robot at the $r^{th}$ row and the $c^{th}$ column should be the $i^{th}$ robot removed. If it is not possible to remove all of the robots, output "Dang Roombas!" If there is more than one solution, output any of them.

**Sample Input 1:**

```
3 3
.v.
>>>
.v.
```

**Sample Output 1:**

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

**Sample Input 2:**

```
3 3
>.v
...
^.<
```

**Sample Output 2:**

```
Dang Roombas!
```

**Sample Input 3:**

```
3 4
<<<<
v<<^
>>>^
```

**Sample Output 3:**

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

# Palindromic Passwords

*Filename:* `password`

Thomas has recently begun playing "The Password Game". In the Password Game, you are challenged to make a password satisfying an increasingly ridiculous set of rules, such as "Your password must contain the current phase of the moon as an emoji." A recently released expansion of the game has created a new set of rules for Thomas to play through. The first two rules are boring, but the third rule is a little more interesting.

- Rule 1: Your password must consist of at least 1 and at most 100,000 characters.
- Rule 2: Your password must consist of only lowercase alphabetic letters.
- Rule 3: Your password must contain exactly $K$ (not necessarily distinct) palindromic substrings.

In a play through of the game that Thomas watched (for entertainment, of course, not to cheat/steal strategies), it was explained that a palindrome is any non-empty string that when reversed is unchanged. By extension, a palindromic substring is a continuous range of characters in the string that is a palindrome.

Thomas needs help satisfying the first three rules and has decided to ask you for help.

**The Problem:**

Given a value $K$, construct any string that follows all three password rules.

**The Input:**

The first and only line of input will consist of a single integer, $K$ ($1 \leq K \leq 1{,}000{,}000{,}000$), representing the exact number of palindromic substrings that must occur in the password.

**The Output:**

Output a single line containing a string of lowercase letters that is between 1 and 100,000 characters long and contains exactly $K$ palindromic substrings.

| **Sample Input 1:** | **Sample Output 1:** |
|---|---|
| `4` | `hspt` |

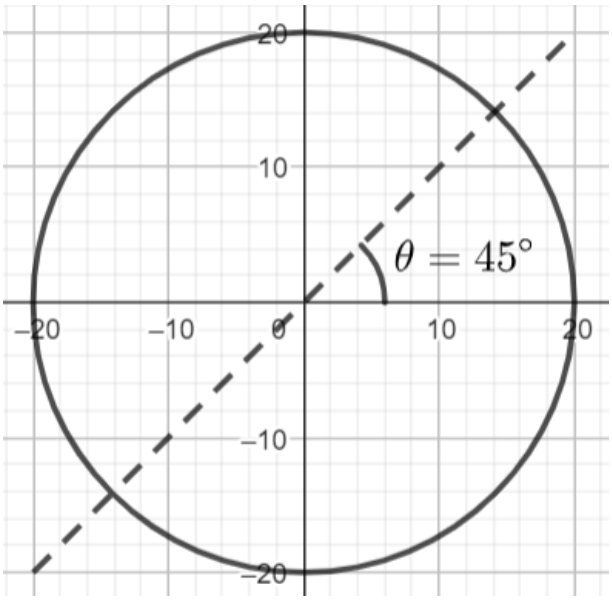
| **Sample Input 2:** | **Sample Output 2:** |
|---|---|
| `12` | `abacaba` |

# Pizza Party

*Filename:* `pizza`

Glenn graciously purchased a comically large pizza for the HSPT Judges to thank them for their efforts in preparing the contest. He plans to cut the pizza through the center as many times as there are judges, so each judge will receive 2 pizza slices. For example, if there are 4 judges, he will cut the pizza 4 times so that there are 8 slices. Glenn is not a master pizza cutter, so each pizza slice may not be equal in size, and each judge may not receive an equal amount of pizza.

However, one judge, Asir, eyed the pizza slices with an extra helping of enthusiasm. Asir has decided to eat half of the pizza slices, ruining Glenn's planning. Because Asir will grab these slices all at once, he will take a wedge of the pizza, consisting of adjacent pizza slices. Glenn now wants to know the maximum area of pizza Asir could eat.




Sample 1: A pizza with a 20-inch radius and one cut at 45 degrees

**The Problem:**

Given the radius of the pizza and the angles of Glenn's cuts relative to the positive $x$-axis, find the maximum area of pizza that Asir could eat.

**The Input:**

The first line of input will contain two integers $n$ ($1 \le n \le 180$) and $s$ ($20 \le s \le 10^3$), representing the number of HSPT judges and the radius of the pizza in inches, respectively. Each of the next $n$ lines will have a single integer $a_i$ ($0 \le a_i < 180$), representing the angle in degrees of the $i^{th}$ cut relative to the positive $x$-axis, in the counterclockwise direction. It is guaranteed that all of the cut angles are distinct.

**The Output:**

Output a line containing a single real number: the maximum area, in square inches, of pizza Asir can eat. This area must be accurate to an absolute or relative error of $10^{-4}$.

**Sample Input 1:** **Sample Output 1:**

| | |
|---|---|
| 1 20<br>45 | 628.318531 |

**Sample Input 2:** **Sample Output 2:**

| | |
|---|---|
| 3 27<br>42<br>5<br>71 | 1145.110522 |

# The Royal Wedding

*Filename:* `wedding`

The faithful union of Princess Circelia and Knight Sagramore of the Round Table is widely anticipated to be the bash of the century. These lofty expectations fall on the shoulders of Ernest Eventide, the royal wedding planner. Ernest has spent the past several months mowing the grass in the east castle courtyard, placing bulk orders for cream puffs and finger sandwiches, and color-matching favors, ribbons, and flowers. Now, it's time to call Thomas's Gazebos for a price estimate.

As the line rings, Ernest thinks through the budgeting and spacing requirements again. To accommodate all of the wedding guests, the gazebo must allow for a certain amount of space. Ernest would like to spend the least amount of money possible on a gazebo with this area.

Ernest jolts as a cheery voice sounds on the other end of the line.

*Thank you for calling Thomas's Gazebos, where our motto is "linear pricing, flexible placing." We charge per yard of the gazebo's perimeter, and the shape is completely up to you!*

Ernest closes his eyes and sighs. He is no stranger to planning stunning ceremonies, but choosing the right shape for the wedding gazebo is a conundrum that seems a bit tougher than the rest.

**The Problem:**

Given the area requirement and the price per yard of perimeter, determine the minimum cost of the wedding gazebo.

**The Input:**

The input consists of two space-separated integers, $A$ and $P$ ($1 \leq A, P \leq 10^6$), representing the gazebo area requirement in square yards and the price per yard of the gazebo perimeter.

**The Output:**

Output a real number rounded to two decimal places: the minimum cost of the wedding gazebo.

| **Sample Input 1:** | **Sample Output 1:** |
|---|---|
| `100 1` | `35.45` |

| **Sample Input 2:** | **Sample Output 2:** |
|---|---|
| `1 100` | `354.49` |

| **Sample Input 3:** | **Sample Output 3:** |
|---|---|
| `10000 17` | `6026.34` |