UTD Battle Of The Brains Fall 2024 Novice

Rules:

1) **DO NOT TURN THE PAGE UNTIL THE CONTEST HAS STARTED**
2) No electronic devices aside from the computer your team is assigned (Cell phones off, no tablets, etc...).
3) No communication between teams.
4) Programming language reference books are permitted.
5) Use of the internet aside from accessing the contest site is not permitted. Failure to follow this rule will result in disqualification.
6) Time and memory limits are stated in the corresponding problem.
7) If any issues arise during the contest, contact the proctors.

Scoring:

1) The team to solve the most problems wins.
2) Ties will be broken by penalty points.
3) Wrong submissions result in 20 minutes of penalty points.
4) Correct submissions give you penalty points equal to the elapsed time in the contest.

Good Luck!

This Page Is Intentionally Left Blank

Fast I/O:

Several of the problems this year are I/O-intensive. As a result, we have provided a short description on how to quickly take in input and process output in each suite of languages...

C++:

Add the following to the beginning of your main method:

```
ios_base::sync_with_stdio(false);
cin.tie(nullptr);
```

Also, only use `std::endl` sparingly since it flushes output, which can be expensive if done repeatedly.

Java:

Add the following to the beginning of your main method:

```
 1: import java.io.BufferedReader;
 2: import java.io.IOException;
 3: import java.io.InputStreamReader;
 4: import java.io.PrintWriter;
 5: import java.util.StringTokenizer;
 6: ⋮
 7: BufferedReader file = new BufferedReader(new InputStreamReader(System.in));
 8: PrintWriter out = new PrintWriter(System.out);
 9: ⋮
10: StringTokenizer st = new StringTokenizer(file.readLine());          // Must redefine for a new line of input
11: int a = Integer.parseInt(st.nextToken());               // Pulls the next token from the line of input from line 9
12: ⋮
13: file.close();                                                                  // Closes standard input
14: out.close();                                                 // Flushes output and closes standard output
```

Use the method `file.readLine()` to read in an entire line of input (null if no such line exists), and use a `st.nextToken()` to read in the next contiguous block of data from a whitespace-separated line.

Python:

Use `sys.stdin` and `sys.stdout` instead of the built-in `input()` and `print()` as this can speed up I/O operations. Add the following imports:

```
import sys
```

To read input, use:

```
data = sys.stdin.read()                                   // Reads all input at once
lines = data.splitlines()                                 // Splits input into lines
```

For output, use:

```
sys.stdout.write(str(output) + “\n”)         // Use sys.stdout.write() instead of print()
```

Kotlin:

Kotlin supports fast I/O through essentially the same I/O classes that Java uses. Add the following to your main function:

```
import java.io.BufferedReader
import java.io.IOException
import java.io.InputStreamReader
import java.io.PrintWriter
```

To read input:

```
val reader = BufferedReader(InputStreamReader(System.`in`))
val out = PrintWriter(System.out)
⋮
val line = reader.readLine()
val tokens = line.split(" ")
val a = tokens[0].toInt()
```

To output results:

```
out.println(result)                    // Use PrintWriter for fast output
out.flush()                            // Ensure to flush the output
```

In Kotlin, avoid using the built-in `readLine()` and `println()` too often, as they are relatively slower. Use `BufferedReader` and `PrintWriter` for more efficient I/O operations.

C:

In C, the standard functions `scanf()` and `printf()` are typically used for I/O. Here are a few tips for fast I/O in C:

```
#include <stdio.h>
⋮
int a;
scanf("%d", &a);                       // Use scanf() for input
printf("%d\n", a);                     // Use printf() for output
```

Tips for fast I/O in C:

- Use `scanf()` and `printf()` instead of other more complex I/O methods for speed.
- Avoid flushing output buffers repeatedly. Using `fflush(stdout)` inside loops can be costly, so only flush when necessary.
- Batch input and output when possible to minimize the number of function calls.

Bishop Attack

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

Problem Statement

A chess board has columns 'a' through 'h' and rows 1 through 8 resulting in a board with a total of 64 squares. The adjacent squares alternate colors between black and white: square 'a1' (column 'a' and row 1) is black, 'a2' is white, 'b1' is white, and so on. While many people's favorite piece is the Queen, Temoc's favorite piece remains as the Bishop.

A bishop is a piece that attacks all squares on the diagonals on which the bishop is placed. A bishop placed on the square 'd4' attacks the squares 'd4', 'e5', 'c5', 'c3', 'e3', 'f6', etc. as long as there is a direct line of sight along a diagonal.

You are given three distinct positions: an attacking bishop, A, a target bishop, B, and a pawn, P. Note that if A and B are in the same diagonal but P is in between them, then A cannot reach B in one move. Additionally, P can neither be jumped over nor captured. Temoc wants to know the minimum number of moves needed for A to reach B under this constraint if it is possible to do so.

Input Format

The input consists of three lines. Each line consists of a single character, c, a space, and a single integer i denoting the square that the piece is on. Line 1 denotes the position of A, line 2 for B, and line 3 for P.

Constraints

$c \in \{\text{a}, \text{b}, \ldots, \text{h}\}$
$i \in \{1, 2, \ldots, 8\}$

Output Format
Print a single integer n indicating the minimum number of moves A needs to reach B, or 0 if it is not possible to do so.

Sample I/O 1

Input	Output

`a 1`
`h 8`
`d 4`
	`5`

Explanation 1
A possible valid solution of minimal distance is a1 $\rightarrow$ c3 $\rightarrow$ b4 $\rightarrow$ d6 $\rightarrow$ e5 $\rightarrow$ h8. Note that since we cannot capture the pawn, we must go around it.

Sample I/O 2

Input	Output

`a 1`
`h 8`
`g 7`
	`0`

Explanation 2
Since we cannot capture the pawn, we are unable to reach B from A.

Chess Board
A sample chessboard has been provided below for your convenience.

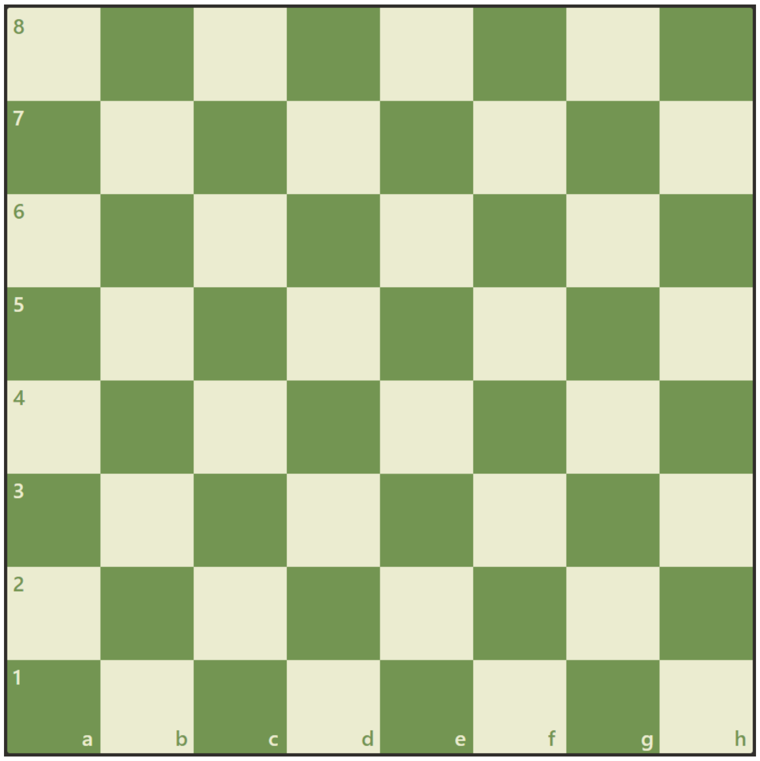

Figure 1: A Sample Chess Board

Two Brains Are Better Than One (Easy)

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

This is the easy version of the problem. The only difference from the hard version is the constraint on n and r_i. The easy version appears in both packets, while the hard version only appears in the advanced packet.

Problem Statement
The students at UTD are known as Comtets, and each Comet has a rating, r. A "sub-rating" of some Comet's rating, r_i, is a value v such that r_i is an integer multiple of v. Note: values 1 and r are both considered sub-ratings of r.

Let x be the number of values that are a sub-rating of r. A Comet's rating is said to be "cool" if x is an odd number ("cool" refers to the temperature of a Comet, not any kind of perceived social status).

Comets are paired to form teams. The rating of a team R is the product of the rating of the two Comets on the team. The rating of a team is said to be cool if, when separate, the two Comet's ratings are not cool, but the team's rating as a whole is cool.

Temoc, being the caring mascot that he is, is interested in knowing how many unique teams are considered to be cool given a set of ratings for various different Comets.

Input Format
The first line of input consists of an integer n denoting the number of Comet ratings to follow. The second line of input consists of n space-separated integers, $r_1, r_2, ..., r_n$, denoting the ratings of the n Comets. Rating r_i is considered to be the rating of the i^{th} Comet.

Constraints
$1 \leq n \leq 10^4$
$2 \leq r_i \leq 10^9$

Output Format
Output a single non-negative integer, p, where p is the number of unique team pairings that are considered to be cool.

Sample I/O 1

Input	Output
5 2 3 4 7 2	1

Explanation 1
The only non-cool number here is $r_2 = 4$. However, the only cool team is $(r_0, r_4) = (2, 2)$.

Sample I/O 2

Input	Output
8 32 2 5 20 3 242 242 363	8

Cylinder Volume

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

Problem Statement

Temocs live in a land where food can be grown only for a part of the year, and due to the harsh temperatures, they need to stay indoors during the winter. Because of this, they must store their food, grains, water, fuel, etc. so that they can survive the harsh winter.

Their method of storage is to make cylinders whose total height equals that of the town's storage godown, stacking the cylinders one on top of each other. Moreover, the cylinder at the bottom of the pile must have the greatest radius and the the one at the top must have the least radius. The same applies to the height of the cylinders. More generally, if cylinder x is on top of cylinder y, then cylinder y must have a radius that is strictly greater than x's radius, and it must also have a height that is strictly greater than x's height. All heights and radii of any cylinder are positive integers measured in centimeters.

Figure 2: An Example of a Storage Godown

Every year a set of lucky numbers is given to the population of Temocs by the food Oracle who states the exact number of cylinders they should make, as well as the sum of the radii of all of those cylinders, in order to increase their fortune during the winter time. Your goal is to maximize the volume of the cylinder at the top of the stack while abiding by all design restrictions.

Input Format

The first line consists of the number of test cases, t. The remaining input denotes the t test cases.

Each test case consists of a single line with three space-separated positive integer values h, r, and k. Here, h is the sum of the height of all k cylinders, r is the sum of the radii of all k cylinders, and k is the number of cylinders to be made.

Constraints

$1 \leq t \leq 5 \cdot 10^5$
$1 \leq h, r \leq 10^{18}$
$1 \leq k \leq 10^9$

Output Format
For each test case print the volume of the topmost cylinder divided by $\pi \cdot r_{\text{top-most}}$ in a line by itself. Since this value can get quite large, print your answer modulo $10^9 + 7$. If the arrangement is impossible, then print 0.

Sample I/O 1

Input	Output

2
4 5 1
875 561 8
	20
	6930

Explanation 1
In the first test case, there is only a single cylinder, so we are free to let it be the sum of the height and radii.

In the last test case, we can make a topmost cylinder with height 105 and radius 66.

Good BoB

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

Problem Statement

The semester-wise UTD training camp – BotB – trains the interested students, i.e. BoBs, in algorithmic problem-solving. This enhances comprehension, critical thinking, and knowledge in many domains including mathematics, data structures, and algorithms. Further, it is a fun event with their team and coach where they get to socialize and enjoy a meal together contributing to their social skills, which many are lacking, initially.

This lays a foundation for BoBs to succeed in life and contribute to society. All serious participants are known to be among the top achievers in society either academically or financially. Initially, the score of a BoB is 0; however, they can increase their score through their actions after receiving the training. Namely, if they satisfy a subset of the following conditions, they will receive the sum of the denoted point values:

1) If a BoB joins a university, then that BoB receives an additional u points.
2) If a BoB joins UTD, then that BoB receives an additional d points.
3) If a BoB competes for UTD in the ICPC, then that BoB receives an additional i points.
4) If a BoB coaches at UTD, then that BoB receives an additional c points.
5) If a BoB completes a Master's in CS at UTD, then that BoB receives an additional m points.
6) If a BoB completes a PhD in CS at UTD, then that BoB receives an additional p points.
7) If a BoB completes a Master's in CS at either Tag or YooT, then that BoB receives an additional $-m$ points.
8) If a BoB completes a PhD in CS at either Tag or YooT, then that BoB receives an additional $-p$ points.
9) If a BoB competes for UTD in the ICPC and completes either a Master's in CS at UTD or a PhD in CS at UTD, then that BoB receives an additional b points.
10) If a BoB competes for UTD in the ICPC and completes both a Master's in CS at UTD and a PhD in CS at UTD, then that BoB receives an additional $2b$ points
    - **Note:** This is instead of earning b points for #9

Note that a few of these options are mutually exclusive with one another. Namely:

- A BoB will only (optionally) receive a Master's from a single school.
- A BoB will only (optionally) receive a PhD from a single school.
- A BoB can only compete for a school that they attend.

Also, since this calculation is being done over an entire BoB's lifetime, it is possible that a BoB both competes and coaches for UTD.

Input Format
The first line consists of the number of test cases, t. The remaining input denotes the t test cases.

The first line of a test case consists of seven space-separated integer values: u, d, i, c, m, p, and b. The second line of a test case consists of an integer n. The next n lines denote a subset of the following statements, in any order, one statement per line, without any duplicates:

- `BoB joins a university`
- `BoB joins UTD`
- `BoB competes for UTD`
- `BoB coaches at UTD`
- `BoB completes a Master's in CS at UTD`
- `BoB completes a PhD in CS at UTD`
- `BoB completes a Master's in CS at Tag`
- `BoB completes a Master's in CS at YooT`
- `BoB completes a PhD in CS at Tag`
- `BoB completes a PhD in CS at YooT`

Constraints
$1 \leq t \leq 100$
$1 \leq n \leq 10$
$0 \leq u, d, i, c, m, p, b \leq 1000$

Output Format
Print t lines corresponding to each test case. If the score of the i^{th} BoB is less than 0, then print "`Needs Serious Improvement`". Otherwise, it is less than 40% of the maximum possible score, then print "`Needs Improvement`". Otherwise, if it is greater than 90% of the maximum possible score, then print "`Good BoB`". However, if it is the maximum possible, instead print "`Awesome BoB`". Otherwise, simply print the score.

Sample I/O 1

Input	Output

`3`
`1 2 3 4 5 6 7`
`4`
`BoB joins a university`
`BoB joins UTD`
`BoB completes a Master's in CS at UTD`
`BoB completes a PhD in CS at UTD`
`1 2 3 4 5 6 7`
`6`
`BoB completes a PhD in CS at UTD`
`BoB joins a university`
`BoB completes a Master's in CS at UTD`
`BoB competes for UTD`
`BoB joins UTD`
`BoB coaches at UTD`
`1 2 3 4 5 6 7`
`3`
`BoB joins a university`
`BoB completes a Master's in CS at YooT`
`BoB completes a PhD in CS at Tag`
	`14`
	`Good BoB`
	`Needs Serious Improvement`

Explanation 1
In all test cases, the maximum number of points that a BoB can earn is $1+2+3+4+5+6+(2\cdot 7) = 35$.

BoB #1 earns $1+2+5+6 = 14$ points. Since $14 \not< 0.4 \cdot 35 = 14$, we only print out the number of points that BoB #1 earned.

BoB #2 earns $6+1+5+3+2+4+(2\cdot 7) = 35$ points. Since this is the maximum number of points that someone can earn, they are denoted as an "`Awesome BoB`".

BoB #3 earns $1+(-5)+(-6) = -10$ points. Since this is less than 0, we note that this BoB "`Needs Serious Improvement`" (clearly, they should have attended UTD instead, and been more like BoB #1 and BoB #2 – perhaps then they might have earned more points during their life).

Chromosomal Similarity Frequency

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

Problem Statement
You are given a string of length n over the alphabet $\{\text{a}, \text{b}, \ldots, \text{z}\}$ denoting a reference chromosome, C. This chromosome is cyclic, that is, after the n^{th} symbol, we wrap back around to the first symbol. You are also given a smaller experimental chromosome, E. A segment of a chromosome $A[1 \ldots m]$, is denoted as $A[i \ldots j]$ where $1 \leq i \leq j \leq m$ consists of the i^{th} symbol of A to the j^{th} symbol of A.

In general, determine the total number of occurrences of all maximal length segments of E, that are all present in C. Note that there may be multiple places in C where a given segment, $A[i \ldots j]$, occurs. Additionally, there may be multiple segments, $A[i \ldots j]$ and $A[k \ldots \ell]$, where $j - i + 1 \equiv \ell - k + 1$ is greatest, and are both present in C.

Input Format
The input will consist of three lines. The first line has two space-separated positive integers c and e, where c is the length of C and e is the length of E. The second line denotes the string C. The third line denotes the string E.

Constraints
$1 \leq c \leq 10^5$
$1 \leq e \leq 10^3$
$e \leq c$

Output Format
A single integer denoting the total number of occurrences of the largest segments of E that are present in C.

Sample I/O 1

Input	**Output**

`10 5`
`abcdefhijk`
`jkabc`
	`1`

Explanation 1
Note that the entirety of E exists in C when we wrap-around. That is, "jk" appears as the last two symbols, and "abc" appears as the first three in C.

Sample I/O 2

Input	**Output**

`12 4`
`lababab dedel`
`deab`
	`5`

Explanation 2
Note that the longest substring of E that exists in C is of length 2. These are the substrings “de” which appears twice in C, and “ab” which appears three times in C.

Sample I/O 3

Input	Output

`13 13`
`abcdefghijklm`
`nopqrstuvwxyz`
	`0`

Explanation 3
Note that the set of letters that comprise E and C are disjoint and do not share any similarities. Thus, there are no strings of maximal length that are shared between E and C.

Pairs Apart

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

Problem Statement
Comets are proud students of UTD. The i^{th} student s_i has a unique positive integer "Mocness" rating of $M[i]$ which indicates how cool the student is (this time, referring to the perceived social status of the student rather than their temperature). Every year, BigT – the leader of Comets – decides to make some $k > 0$ unique pairs of Comets that conform to the following rules:

- The first pair p_1 is a pair of students s_i, s_j such that the difference in their Mocness ratings is the highest. That is $|M[i] - M[j]|$ is the maximum among all pairs i, j.
- Then, these students are no longer considered, and the second pair is chosen using the same rule.
- For each pair, p_i, the strength of the pair is the sum of the Mocness scores of both students.

Input Format
The input consists of two lines. The first line contains two space-separated integers n and k, where n is the total number of students and k is the number of pairs of students that are to be formed. The second line contains n space-separated integers denoting the Mocness ratings of the students.

Constraints
$2 \leq n \leq 10^6$
$1 \leq k \leq 1000$
$k \leq \left\lfloor \frac{n}{2} \right\rfloor$
$\forall i,\ 1 \leq i \leq n\ : 1 \leq M[i] \leq 10^6$

Output Format
Print k space-separated integers indicating the strengths of pairs $p_1, p_2, \ldots, p_k$ in that order.

Sample I/O 1

Input	Output

`2 1`
`3 4`
	`7`

Explanation 1
There is only enough to form the one team required, which is $(4, 3)$ and has a sum of 7.

Sample I/O 2

Input	Output

`9 3`
`23 2 7 99 34 3 1 6 5`
	`100 36 26`

Explanation 2
The teams that are formed are $(99, 1)$, $(34, 2)$, and $(23, 3)$, in that order, with $7, 6,$ and 5 being left without a group.