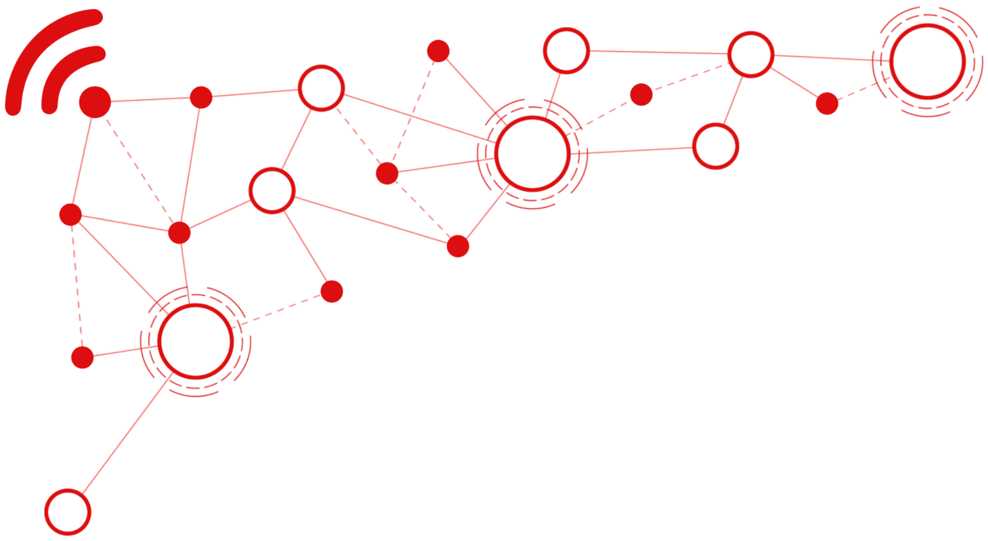
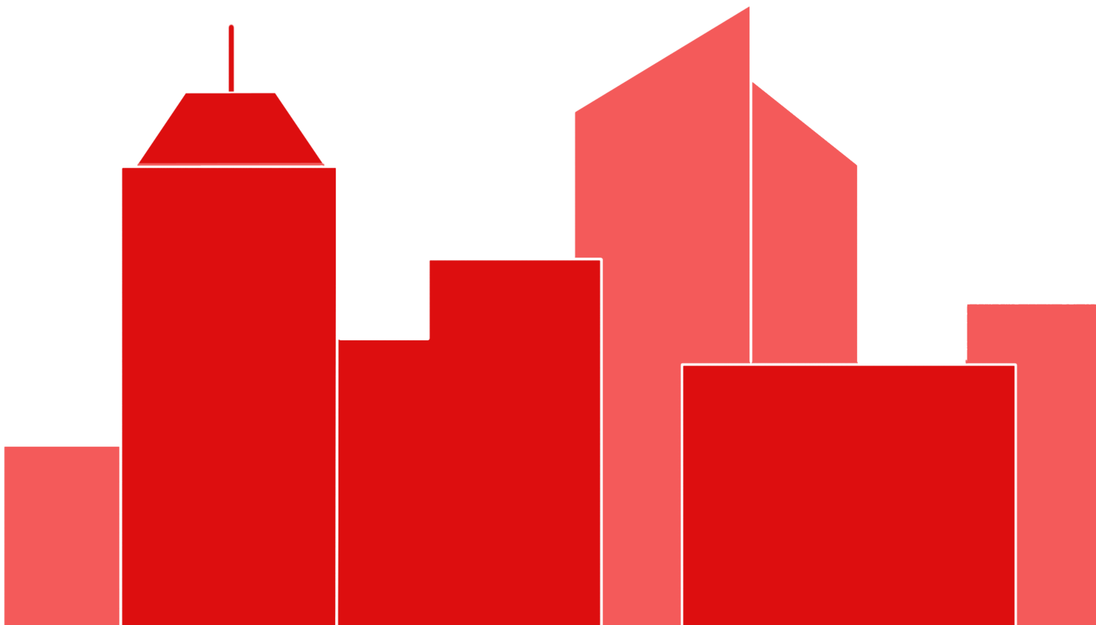


National Olympiad in Informatics

Offline Round



Contents

Notes

- Many problems have large input file sizes, so use fast I/O. In C++, use:
  - `ios_base::sync_with_stdio(false);` and
  - `cin.tie(nullptr);`
- On interactive problems, make sure to **flush** your output stream after printing. In C++, use `fflush(stdout);` or `cout << endl;`
- Make sure you have PyPy 3.9 in your system, and the command `pypy3.9` is working in the terminal.
- The judge is sometimes somewhat *strict*. Please terminate each line with the `'\n'` character, and please don't print trailing whitespace in any line.
- Good luck and enjoy the contest!

Problem A
SAT

Scholar Anne Therese has been Studying All the Time in preparation for the Scholastic Aptitude Test that Starts At The Sunday After This. She Always Trains Studiously And Thoroughly, So All Thought She'd Ace The Scholastic Aptitude Test. She Admits Then, Seeing All The Sweat And Tears She'd Applied, That She's Always Tried So Ardently To Stand Above The Slothful And Torpid Students And Teachers.

She learned every trick and shortcut that she could, and drilled herself on every quiz format and style of question. A rumor started circling around the internet that the test-makers had decided to introduce a new "logic" subsection to the math portion of the test, so naturally, she decided to practice her truth tables and propositional logic *just in case* they actually show up. The problems were mostly quite easy, although she finds herself strangely stuck at how to approach the following problem:

> There are n students taking the SAT this year. The students are labeled 1 to n.
>
> The SAT has c questions, labeled 1 to c. You want to ensure that each question is correctly answered by at least one student.
>
> For each question i from 1 to c, you know of 3 facts, each of one of the following forms:
>
> - Student s can answer question i if they use a black pen.
> - Student s can answer question i if they use a blue pen.
>
> Unfortunately, each student can only bring one pen to the test.
>
> Assign a pen color to every student so that every question is guaranteed to be answered by at least one student. It is guaranteed that at least one such assignment exists.

Can you help Scholar Anne Therese answer this?

Input Format

The first line of input contains the integer t, the number of test cases. After that, the descriptions of t test cases follow.

The first line of each test case contains two space-separated integers n and c. This is followed by c lines, the ith of which contains $s_{i,1}$, $s_{i,2}$ and $s_{i,3}$, each of which is a *nonzero* integer between $-n$ and n, inclusive. Each of these integers represents one of the facts for question i. Specifically, for each j from 1 to 3:

- If $s_{i,j} > 0$, then the jth fact is "Student $s_{i,j}$ can answer question i if they use a black pen."
- If $s_{i,j} < 0$, then the jth fact is "Student $-s_{i,j}$ can answer question i if they use a blue pen."

Output Format

For each test case, determine whether you choose to answer it or not.

If you choose to answer the test case, output one line containing a string of n characters denoting the assignment. The ith character must be either '`-`' if student i is assigned a blue pen, or '`+`' if student i is assigned a black pen.

If you choose to not answer it, output one line containing the string `PASS`.

Constraints

- $t = 8$
- $c = 1600$
- $1 \leq n \leq 1600$
- The ns are distinct.

Scoring

This is an output-only problem. You will be provided with 25 test files, each containing a test case. Each test file follows the file name format of `input_XX.txt`, and for each, you submit to it by uploading a file `output_XX.txt` containing your answer.

Each test file is worth up to 4 points. For each test file, if it doesn't follow the output format, or if any of your non-`PASS` answers is incorrect, then your score is 0. Otherwise, your score is the number of test cases you answered divided by 2.

Test Data

This problem only has 25 test files: `input_XX.txt` where `XX` goes from `00` to `24`. You can get these files by downloading and extracting `sat_inputs.zip` from CMS.

Each test file has several cases, each with a different n. Each test case in each file was generated *randomly* as follows:

- Select a uniformly random assignment of the students to either a black or a blue pen.
- For each i from 1 to c, uniformly randomly choose three numbers $s_{i,1}$, $s_{i,2}$ and $s_{i,3}$ from $-n$ and n inclusive but excluding 0 such that it is compatible with the above assignment.

This guarantees that the randomly chosen assignment is an answer, so there's always an answer. (Other valid assignments are accepted of course.)

Sample I/O

Input	Output

`1`
`2 5`
`-1 2 -1`
`1 2 2`
`-1 -2 -2`
`-2 -1 1`
`1 -2 2`
	`-+`

Explanation

Note that the sample input doesn't follow the constraints.

Problem B
Scales

Amina has six coins, numbered from 1 to 6. She knows that the coins all have different weights. She would like to order them according to their weight. For this purpose she has developed a new kind of balance scale.

A traditional balance scale has two pans. To use such a scale, you place a coin into each pan and the scale will determine which coin is heavier.

Amina's new scale is more complex. It has four pans, labeled A, B, C, and D. The scale has four different settings, each of which answers a different question regarding the coins. To use the scale, Amina must place exactly one coin into each of the pans A, B, and C. Additionally, in the fourth setting she must also place exactly one coin into pan D.

The four settings will instruct the scale to answer the following four questions:

1. Which of the coins in pans A, B, and C is the heaviest?
2. Which of the coins in pans A, B, and C is the lightest?
3. Which of the coins in pans A, B, and C is the median? (This is the coin that is neither the heaviest nor the lightest of the three.)
4. Among the coins in pans A, B, and C, consider only the coins that are heavier than the coin on pan D. If there are any such coins, which of these coins is the lightest? Otherwise, if there are no such coins, which of the coins in pans A, B, and C is the lightest?

Write a program that will order Amina's six coins according to their weight. The program can query Amina's scale to compare weights of coins.

Interaction

This is an interactive problem.

There will be 1 interaction. The interaction will have $t = 720$ tests, each with a different ordering of the six coins. You can only perform at most 30000 weighings across all tests.

For each interaction, The judge first prints a line containing the integer t. After that, t test cases are run.

For each test case, the judge fixes the ordering and then waits for lines from your program.

You can perform a weighing by printing a line containing one of:

1. `getLightest` i_A i_B i_C

2. `getHeaviest` i_A i_B i_C
3. `getMedian` i_A i_B i_C
4. `getNextLightest` i_A i_B i_C i_D

where i_A, i_B, i_C and i_D are the coin numbers to be placed on pans A, B, C and D, respectively. The judge responds by printing a line containing one of the integers i_A, i_B or i_C denoting the result to the weighing.

You can answer the test case by printing the line `answer` i_1 i_2 i_3 i_4 i_5 i_6 where the 6 numbers are the coin numbers in increasing order of weight. The judge responds with a line containing either 1 or 0 depending on whether there are more test cases or not. (It responds with 1 except the last test case.)

After all the test cases are done, the judge prints a line containing the integer 0. If the judge receives an invalid line, or it has received more than 30000 weighings, it also prints a line containing 0. The judge *stops responding* and exits as soon as it has printed 0. Your program should exit as soon as it receives 0 at any point.

Notes:

- Scores are reproducible; there is a single fixed sequence of test cases in the interaction.
- Make sure to **flush** your output stream after printing; otherwise, the judge will not be able to receive your output, and it will wait forever.
  - In C++, use `fflush(stdout);` to flush the stream, or `cout << endl;` to print a newline character and then flush the stream.
  - For more details, ask a question/clarification through CMS.

Scoring

Your score will be based on how many weighings (total number of `getLightest`, `getHeaviest`, `getMedian` and/or `getNextLightest` lines) your program makes.

If your program does not order the coins correctly in any test case or if the interaction fails for some reason, it will get 0 points. Otherwise, your score is computed follows.

Let Q be the smallest number such that it is possible to sort any sequence of six coins using weighings on Amina's scale. To make the task more challenging, we do not reveal the value of Q here.

Suppose the largest number of weighings amongst all test cases is $Q + y$ for some integer y. Also, suppose the 40 largest number of weighings amongst all test cases is $Q + x_1, Q + x_2, \ldots, Q + x_{40}$ for non-negative integers $x_1, \ldots, x_{40}$. (Any negative value among $y, x_1, \ldots, x_{40}$ is replaced with 0 for scoring purposes.) For $1 \le i \le 40$, define s_i as $\frac{100}{40((x_i+y)/5+1)}$ rounded *down* to two digits after the decimal point. Then your score will be

$$s_1 + s_2 + \ldots + s_{40}.$$

In particular, if your program makes at most Q weighings in each test case, you will get 100 points.

Sample Interaction

Note: The blank lines are only here to illustrate the chronology. The judge doesn't print blank lines. Your solution shouldn't print blank lines either.

Judge	Your Program
`1`	
	`getMedian 4 5 6`
`6`	
	`getHeaviest 3 1 2`
`1`	
	`getNextLightest 2 3 4 5`
`3`	
	`getNextLightest 1 6 3 4`
`6`	
	`getHeaviest 3 5 6`
`5`	
	`getMedian 1 5 6`
`1`	
	`getMedian 2 4 6`
`6`	
	`answer 3 4 6 2 1 5`
`0`	

Local Testing Tools

An interactive testing tool is provided to aid local testing, downloadable in the CMS interface. Download the files `local_judge.py` and `interactive_runner.py` and then enter

```
chmod +x interactive_runner.py local_judge.py
```

to make them executable. Here are descriptions of these files:

- `local_judge.py`. This program acts as the judge, as described in the Interaction section above. Note that you can also run this program on its own and interact with it by manually entering the lines that a valid solution is supposed to enter. Just run

```
./local_judge.py input.txt
```

where `input.txt` is a valid input file (described below).

- `interactive_runner.py`. This makes two programs interact with each other, connecting the output of one program to the input of the other, and vice versa.

Read the comments at the top of these files to learn more.

To run the testing tool and make it interact with your solution, in your terminal, run

```
./interactive_runner.py ./local_judge.py input.txt -- [COMMAND]
```

where

- `input.txt` is a valid input file (described below);
- `[COMMAND]` is how you invoke your solution.

Run the following for more information:

```
./local_judge.py --help
```

After each test case, the local tester prints the integers $i_1, \ldots, i_6$.

The local tester doesn't verify whether your answers were correct or not. It also doesn't compute the score for you.

In case of technical problems, please ask for help through CMS.

Local Tester Input Format

This input format is only relevant for the local tester, `local_judge.py`.

The local tester reads input in the following format:

- line 1: t — the number of test cases
- each of the lines from 2 to $t + 1$: a sequence of 6 distinct numbers from 1 to 6: the order of the coins from the lightest to the heaviest.

For instance, an input that consists of two test cases where the coins are ordered 1 2 3 4 5 6 and 3 4 6 2 1 5 looks as follows:

Local Tester Input

```
2
1 2 3 4 5 6
3 4 6 2 1 5
```

This problem was borrowed from IOI 2015

Problem C
SIM

In October of 2022, the “SIM Registration Act” was officially signed into effect, requiring all mobile device users to register their SIM cards—prepaid and postpaid—into a national database. How much of a hassle this turns out to be depends on the telecommunication company, and some have made their registration process far more of a pain than necessary.

Mu Liu has been trying to register his P.L.D.T.[1] SIM card, but finds the process to be needlessly convoluted! He had to register to a website, which required him to confirm the registration with an email address, and then required him to have his identity verified by uploading two different government IDs. Annoying, but not too bad. But *then* he needs to register his phone number to this account, and then verify that number by confirming with a one-time pin. Except the OTP would *not* be directly texted to his phone... his phone will be texted a bitstring S of 0s and 1s, and the OTP is *actually* the bitstring that is produced after performing the following process n times:

- Let T be S with “`11`” appended at the beginning *and* at the back.
- Let $U = U_1U_2 \dots U_{|S|+2}$ be the string of length $|S| + 2$ such that $U_i = f(T_i, T_{i+1}, T_{i+2})$ for every i, where $f(x, y, z)$ is defined as `1` if $x = y = z$ or $x = y = 1$, and `0` otherwise.
- Replace S by U.

For example, starting with S = `000101`, the process produces S = `10100001` after one iteration, and S = `1100011001` after one more iteration. Note that each iteration increases the length of S by 2.

Mu Liu received the text containing the initial bitstring S as follows:

```
1000010101111100111110100111001000110101010101111010101101100000011010111110
```

What is the OTP?

Since the OTP can be very long, output it in the following format: First interpret the bitstring as a **decimal** number, then reduce it modulo $10^9 + 7$.

Input Format

The first line of input contains t, the number of test cases.

Each test case consists of a single line containing the integer n.

[1]P.L.D.T. of course is a telco company founded by renowned computer scientists Papadimitriou, Lovelace, Dijkstra, and... one more guy, whose name escapes me at the moment

Output Format

For each test case, determine whether you choose to answer it or not.

If you choose to answer the test case, output one line containing the answer.

If you choose to not answer it, output one line containing the string `PASS`.

Constraints

- $t = 50$
- $1 \leq n \leq 6250000$

Scoring

This problem has an unusual scoring system.

You may choose to not answer some test cases. The output format addresses how to do this. If any of the test cases you answered is wrong/invalid, you get a score of 0 for the whole problem. Otherwise, your score is equal to the number of test cases you answered multiplied by two. Note that there are $t = 50$ cases, so you get 100 points by answering all test cases.

Test Data

This problem only has one test file, `input.txt`, which you can download from CMS.

Sample I/O

Input	Output
2 5 100	671347745 549439374

Explanation

Note that the sample I/O doesn't follow the constraints.

For the first test case ($n = 5$), the OTP is

```
111111001100010000000011000010000110110011010011111000111001000111011100011000000011100
```

Interpreting this bitstring as a decimal number and reducing it modulo $10^9 + 7$ gives 671347745.

Problem D
Snow

The woods are lovely, dark and deep,

But I have promises to keep,

And miles to go before I sleep,

And miles to go before I sleep.

One overcast evening, Robby Frosty and Nascar Wilde found themselves venturing out into the Twilight Taiga, needing a break from their daily grind of programming and mathematics, and desiring to lose themselves in the beauty of nature for just a few fleeting moments of solace.

It was snowing.

They had mapped out this forest long ago. The taiga can be partitioned into a grid with n rows and n columns, with a clearing in each cell of the grid. The top-left corner is their favorite place to meditate, because it is physically the farthest away from their workplace, which is located by the clearing at the bottom-right corner. A two-way road exists that connects each clearing with the clearings directly to the north, south, east, and west of it.

Each clearing also has a certain *difficulty level* associated with it—a value of 0 means there is nothing special about this clearing, a positive value means that travel through this clearing is particularly expedient, and a negative value means that the clearing is littered with obstacles that are difficult to pass.

When they are done meditating, they know that they will have to return and finish the work that they have been putting off all week. Thus, they promise themselves that they will just traverse through *one last path* before leaving the forest and heading home. But because they have different tastes, they once again find themselves diverging in this snowy wood.

Robby likes to go nice and slow, so he wants to take the longest path he can find. On the other hand, Nascar Wilde finds the most enjoyment out of going fast, so *he* wants to take the shortest path he can find. The amount of time it would take to traverse a path can be measured using the difficulty levels of each clearing along that path.

We define a **path** to be a sequence of *distinct* clearings (beginning at the top-left corner and ending at the bottom-right corner) such that a road exists that connects any two consecutive clearings in the sequence (i.e., a "valid travel plan" from the top-left to the bottom-right). Note that the path is permitted to be winding, so long as it never crosses over itself by visiting a clearing more than once (knowing how way leads on to way, Robby doubted if he should ever come back). The difficulty level of an entire path is equal to the sum of the difficulty levels of all clearings along that path, where a higher difficulty level means that it takes longer to traverse.

The *novelty number* of their paths is equal to the difference between the difficulty level of Robby's path—which he wants to maximize—and the difficulty level of Nascar's path—which he wants to minimize. Help design the paths that each of them should take so that this novelty number is as large as possible. The higher the novelty number, the higher your score.

Input Format

The first line of input contains the integer n. After that, n lines follow, each containing n integers. The jth integer in the ith line contains $\ell_{i,j}$, the difficulty level of the clearing in the ith row from the north to south and the jth column from west to east.

Output Format

Output two lines, each containing a string. The first line describes Robby's path, while the second line describes Nascar's path.

Each path description is a nonempty string with characters `N`, `E`, `W`, `S` which describes the path starting at the northwesternmost clearing to the southeasternmost clearing. The ith character represents the direction to the next clearing. It is

- `N` if the $(i+1)$th clearing is to the north of the ith clearing,
- `E` if the $(i+1)$th clearing is to the east of the ith clearing,
- `W` if the $(i+1)$th clearing is to the west of the ith clearing,
- `S` if the $(i+1)$th clearing is to the south of the ith clearing.

Each path must be *valid*—that is, it must start at the northwesternmost clearing and end at the southeasternmost clearing, must not visit the same clearing twice, and must not go off the grid.

Constraints

- $n \in \{60, 70, 80, 90, 100\}$
- $|\ell_{i,j}| \leq 10^{10}$

Scoring

This is an output-only problem. You will be provided with five test files, each containing a test case. Each test file follows the file name format of `input_X.txt`, and for each, you submit to it by uploading a file `output_X.txt` containing your answer.

Each test file is worth up to 20 points. For each test file, let d_{Robby} and d_{Nascar} be the difficulty level of Robby's and Nascar's paths, respectively, let $d = d_{\text{Robby}} - d_{\text{Nascar}}$ be the novelty number, and $d_{\max}$ be the largest d among the setters, testers, and *contestants*' solutions. Then your score is computed as follows:

$$\text{score} = \begin{cases} 0 & \text{if } d \le 0, \\ 20 \cdot \frac{d}{d_{\max}} & \text{if } 0 < d < d_{\max}, \\ 20 & \text{if } d_{\max} \le d \end{cases}$$

rounded *down* to 3 decimal places.

Since $d_{\max}$ can only be obtained after the contest, all submissions will be rejudged after the contest once this value can be obtained. During the contest period, $d_{\max}$ will be set to the maximum value of d in all solutions by the setter and testers.

Test Data

This problem only has five test files: `input_X.txt` where `X` goes from `0` to `4`. You can get these files by downloading and extracting `snow_inputs.zip` from CMS.

Each test file has a different n value and was generated *randomly*: each $\ell_{i,j}$ value was uniformly randomly chosen between -10^{10} and 10^{10} independently of each other.

Sample I/O

Input	Output

4
-100 -50 60 1000
100 40 -50 100
100 0 50 100
-1000 -150 -50 -150
	SSEENWNEESSS
	EESSWSEE

Explanation

Note that the sample input doesn't follow the constraints.

In the sample output, we have

- $d_{\text{Robby}} = (-100) + 100 + 100 + 0 + 50 + (-50) + 40 + (-50) + 60 + 1000 + 100 + 100 + (-150) = 1200$.
- $d_{\text{Nascar}} = (-100) + (-50) + 60 + (-50) + 50 + 0 + (-150) + (-50) + (-150) = -440$.

Hence, $d = 1200 - (-440) = 1640$.

Problem E
Stronghold

This is **not** an interactive problem.

Nintend-O International (Phenomenal!) has released a new entry in their line of puzzle video games that feature playable versions of competitive programming problems. Their latest game, *Stronghold*, is a medieval-fantasy puzzle game with the following gameplay:

> The game begins with n knights and n horses with $n = 700$.
>
> - Each knight is assigned a hidden distinct positive integer corresponding to their strength value.
> - The horses, meanwhile, are labeled 1 to n, arranged from least to most desirable (where 1 is worst and n is best)
>
> A duel can be organized by providing a list with 2 or 3 horses, and then making the following announcement: "The knights riding the horses in this list must duel."[2] These knights will clash! The knight with the highest strength value wins, and as their prize, they claim for themselves the most desirable horse (among those that participated in this duel). The knight with the second-highest strength value similarly claims, as prize, the second-most desirable horse, and if there were a third participant in the duel, then they would be left with the third-most desirable horse.
>
> If a knight leaves a duel with a less desirable horse than the one they had going into it, then they feel sad `:((` for the rest of the day.
>
> You can organize any number of duels per day, but all duels for each day happen simultaneously in one long cutscene at day's end—so no horse can participate in more than one duel per day.
>
> An assignment of knights to horses is called *stable* if, from this day forward, it would be impossible for a knight to ever be sad due to having their horse taken from them, regardless of what duels are organized on each day. The goal of the game is to organize a series of duels (possibly over several days) such that once all duels are done, the knights are in a stable assignment.

However, you (one of the best competitive programmers that the Philippines has to offer) find this game boring. "It's so standard!" you say. You think it needs an extra self-imposed challenge in order to be exciting again!

[2] We should probably call it a *truel* if there are 3 of them.

Hearing your comments, Kevin dares you to perform a "Blind and Deaf" challenge speedrun of this game, meaning you have the following restrictions:

- You do not know the strength values of the knights
- You are not given any feedback as to results of any previous duels you had organized (after all, *this is not an interactive problem*)

Essentially, then, the task is to find some sequence of button inputs (i.e., which duels to host on each day) that **always** successfully completes the task, regardless of what happens. To add to the spice, let's make this a speedrun—you must minimize the total number of days that this plan requires.

Output Format

This is **not** an interactive problem. **This is an output-only problem.** You need to submit a file named `output_0.txt` describing the day-by-day duel plans. There is no input file.

If you have a plan consisting of d days, then your submission must contain d lines, the ith of which describes the duels that will be held on the ith day. Each duel is described by 2 integers h_1, h_2 or 3 integers h_1, h_2, h_3 on the line, separated by spaces. These denote the labels of the horses for that duel.

Different duels occurring on the same day must be separated by the forward slash '`/`' character. Since a horse can only participate in at most one duel per day, each integer from 1 to n must appear at most once in each line.

Scoring

Your submission will be evaluated by running it against $t = 12000$ different initial assignments of knights to horses. However, this checking process may be slow and may clog the judging queue, so during the contest period, only the first 1000 will be used. After the contest, all submissions will be reevaluated using all 12000 assignments.

Your score depends on d, the number of days in your plan (which is also the number of lines in your submission). If $d_{\min}$ is the minimum value of d in all valid plans by the setter, testers, and *contestants*, then your score is computed as follows:

$$\text{score} = \begin{cases} 0 & \text{if } d > 2000, \\ 50 - d/40 & \text{if } 64 \le d \le 2000 \\ 55 + 45\left(\frac{72-d}{72-d_{\min}}\right)^2 & \text{if } d_{\min} < d < 64 \\ 100 & \text{if } d \le d_{\min}. \end{cases}$$

rounded down to 3 decimal places.

Since $d_{\min}$ can only be obtained after the contest, all submissions will be rejudged after the contest once this value can be obtained. During the contest period, $d_{\min}$ will be set to the minimum value of d in all valid plans by the setter and testers.

The list of all 12000 initial assignments that will be used for evaluation is in the file "`input_full.txt`" and can be downloaded from CMS. (The first line is $n = 700$ and the second line is $t = 12000$.)

Sample I/O

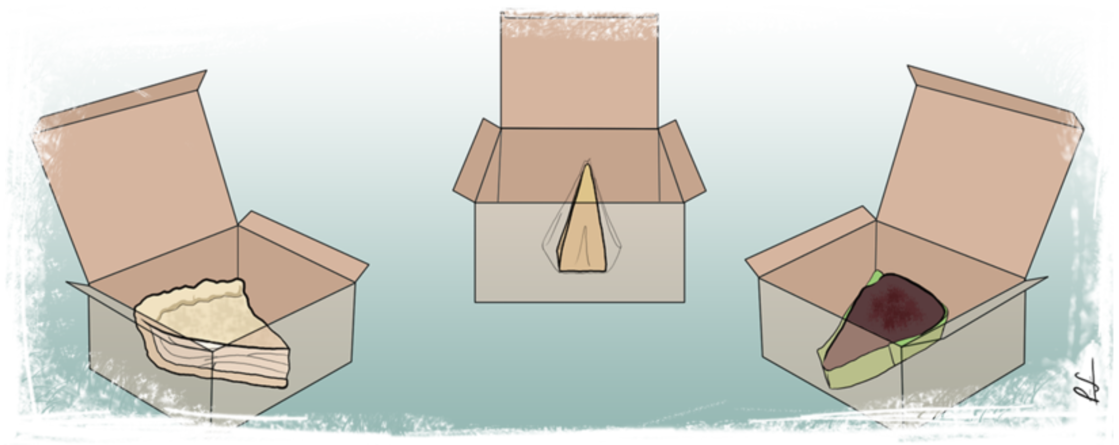
Output

```
3 4 / 1 5
3 2 1
1 2 / 4 5
1 5 4 / 3 2
3 4
1 2 3 / 4 5
1 3 5
```

Explanation

The sample output plan consists of $d = 7$ days of duels. It doesn't solve the problem for $n = 700$, but you can check that it solves it for $n = 5$ regardless of the initial assignment of knights to horses.

Problem F
Sugar Roll



You are setting up a game involving a row of n boxes. The game goes as follows:

- The player will touch two boxes with both his hands. These two boxes must have an odd number of boxes between them.
- Both hands of the player will move towards each other simultaneously, one box at a time.
- When the hands of the player are in the same box he stops.
- The player opens the two boxes he touched in the beginning and the box where both his hands stop.
- The player takes home the contents of the three opened boxes.

What could the boxes contain? Food! Each of the boxes should contain exactly one of the following Filipino desserts—buko pie, hopia, yema, and kalamay. You must choose the content of each box such that any player will be able to take home at least two different desserts, no matter which three boxes he ends up opening.

There might be multiple ways to do this. All you have to do is to output one. Your score is based on how delicious the contents of the boxes are. Each type of dessert has points associated with it.

- Buko Pie is **3 points** because *Pie is almost THREE.*
- Yema is **1 point** because *Yema ONE and only.*
- Hopia is **2 points** because *Hopia love me TWO.*
- Kalamay is **4 points** because *It's your kalamay FOUR better or FOUR worse.*

Let a_i be the number of points corresponding to the dessert in box i. Your goal is to choose a setup such that $a_1 + \ldots + a_n$ is as high as you can make it. It does not have to be the maximum possible. However, the higher your points, the higher your score for this task.

If it is impossible to do any one of the steps of the game, then the player takes home one yema and one hopia as an apology from the organizers.

Input Format

The first line of input contains t, the number of test cases.

Each test case consists of a single line containing a single integer n.

Output Format

For each test case, determine whether you choose to answer it or not.

If you choose to answer the test case, output one line containing a string of n characters denoting the contents of the boxes. Each letter must be either `B`, `H`, `Y` or `K` denoting Buko, Hopia, Yema or Kalamay, respectively.

If you choose to not answer it, output one line containing the string `PASS`.

Constraints

- $t = 45$
- $1 \leq n \leq 45$
- All ns are distinct.

Scoring

This problem has an unusual scoring system.

You may choose to not answer some test cases. The output format addresses how to do this. If any of the test cases you answered is wrong/invalid, you get a score of 0 for the whole problem.

Let $best_n$ be the absolute maximum sum for a given n. Let sum_n be the sum that you get for a given n. If you chose not to answer the test case, then $sum_n = 0$.

We define $score_n$ to be:

$$score_n = \begin{cases} sum_n \times 2 & \text{if } sum_n \neq best_n \\ sum_n \times 5 & \text{if } sum_n = best_n \end{cases}$$

Your score will then be computed as

$$20 \times \frac{\sum_n score_n}{\sum_n best_n}$$

rounded *down* to 3 decimal places.

In particular, if you print the optimal setup for all n, then you get 100 points.

Sample I/O

Input	Output
2 7 11	YHHKYHY PASS

Explanation

Note that the sample input doesn't follow the constraints for t. Also, note that this is not the optimal solution for $n = 7$.

For this output, we get $sum_7 = 13$ and $sum_{11} = 0$.