University of Texas at Austin - High School CS Contest - 2014

General Instructions:

1. Do the problems in any order you like.

2. Your Java classes must be part of the default package. (DO NOT INLCUDE A PACKAGE STATEMENT AT THE TOP OF THE CLASS.)

3. For problems that read from a data file: When the program is judged, the data file will be in the same directory as your program. Do not include any extraneous path information in your program, just the name of the data file. Data file names are case sensitive. If you do not use the correct file name as listed in the question your program may be judged as an incorrect submission.

4. Submit your source code, the .java file, for judging via the PC^2 software. File names are case sensitive. If you do not use the correct file name for the Java file your submission may be graded as incorrect submission.

5. All problems are worth 60 points. Incorrect submissions receive a deduction of 5 points, but **ONLY** if the problem is ultimately judged correct. Problems may be reworked and resubmitted as many times as you like.

6. There is no extraneous input. All input is exactly as specified in the problem. Integer input will not have leading zeros unless specified by the problem.

7. Your program shall not print extraneous output. Follow the format exactly as given in the problem statement and as shown in example output.

8. The time limit for problems is 120 seconds. If the program has not terminated after 120 seconds on the judge's computer, the program will be judged as an incorrect submission.

Number	Problem Name
1	Collatz
2	Crops
3	Gas
4	Golf
5	Levels
6	Perfect
7	Points
8	Prison
9	Scales
10	Snap
11	Soccer
12	Words

1. Are You Ready for CS371p and CS373?

Program Name: Collatz.java **Input File: collatz.dat**

The Collatz conjecture (aka the Ulam conjecture, aka Kakutani's problem, aka the Thwaites conjecture, aka Hasse's algorithm, aka the Syracuse problem) states:

Take any natural number (meaning any integer greater than or equal to 1).
If the number is 1, stop,
else if the number is even divide it by 2 using integer division,
else if the number is odd multiply it by 3 and add 1,
then repeat the steps above

The conjecture is that no matter what number you start with the algorithm above will eventually reach 1.

Write a program that given a starting number determines and prints the number of steps to reach 1 given the algorithm above.

Input

- The input file contains one integer per line. Each integer is greater than or equal to 1.

Output

For each integer in the input file print out the number of steps to reach 1 following the algorithm described above.

Example Input File

```
1
2
3
4
1234567
2014
7282392
23
113383
8400511
9953129
1048576
2268955
```

Example Output To Screen

```
0
1
7
2
111
94
297
15
247
685
600
20
489
```

2. Hayville? Farm Day? Pocket Story? Farm Harvest?

Program Name: Crops.java Input File: crops.dat

My daughter Isabelle likes to play the game Hay Day. (Truth be told, her dad likes it as well.) Virtual play houses and sand boxes can be addictive.

One of the game functions in Hay Day is growing crops. Each crop requires a single plot of land and a certain amount of time to grow before it is ready to harvest. Each plot produces two units of the planted crop when harvested. Crops can then be used to produce other goods or sold for gold at the farm's road side stand.

Write a program that determine which potential crop from a group yields the most gold per minute of growing time.

Input

- The first line will contain a single integer N that indicates the number of data sets.
- Each data set starts with an integer M that indicates the number of crops in that data set.
- The next 2M lines of the data set are information about the crops in that data set.
- The first line of the crop data is the name of the crop.
- The second line of the crop data consists of 3 integers, x y and z.
- x and y indicate how long it takes for the crop to grow where x is the hours and y is the minutes
- z is the maximum amount of gold 10 units of the crop can be sold for at the road side stand. (Recall each plot of land produces 2 units of a crop.)

Output

For each data set print out DATA SET, followed by the number of the data set, followed by the name of the crop from the data set with the highest gold per minute per plot for the crop. After the name of the crop, print out the amount of gold per minute per plot for that crop truncated to 2 decimal places. Include a leading zero if necessary.(Recall each individual plot produces 2 units of a crop when harvested.)

In other words for each data set print out:

```
DATA SET <NUMBER> <CROP NAME> X.XX
```

If there is a tie in a data set (based on the truncated gold per minute per plot for a crop) print which ever crop appeared first in the data set.

Example Input File

```
4
2
wheat
0 2 7
tomato
6 0 432
3
wheat
0 2 38
tomato
```

```
6 30 523
carrot
0 10 72
4
soybean
0 40 108
chili pepper
4 0 360
lettuce
3 30 1077
wheat
1 1 100
3
carrots
0 10 103
beets
1 10 200
sorghum
0 30 309
```

Example Output To Screen

```
DATA SET 1 wheat 0.70
DATA SET 2 wheat 3.80
DATA SET 3 lettuce 1.02
DATA SET 4 carrots 2.06
```

3. I'm Gassed

Program Name: Gas.java Input File: gas.data

When gasoline prices rise some people will drive far, far away from their neighborhood gas station to get the best price. 20, 30, 40 miles or more out of their way to get gas at a price 10 cents cheaper than their neighborhood store.

But are they really saving money? What about the cost of the gas they burn to get to that gas station in Shangri La with cheap gas? What about the cost of wear and tear on the car driving that extra 80 miles?

Write a program that given a set of gas stations determines which will cost the least given its price for gas and distance from a person's home.

The cost of buying gas from a station includes the price of the gas itself, plus the cost of the gas used, plus the cost of wear and tear on the car driving to and from the station. The cost of the gas used is based on the price of gas at the station you obtain the gas from.

Input

- The first line will contain a single integer N that indicates the number of data sets.
- The first line in each data set consists of 4 numbers: X G M W
  - X is an integer that indicates the number of gas stations in the data set. X >= 1
  - G is a floating point number that indicates the number of gallons the driver will buy at the gas station. G is the same regardless of the distance from the driver's home to a gas station
  - M is a floating point number that indicates the miles per gallon the driver's car uses when driving
  - W is a floating point number that indicates the cost in wear and tear per mile for the driver's car
- The next X lines of the data set are the gas stations. Each line consist of two floating point numbers D P.
- D is the distance in miles from the driver's home to the gas station and P is the price per gallon for gas at the station.

Output

For each data set output the number of the data, a colon, a space and then, the number of the gas station (as it appeared in the data set) with the cheapest total cost for the desired gas. The first gas station in a data set is number 1, the second is number 2 and so forth. Finally print out the lowest total cost for the desired amount of gas from the station rounded to the nearest cent.

Example Input File

```
4
3 15.7 17.2 0.27
3.6 3.75
40.5 2.99
10.0 3.05
1 8.8 25.0 0.22
0.8 3.25
4 96.7 12.4 0.156
```

```
1.7 4.26
125.0 3.75
60.0 4.01
10.0 4.33
1 100.0 20.0 0.25
100.0 4.25
```

Example Output To Screen

```
1: 3 56.83
2: 1 29.16
3: 1 413.64
4: 1 517.50
```

4. A Good Walk Spoiled

Program Name: Golf.java Input File: golf.dat

A very precise golfer is trying to determine the minimum number of times he can a hit a ball and reach a target. The distance to the hole is given as an integer.

The golfer has a choice of clubs. The golfer is very precise and when using a particular club always hits the ball the exact same distance.

The golfer doesn't use traditional names for his clubs. (e.g. driver, 3 wood, 5 iron) Instead he names each club based on the exact distance he can hit the ball with that club. (e.g. 10, 40, 60).

The golfer may use a particular club as many times as he wants. The golfer always hits the ball straight. The golfer NEVER hits the ball any distance to the right or left of the target, always straight at the target. The golfer DOES NOT want to consider scenarios where he hits the ball past the target (overshoots) and then turns around to hit the ball back to the target.

Write a program that determines the minimum number of times the golfer can reach the hole given a set of clubs and a starting distance from the hole. If it is not possible to reach the hole given the set of clubs and the starting distance print out NOT POSSIBLE.

Input

- The first line will contain a single integer N that indicates the number of data sets.
- Each data set will consist of exactly 2 lines.
- The first line of the data set is a series of 1 or more integers, all greater than zero and separated by a single space. These represent the clubs available to the golfer for this data set.
- The second line is a single non negative integer that represents the starting distance from the hole

Output

For each data set print out a single line. Print out the minimum number of strokes the golfer must make to get from the starting point to the hole using the given clubs or NOT POSSIBLE if it is not possible to reach the hole.

Example Input File

```
12
30 10 50
30
2
1
1
2
30 10 50
15
30 10 50
50
30 10 50
70
```

```
40 10 60 200
105
40 10 60 200
190
40 10 60 200
90
15
90
1
15
20 2 3 4 10
30
```

Example Output To Screen

```
1
NOT POSSIBLE
2
NOT POSSIBLE
1
3
NOT POSSIBLE
4
3
6
15
2
```

5. Nine Levels of Java Frustration

Program Name: Levels.java **Input File: NONE**

Write a program that prints out the 9 levels of Java Frustration.

Look at your shirt and the sample output below.

Input

- There is no input file for this question

Output

Output the 9 levels of Java Frustration EXACTLY as shown below. Do not add any extra spaces to the end's lines. The last line ends with a newline character just like every other line. There are no spaces in the output what so ever.

Example Output To Screen

```
************************************************************
*......OVERRIDING.HASHCODE().AND.NOT.OVERRIDING.EQUALS().....*
************************************************************
..********************************************************..
..*..............NO.TRUE.REFERENCE.PARAMETERS.............*..
..********************************************************..
....******************************************************....
....*...........ArrayIndexOutOfBoundsException...........*....
....******************************************************....
......****************************************************......
......*........THE.GREAT.PRIMITIVE-OBJECT.SCHISM........*......
......****************************************************......
........**************************************************........
........*..OVERLOADING.INSTEAD.OF.OVERRIDING.EQUALS()..*........
........**************************************************........
..........************************************************..........
..........*..LENGTH()..LENGTH.OR..SIZE().WOE.IS.US...*..........
..........************************************************..........
............**********************************************............
............*..==.AND..EQUALS()-TEARS.UNNUMBERED...*............
............**********************************************............
..............******************************************..............
..............*...NullPointerException.OF.DOOM...*..............
..............******************************************..............
................****************************************................
................*......ClassCastException......*................
................****************************************................
```

6. Perfect Placement

Program Name: Perfect.java **Input File: perfect.dat**

A group of CS students is looking for the perfect place to meet up after classes. They have divided the campus into a rectangular grid of equal sized cells. Each cell is designated with a row and column number. The upper left cell in the grid is row 0 column 0.

The CS students are in different classes around campus. Each student's location is a cell in the grid.

Determine which cell in the grid is the perfect one to meet in after class. The perfect cell is the one that minimizes the sum of the distances the students must travel from their initial cells to the destination cell.

Students can travel along a single straight line from their initial cell to any other cell in the grid.

Input

- The first line will contain a single integer N that indicates the number of data sets.
- Each data set will consist of exactly 2 lines.
- The first line in a data set contains 2 integers r and c separated by a space. r is the number of rows in the grid and c is the number of columns in the grid. r and c will both be greater than 0.
- The second line in a data set contains the initial coordinates of the students in the data set in the form r1 c1 r2 c2 … rx cx where x is the number of students. There will be an even number of integers in the second line and there will be at least 2 integers. (At least 1 student.)
- All student coordinates will be in bounds: 0 <= rs < r and c <= cs < c where rs and cs are the coordinates for a student and r and c are from the first line of the data set.

Output

For each data set print out the cell in the grid that is the minimum distance from the students' initial cells. Print out the cell in the form r c.

If there is a tie among one or more cells print out the one with the lowest row number. If there a tie for the cell with the lowest row number pick the cell in that row with the lowest column number.

Example Input File

```
5
5 10
0 0 4 3 1 2
10 10
3 7
10 20
0 0 0 0 5 5 9 8 8 9 0 0
5 5
4 4 0 4 4 0 0 0
4 4
2 3 1 2
```

Example Output To Screen

```
1 2
3 7
0 0
2 2
1 2
```

7. Some of Our Favorite Blues: Eden Hazard, Peter Cech, Frank Lampard, John Terry, and Dave

Program Name: Points.java **Input File: points.dat**

Yep, another soccer question.

Olivia and Mike have become rabid Chelsea fans. Chelsea is a soccer team in the English Premier League.

Mike and Olivia want to know how many points their favorite team earns under various scoring scenarios.

Each scenario will contain a list of results (wins, losses, and draws) and the points assigned for each result.

For each scenario print out the points the team earns.

Input

- The first line will contain a single integer N that indicates the number of data sets.
- Each data set has 2 lines
- The first line has 3 integers `W D L`. `W` indicates the number for points earned for a win, `D` the number of points for a draw, and `L` the number of points for a loss. `W >= D >= L`
- The next line of the data set will consist of 1 or more letters with no spaces. Each letter will be a W, D, or L indicating a win, loss, or draw.

Output

For each data set print out the number of points the team earns based on the scoring system and the game results.

Example Input File

```
4
3 1 0
WWDDWDWDLWDLWDLWDLWDL
5 0 -1
WWWWLDLDLDLDLDLWLDWLD
10 3 1
WWWLDLDWLDLWDLW
1 0 0
W
```

Example Output To Screen

```
32
22
77
1
```

8. Who is Number One?

Program Name: Prison.java **Input File: prison.dat**

The Prisoner, shown to the right, is trapped in a devious prison. The Prisoner wants to escape the prison, but also find out who is Number One, the head of a shadowy spy organization.

The Prisoner wakes up in the prison. He wants to know two things. First is it possible to escape the prison and second is it possible to escape the prison while picking up all the clues to the identity of Number One?

The prison consists of 5 kinds of cells.

1. The starting location of the prisoner is marked with a S. The starting cell is also a Red cell as described below.
2. Impassable cells are marked with a *.
3. Green cells are open cells in the prison. Green cells are marked with a G. The devious twist is that when The Prisoner leaves a Green cell it becomes impassable.

   For example, if there are two cells in the prison next to each other GG and the prisoner is on the left most cell. When he moves to the right most G that section of the prison changes from GG to *G. The Prisoner can't go back to the cell he just left.

4. Red cells are the other kind of open cell. Each Red cell contains a clue to the identity of Number One. When The Prisoner enters a Red cell he picks up the clue and the cell changes to a Green cell that behaves as described above.
5. Exit cells are the last kind of cells. They are marked with an E. If The Prisoner can reach an Exit cell he escapes the prison. Exit cells may appear anywhere in the prison and there may be more than one exit cell. When The Prisoner steps on an exit cell he exits the prison. In other words the prisoner cannot move through an exit cell and then back on to it.

The Prisoner can move in the four cardinal directions (north, east, south, west) into open cells (Red and Green) or the exit. The Prisoner cannot move cells beyond the edges of the prison.

Write a program that given a prison determines if The Prisoner can escape the prison (regardless of picking up all the clues) and if Prisoner can escape the prison while picking up all the clues.

Consider the following simple prison:

RGSRE

The Prisoner starts at the middle cell of the prison. It is possible for The Prisoner to escape. He simply has to move to the right two cells:

The first move to the right changes the prison to this: (Recall the starting cell is a Red cell with a clue. When leaving a Red cell it becomes a Green cell. P is shown just to mark the current location of The Prisoner. He is standing on a Red cell.)

RGGPE

One more move to the right and The Prisoner is on an Exit cell:

RGGGE

So, The Prisoner can escape this prison. However The Prisoner cannot collect all the clues from Red cells and escape the prison.

If The Prisoner moves to the left to collect the clue in the far left Red cell the situation is this:

RPGRE

The Prisoner is standing on a Green cell. When he moves another cell to the left the situation becomes:

P*GRE

Moving off of the Green cell changes it to an impassable cell and The Prisoner is trapped.

Input

- The first line will contain a single integer N that indicates the number of data sets.
- The first line of each data set will be a single integer R that indicates the number of rows in the prison for the data set
- The next R lines will contain the characters that define the prison for that data set.
  Each row in a data set will contain the same number of characters.
  The only characters that will appear are S, G, R, E, and *. The meaning of each character is described above.
  There will be exactly one S cell in the data set.
  There will be at least one E cell in the data set.

Output

For each data set print out the number of the data set followed by a space followed by YES if The Prisoner can escape the prison (regardless of picking up all the clues) or NO if The Prisoner cannot escape the prison (with or without all the clues.) Then print a single space and YES if The Prisoner can escape the prison while picking up all the clues, or NO if he cannot.

Example Input File

```
4
1
RGSRE
3
ESRGG
*G**R
GGRGG
4
EGRGGGG***GG
*G***G*GGRGG
```

```
R****G*RRRGS
RGRRGG***GR*
5
EGG****GGGE
GGR****GGRG
**S*G****RG
**RGG***R**
***G*******
```

Example Output To Screen

```
1 YES NO
2 YES YES
3 NO NO
4 YES NO
```

9. Up and Down the Scales

Program Name: Scales.java **Input File: scales.dat**

For this problem we will define a scale word as one where the letters are in ascending order, descending order or both.

In this problem words consist of one or more upper case letters A through Z.

A word is an ascending scale word if each letter in the word is greater than or equal to the previous letter in the word.

A word is a descending scale word if each letter in the word is less than or equal to the previous letter in the word.

A word made up of all the same letter is both ascending and descending

Input

- The first line will contain a single integer N that indicates the number of data sets.
- Each data set will consist of exactly 1 line.
- The data set will be a word as defined above

Output

For each data set print out the number of the data set, a single space, and then A if the word is an ascending scale word, D if the word is a descending scale word, AD if the word is an ascending and descending scale word, or N if the word is neither.

Example Input File

```
4
A
SPOOLED
KNOT
ELEVATOR
```

Example Output To Screen

```
1 AD
2 D
3 A
4 N
```

10. Snap Crackle Pop! aka Son of Fizz Buzz

Program Name: Snap.java Input File: NONE

Have you heard of the Fizz Buzz problem? Some businesses claim they interview Computer Science graduates they can't solve the Fizz Buzz problem. (See http://tinyurl.com/yzdlqx6 after the contest.)

Write a program that prints out the integers 1 to 100 in order, 1 per line with the following changes.

- If a number is divisible by 2 (but not 3 or 5) print out SNAP instead of the integer.
- If a number is divisible by 3 (but not 2 or 5) print out CRACKLE instead of the integer.
- If a number is divisible by 5 (but not 2 or 3) print out POP instead of the integer.
- If a number is divisible by 2 and 3 (but not 5) print out SNAPCRACKLE instead of the integer.
- If a number is divisible by 2 and 5 (but not 3) print out SNAPPOP instead of the integer.
- If a number is divisible by 3 and 5 (but not 2) print out CRACKLEPOP instead of the integer.
- If a number is divisible by 2, 3, and 5 print out SNAPCRACKLEPOP instead of the integer.

Input

- There is no input file for this question

Output

As described in the question. The sample out shows the first and last 10 lines of the expected output. Your program must produce all 100 lines of output correctly. Replace the … shown below with the other 80 lines of output

Example Input File

NONE

Example Output To Screen

```
1
SNAP
CRACKLE
SNAP
POP
SNAPCRACKLE
7
SNAP
CRACKLE
SNAPPOP
…
91
SNAP
CRACKLE
SNAP
POP
SNAPCRACKLE
97
SNAP
CRACKLE
SNAPPOP
```

11. Go Oscar, David Luiz, and Ramires

Program Name: Soccer.java **Input File: soccer.dat**

World Cup soccer is back this summer in Brazil. I will be rooting for the USA, England, Brazil, and Belgium.

Write a program that determines the first place team from a group in the World Cup Soccer tournament. The final tournament of the Soccer World Cup competition consists of group play in a round robin tournament followed by a single elimination tournament.

Groups in the initial round robin tournament are composed of four teams. (Team USA got a very tough draw this time around. They are in a group with Germany, Portugal, and Ghana.)

In the group round robin tournament, each team in a group plays every other team in the group one time for a total of six matches. Matches can result in a win, a loss, or a draw based on the number of goals scored. If a team scores more goals in a match, it wins the match and the other team loses the match. If each team scores the same number of goals in a match, the result is a draw. A win is worth 3 points for the winning team, a draw is worth 1 point for each team, and a loss is worth 0 points for the losing team.

The first place team is the one with the most points in the group after the round robin tournament. If two or more teams are tied for the most points, a series of tiebreakers is used.

1. If two or more teams are tied for most points, the first tiebreaker is largest goal difference. (A team's goal difference is the sum of the goals the team scored in the group matches minus the sum of the goals the team allowed in the group matches.) The team with the largest positive goal difference among the teams tied on points is the first place.
2. If two or more teams are tied for most points and largest goal difference, the second tiebreaker is the total number of goals a team scored in the group matches. The team with the most goals scored from the teams tied with points and tied for goal difference is the first place team.
3. If there are exactly two teams tied for most points, largest goal difference, and most goals scored, the third tiebreaker goes to the team that won the head to head match between the two teams that are tied.
4. If there are three or four teams tied for most points, largest goal difference, and most goals scored, OR if there are exactly two teams tied for most points, largest goal difference, and most goals scored and there head to head match was a draw THEN first place is unresolved. (There are actually more elaborate tie breakers used, but we are not using them in this problem.)

Write a program that given the results of the 6 matches in a round robin group tournament prints out the first place team in the group.

Input

- The first line will be a single integer N that indicates the number of data sets.
- Each data set will consist of six lines with the format:
  `TEAMNAME1 GOALS1 TEAMNAME2 GOALS2`

- Each line in a data set is the result of the match between `TEAMNAME1` and `TEAMNAME2`. `GOALS1` is the number of goals scored by `TEAMNAME1` in the match and `GOALS2` is the number of goals scored by `TEAMNAME2` in the match.
- Team names will consist of uppercase letters only.
- `GOALS1` and `GOALS2` will be non-negative integers.
- A data set will contain four distinct team names and each team will play each other team once.

Output

- For each data set output one line.
- Each line of output will be `data set <N>: <WINNINGTEAM>` where `<N>` is the number of the data set and `<WINNINGTEAM>` is the first place team from the group based on the criteria explained above.
- If there is a tie that cannot be broken, print `UNRESOLVED` in place of `<WINNINGTEAM>`.

Example Input File

```
5
USA 1 ENGLAND 1
ENGLAND 1 SOUTHAFRICA 1
USA 1 IRAN 1
SOUTHAFRICA 1 IRAN 1
ENGLAND 1 IRAN 1
SOUTHAFRICA 1 USA 1
BRAZIL 3 SPAIN 1
IRAQ 5 FRANCE 2
FRANCE 1 BRAZIL 3
BRAZIL 3 IRAQ 0
SPAIN 2 IRAQ 2
SPAIN 4 FRANCE 4
SOUTHAFRICA 5 MEXICO 4
URAGUAY 3 FRANCE 0
SOUTHAFRICA 2 URAGUAY 2
FRANCE 0 MEXICO 2
MEXICO 0 URAGUAY 1
FRANCE 2 SOUTHAFRICA 4
ARGENTINA 1 NIGERIA 0
ENGLAND 1 SWEDEN 1
SWEDEN 2 NIGERIA 1
ARGENTINA 0 ENGLAND 1
SWEDEN 1 ARGENTINA 1
NIGERIA 0 ENGLAND 0
ITALY 5 SOUTHKOREA 4
ITALY 2 RUSSIA 0
SOUTHKOREA 2 RUSSIA 0
ITALY 1 IRAN 2
SOUTHKOREA 2 IRAN 1
IRAN 0 RUSSIA 0
```

Example Output To Screen

```
data set 1: UNRESOLVED
data set 2: BRAZIL
data set 3: URAGUAY
data set 4: SWEDEN
data set 5: ITALY
```

12. Good Words

Program Name: Words.java **Input File: words.dat**

This questions deals with "words". A word is defined as 1 or more lower case English letters.

For this question a word is "good" if it meets all the following criteria:

- It only consists of the first N letters of the alphabet.
  So if N is 4 good words only contain letters a, b, c, or d.
- A given letter cannot be repeated **exactly** X times in a row within the word.
  So if N is 4 and X is 3 aabbaa is a good word but aabbbaa is not due to the 3 b's in a row. aabbbbaa is a good word because there are 4 b's in a row, not exactly 3.
- There must be at least Y distinct letters in the word. Y <= N.
  So if N is 4, X is 3, and Y is 3 abbaac is a good word (3 distinct letters, a, b, c) but aabbaa is not (2 distinct letters, a, b)

Input

- Each data set consists of a single line
- The line will contain 4 integers and the word in the form N X Y word
- The integers are described above
- 1 <= N <= 26, X >= 2, Y >= 0 and Y <= N
- each word will have at least one letter

Output

For each data set print out on a single line GOOD if the word is a good word based on the criteria above or NOT GOOD if it is not a good word.

Example Input File

```
6 2 3 abcdem
6 2 3 bdcdead
5 3 1 aabbaa
5 3 1 aabbbaa
4 3 3 aabbaac
4 3 3 aabbaa
4 3 3 aabbaacc
4 3 3 aabbaaaacc
```

Example Output To Screen

```
NOT GOOD
GOOD
GOOD
NOT GOOD
GOOD
NOT GOOD
GOOD
GOOD
```