Irish Collegiate Programming Competition 2019

Problem Set

University College Cork ACM Student Chapter

March 9, 2019

Happy International Women's Day!

Contributors

The UCC ACM student chapter would like to warmly thank the following people.

For leading the question writing:

- Dr Bastien Pietropaoli, Insight Centre for Data Analytics, UCC, Cork
- Dr Milan De Cauwer, Insight Centre for Data Analytics, UCC, Cork

For contributing:

- Andrea Visentin, Insight Centre for Data Analytics, UCC, Cork
- Dr Eduardo Vyhmeister, Insight Centre for Data Analytics, UCC, Cork
- Dr Guillaume Escamocher, Insight Centre for Data Analytics, UCC, Cork
- Raffaele Baldassini, UCC, Cork

1 Pandora's Preposterous Polygon Project (CPU:1sec - RAM:256MB)

Pandora likes abstract art and more particularly abstract geometry art. For her last year art project, she decided to go crazy with polygons. She would like to name them based on their general shape and their surface area but she realised she has no clue on how to measure their surface area. She would like you to write a program capable of computing the surface area of any polygon.

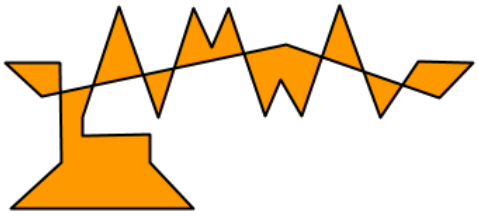


Figure 1: One of the latest "creations" of Pandora, named "Crippling Lamp".

One way to represent polygons is by using a sequence of coordinates for their vertices. Each vertex (i.e. point) is connected to the following one by a straight line. The last vertex is linked to the first one to close the polygon. For instance, a simple square can be described using the following coordinates: $(0,0),(0,1),(1,1),(1,0)$.

Pandora will be providing the vertices, it's up to you to compute the surface area of each one of the polygons.

Input The first input line contains the number of vertices N, with $3 \leq N \leq 20$, composing the polygon. The following lines are the vertices, one per line, provided as two space-separated floating point values (x,y), with $0 \leq x,y \leq 1$, with a precision of up to 16 digits.

The provided polygons may have intersecting edges but will NOT have self-overlapping surfaces.

Output The output line should be a single floating point value corresponding to the surface area of the polygon provided as input. The maximum error accepted is 0.01 (in absolute terms).

Sample Input 1

```
4
0.0 0.0
1.0 0.0
1.0 1.0
0.0 1.0
```

Sample Output 1

```
1.0
```

Sample Input 2

```
3
0.0 0.0
1.0 0.0
0.0 1.0
```

Sample Output 2

```
0.5
```

2 Caroline's Cheap Carpets **(CPU:1sec - RAM:256MB)**

Caroline knows that she is not in the game of mathematical research for the money. Her work, and passion, in the math department in University College Babylon deals with highly recursive mathematical structures.

To be able to afford rent in downtown Babylon, as a side job, she is knitting and selling her own interpretation on what Persian carpets should really be. Bringing a hobby together with her passion, she designs carpets following a clear procedure elegantly written as a slightly broken Haiku in the Book of Recursive Poetry, Al-Rhymus, 932–1002.

"Get from one black square,
nine equal squares. Center is white.
Repeat with remaining."

When starting a new carpet, Caroline first decides on its dimension D. From the dimension, she computes the carpets final size, a square of $3^D * 3^D$ points that need to be knitted. She then picks the order O applied to the carpet. As illustrated below, the order determines how many times the recursive procedure is to be applied to the carpet before having the final design.

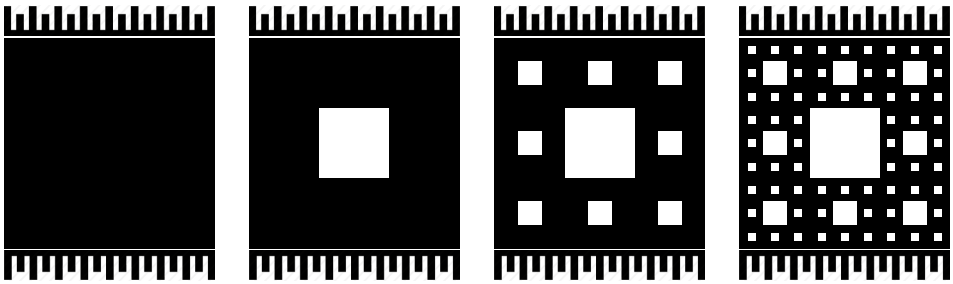


Figure 2: Carpet drawn for orders 0 to 3 for some arbitrary dimension $D \geq 3$.

Unfortunately, Caroline is fairly forgetful and quite often loses track of whether the point that she has to knit is white or black. Help Caroline out by implementing a program that takes in (x, y) positions on the carpet and returns whether these positions should be of black or white color.

Input The first line contains three space-separated integers, $1 \leq D \leq 50$, $0 \leq O \leq D$, $1 \leq N \leq 100000$ respectively the dimension of the carpet, its order, and the number of points that your program should be checking for.

The next N lines contain two space-separated integers $0 \leq x_i < 3^D$ and $0 \leq y_i < 3^D$ providing the position of the i-th point.

Output The output should be a single line consisting of N space-separated characters in $\{b, w\}$ followed by a newline character.

Sample Input 1

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

Sample Output 1

```
b b w
```

Sample Input 2

```
2 2 6
4 7
5 6
8 0
6 0
8 6
1 0
```

Sample Output 2

```
w b b b b b
```

3 Laoise's Ludicrous Locutions (CPU:1sec - RAM:256MB)

Laoise is a clever girl. She claims that she can immediately know whether a text is an anagram of a palindrome. Challenge accepted, you say. Unfortunately you actually don't know what an anagram is, nor what a palindrome is for that matter. But fear not for definitions shall appear before you.
Anagram: *"A word, phrase, or name formed by rearranging the letters of another, such as 'spar', formed from 'rasp'."*
Palindrome: *"A word, phrase, or sequence that reads the same backwards as forwards, e.g. 'madam' or 'nurses run'."*

Praise your friend Google for those definitions. Now that you know what Laoise is talking about, you plan on writing a program that will tell you, faster than Laoise ever could, if a piece of text is an anagram of a palindrome. Laoise tells you that your program won't work because spaces, punctuation, and capitalisation should not be considered when checking if two texts are anagrams or if a text is a palindrome! Challenge accepted, you shout again. You think you got her since Laoise never said the anagrams or the palindromes had to make sense!

Input The first input line contains a single integer N, $4 \leq N \leq 5000$, providing you with the number of pieces of text you will have to check. The following lines contain the N pieces of text you need to check. Each piece of text will contain a maximum of 1000 characters.

Output The output should consist of a single line containing N space-separated integers, only 0s and 1s. The i-th integer should be:

- 0 if the i-th piece of text IS NOT an anagram of a palindrome.
- 1 if the i-th piece of text IS an anagram of a palindrome.

Sample Input 1

```
4
Run, nurses!
Whatever works for Lucy.
I love tacos, I love tacos!
Tacocat forever!
```

Sample Output 1

```
1 0 1 0
```

Sample Input 2

```
6
AAAaaaABbbbddDdssss
sofhqgohqhqohg
Not so clever now, are you?
Animosity is no amity.
Election results. Lies! Let's recount!
A decimal point, I am a dot in place.
```

Sample Output 2

```
1 0 0 1 1 1
```

4 Aoife's Awesome ASCII Art

(CPU:1sec - RAM:256MB)

Aoife is a very resourceful computer science student here in UCC. Her bachelor's thesis discusses a revolutionary method to decorate string sentences to be displayed on screen. Here are the specifications that she developed in her thesis.

The awesome string decoration method can be applied to sentences composed of an arbitrary number of words. Words are defined as a string of alphanumeric characters, and are separated by a white space. A sentence may contain commas followed by a space as per normal typography rules and they are ended with a period, a question mark, or an exclamation mark.

The string decoration method takes as an input a sentence to be decorated with a frame of '$*$' characters according to the following rules:

- The sentence will be broken down in several lines no longer than the longest word contained in it plus the decorating window and its padding.
- A word, in this case, is any continuous string of characters. Continuity being broken **only** by whitespace, and punctuation, such as commas, should be considered as part of a word.
- The first line and the last line are made out of the string '* ' (star and space) repeated as many times as necessary. Care should be taken to ensure that no two '*'s are placed next to each other, nor two blank spaces.
- Each line in between starts with '$*$', contains as many words as possible, and ends with '$*$'. The '$*$' ending the line should be separated from the text of the line by at least one blank space.

On page 542 of her thesis, Aoife provides us with two awesome examples of the capabilities of her method. These examples are reproduced in the input/output samples below.

Input The first input line contains two integers $2 \leq C \leq 1000$ and , $1 \leq W \leq 1000$, providing you with the number of characters and the number of words forming the sentence that should be decorated. The following line contains the W space-separated words.

Output The output should consist of the input string decorated according to Aoife's instructions.

Sample Input 1

```
23 4
Welcome to IrlCPC 2019!
```

Sample Output 1

```
* * * * * *
* Welcome *
* to      *
* IrlCPC  *
* 2019!   *
* * * * * *
```

Sample Input 2

```
36 8
Baking is both an art and a science.
```

Sample Output 2

```
* * * * * * *
* Baking    *
* is both   *
* an art    *
* and a     *
* science.  *
* * * * * * *
```

5 Kiera's Kooky Knights

(CPU:1sec - RAM:256MB)

Kiera has always found fascinating how knights move in chess. Their L-shaped movements look like a dance in her eyes! She imagines herself riding a horse around the chessboard eating all the other pieces! But her imagination goes far beyond a standard chessboard. She dreams of unfathomably large boards... in four dimensions! And with her horse she is running around in this strange L-shaped gallop.

Kiera sees the enemy king on the horizon and wants to catch him as fast as she can. Can you help her discover how many moves she needs to catch him?

The chessboard is a 4-dimensional board (a hypercube). The 4-dimensional knight moves 2 spaces in one dimension and then 1 space in a different one, or 1 space in one dimension and then 2 spaces in a different one.

Input The first line contains a single integer N indicating the size of the chessboard, $8 \leq N \leq 4294967297$, generating a hypercube board of $N \times N \times N \times N$ cells. The second line contains four space-separated integers x_1, x_2, x_3, x_4 representing the initial position of the knight. The third line contains four space-separated integers y_1, y_2, y_3, y_4 representing the position of the king. All coordinates respect $0 \leq x_i, y_i \leq 2^{32}$.

Output The output should consist of a single integer corresponding to the number of moves the knight has to make to reach the king.

Sample Input 1

```
8
0 0 0 0
2 1 0 0
```

Sample Output 1

```
1
```

Sample Input 2

```
8
4 2 4 2
6 3 0 0
```

Sample Output 2

```
3
```

6 Stella's Stellar Sightseeing Suggestions (CPU:5sec - RAM:512MB)

Stella is planning her space holidays. Since she does not mind taking the TGV (Transluminic Gadolinium-fueled Vehicle™) to roam from one star to the others in the same cluster, she looks for a place with the best sightseeing opportunities. Planning her trip, she decides to go for the most interesting star first and then have a look at the accessible neighbourhood.

Unfortunately, galaxy cartographers are not particularly good at making maps anymore. In particular, given the high number of colonised stellar systems, they tend to focus their interest on the main centres of power. Hence, Stella would like to use the raw mapping data she found on the Interstellar Dark Web (open data is illegal those days). The raw data simply consists of the stellar objects' names and their coordinates compared to Sagittarius A* (the centre of the Milky Way).

Given a place's name, Stella would like to find the closest, easily reachable locations. Being an edgy developer of the future, you don't mind handling illegal data. You put on your mandatory black hat (to let people know you are currently committing crime) and start working on the illegal data.

Figure 3: Poster from NASA.

Input The input consists of the following lines:

- A line containing three integers: N indicating how many stellar objects are in the dataset, M indicating how many objects Stella will be checking ($M \leq N$), and k indicating the number of neighbours considered for each object.
- Followed by N lines, each one consisting of the name of an object in double quotes (e.g. "New Terra") followed by three floating point numbers, x, y, z, corresponding to the 3D coordinates of the object. Values are separated by spaces.
- Followed by M lines, each one consisting of the name of a single object in double quotes.

Each dataset will contain the coordinates of up to 500000 objects. Stella will query up to 10 nearest neighbours of up to 40000 different locations (she is being unreasonable but hey, you do what she tells you to do). The 3D coordinates will be provided in light-years (Ly), with $-50000 \leq x, y \leq 50000$ and $-1000 \leq z \leq 1000$. Each coordinate will be precise up to 16 digits.

Output The output will consist of M lines corresponding to the M queries made by Stella. The result of the query should appear in the same order as the queries. Each line will contain the following:

- The name of the location in double quotes.
- Followed by a colon and a space.
- Followed by a list of k location names in double quotes, separated by a comma and a space. The names should be ordered in increasing order of Euclidian distance (i.e. the closest appearing first in the list). In case of a tie, the alphabetical order should be considered.

Sample Input 1

```
6 2 2
"Arcturus" 25520.0 -100.0 -55.0
"Aurora" 25632.0 8.0 7.0
"Earth" 25640.0 0.0 0.0
"Neotrantor" -4128.0 390.0 -58.0
"Terminus" 38938.0 -36441.0 387.0
"Trantor" 3682.0 488.0 112.0
"Earth"
"Trantor"
```

Sample Output 1

```
"Earth": "Aurora", "Arcturus"
"Trantor": "Neotrantor", "Arcturus"
```

Sample Input 2

```
10 4 2
"Shinrarta Dezhra" 25612.84 55.72 17.59
"Deciat" 25592.72 122.63 -0.81
"Sol" 25640 0 0
"Maia" 25983.38 -81.78 -149.44
"Wolf 397" 25650.41 40.0 79.22
"Leesti" 25579.75 72.75 48.75
"Colonia" 5832 -9530.5 -910.28
"Eurybia" 25670.5 51.41 -54.41
"Lave" 25569.25 75.75 48.75
"LHS 3447" 25583.84 -48.18 -5.28
"Sol"
"Deciat"
"Leesti"
"Colonia"
```

Sample Output 2

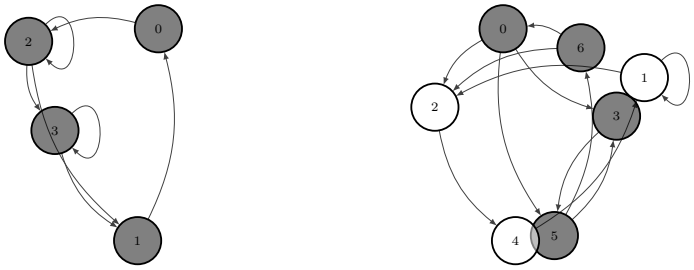
```
"Sol": "Shinrarta Dezhra", "LHS 3447"
"Deciat": "Leesti", "Lave"
"Leesti": "Lave", "Shinrarta Dezhra"
"Colonia": "LHS 3447", "Lave"
```

7 Janine's Jolly Jumper (CPU:1sec - RAM:256MB)

"We've sent number 137 more than an hour ago... I don't think they'll make it back... There must be a problem!" - said the show jumping organiser to Janine.

On week days, Janine is employed as a skillful researcher rather busy developing bleeding edge science to save humanity[1]. But on weekends, there is no place she would rather be than on the back of her faithful horse: Jolly Jumper. On this sunny Sunday, Janine is competing as rider 138 in a rather unusual show jumping event.

The event features a number of courses that the rider needs to clear. A course is a set of obstacles over which a sequence is defined. Such a sequence is called a route and is ending where it started. Right after each obstacle in a route, a sign displays one or more numbers referring to the next obstacle the rider should be aiming for. An example of such a course is illustrated on the leftmost side of the figure below. A possible route on that course is 0, 2, 1, 0, 2, 3, 3, 1, 0.



Unfortunately, Jake, in charge of setting the routes for the event, was out in the pubs last night and was not in the best shape to set the signs up early this morning. Although, he did set each individual course correctly, he did accidentally link arbitrary courses together. Quite fortunately, he never created loops between any two courses. The black and white courses on the rightmost side of the figure above are illustrating the situation as Jake did put a sign for obstacle 2 after obstacles 0 and 6.

Help Janine to save the event by finding out which obstacles belong together within the same course.

Input On the first line, you are provided with an integer number $2 \leq N \leq 20000$ of obstacles used during the event. The N following lines are featuring first the index $0 \leq i \leq N$ of the obstacle followed by up to N space-separated integers providing you with all the indices of obstacles signed from obstacle i.

Output The output of your program should consist of one or more lines. On each line you should list all obstacles belonging to the same course in increasing order of their indices. Courses (lines) should be ordered by increasing order of their smallest obstacle index.

Sample Input 1

```
4
2 3 2 1
3 1 3
1 0
0 2
```

Sample Output 1

```
0 1 2 3
```

Sample Input 2

```
7
6 0 2
0 3 5 2
3 5
5 6 3
2 4
4 1
1 2 1
```

Sample Output 2

```
0 3 5 6
1 2 4
```

[1]Coming up with a cure for cancer keeps a lot of scientists busy.

8 Shirley's Surprisingly Simple Song[2] (CPU:1sec - RAM:256MB)

Shirley!
Shirley, Shirley, bo-bhirley,
bo-na-na fanna, fo-fhirley,
fee fi mo-mhirley, Shirley!
–
Lincoln!
Lincoln, Lincoln, bo-bincoln
bo-na-na fanna, fo-fincoln
fee fi mo-mincoln, Lincoln!
–
Come on ev'rybody,
I say now let's play a game
I betcha I can make a rhyme
out of anybody's name
The first letter of the name
I treat it like it wasn't there
But a "B" or an "F"
or an "M" will appear
And then I say "Bo" add a "B"
then I say the name
Then "Bo-na-na fanna" and "fo"
And then I say the name again
with an "f" very plain
Then "fee fi" and a "mo"
And then I say the name again
with an "M" this time
And there isn't any name
that I can't rhyme
–
Arnold!
Arnold, Arnold, bo-brnold
bo-na-na fanna, fo-frnold
fee fi mo-mrmold, Arnold!
–
But if the first two letters
are ever the same
Crop them both, then say the name
Like Bob, Bob, drop the "B's", Bo-ob
Or Fred, Fred, drop the "F's", Fo-red
Or Mary, Mary, drop the "M's", Mo-ary
That's the only rule that is contrary
And then I say "Bo" add a "B"
then I say the name
Then "Bo-na-na fanna" and "fo"
And then I say the name again
with an "f" very plain
Then "fee fi" and a "mo"
And then I say the name again
with an "M" this time
And there isn't any name
that I can't rhyme
–
Say Tony,
Tony, Tony, bo-bony
bo-na-na fanna, fo-fony
fee fi mo-mony, Tony!
–
Let's do Billy!
Billy, Billy, bo-illy,
bo-na-na fanna, fo-filly,
fee fi mo-milly, Billy!
–
Let's do Marsha!
Marsha, Marsha, bo-barsha
bo-na-na fanna, fo-farsha
fee fi mo-arsha, Marsha!
–
Little trick with Nick!
Nick, Nick, bo-bick,
bo-na-na fanna, fo-fick,
fee fi mo-mick, Nick

Input The input is a single line containing a name (any) of N letters, $2 \leq N \leq 100$, and starting with a capital letter.

Output The output should contain three lines corresponding to the rhyming verse for the name provided as an input, as suggested by the instructions in the song.

Sample Input 1

```
Shirley
```

Sample Output 1

```
Shirley, Shirley, bo-bhirley
bo-na-na fanna, fo-fhirley
fee fi mo-mhirley, Shirley!
```

Sample Input 2

```
Marjorie
```

Sample Output 2

```
Marjorie, Marjorie, bo-barjorie
bo-na-na fanna, fo-farjorie
fee fi mo-arjorie, Marjorie!
```

[2] taken from "The Name Game", by Shirley Ellis (1964)

9 Tara's Terrific Tugging Tale

(CPU:1sec - RAM:256MB)

Tara is a prolific author of fantasy fiction. In one of her fascinating worlds, she created a people of tiny creatures very fond of tug-of-war contests. In those epic battles, she describes the jolly injunctions those merry farmers throw at each other to come give a hand to their favourite team.

Yes, in those tugs-of-war, the teams can gain new members! Actually, in her book, Tara describes in great length the notes taken by the anthropologist who follows the gnomes. Here is an extract of the scientist's observations:

" Those teensy-weensy folks are fascinating. They all appear to be strictly identical in all matters: intelligence, strength, clothes, behaviour, and personality. Still, they really enjoy each other's company. In particular, they seem to be well versed in the art of synchronised tugging. After forming lines, they start playing drums to set the rhythm, and they all tug on the rope at the same time every 16 beats. In between two rounds of tugging, they all seem to follow the same pattern:

- *If a gnome is tugging and both teammates in front of him and the one behind him are tugging too, then he will not be tugging on the next round.*
- *If a gnome is tugging and at most one of his neighbouring teammates is tugging too, then he will keep tugging on the next round.*
- *If a gnome is not tugging, then if at least his teammate in front of him is tugging, he will be tugging on the next round. Otherwise, he will continue being idle.*
- *At the end of the round, if the last gnome in the line has tugged, then an additional gnome will join the team to tug on the next round. Both teams often gain a member every round!*

The first team to tug the other team over the central line wins. Funnily enough, they count in gnome length (gl) and each gnome is capable of tugging for exactly one gnome length per round. They usually agree on the distance to tug to win before they start the match. I guess it depends on the mood of the crowd and on the number of gnomes capable of joining the battle. "

Mesmerised by the tale, you decide to simulate the epic contests.

Input The input will consist of two lines:

- The first line will contain 2 integers, N being the number of gnome length advantage a team needs to win ($5 \leq N \leq 5000$), and M being the total number of gnomes present ($20 \leq M \leq 5000$).
- The second line will contain the initial state (round 0) of the two teams in the form of two space-separated strings of 0s and 1s. 0 indicates a gnome that is not tugging this round, 1 indicates a gnome that is tugging this round. Each team will start with a maximum of 1000 gnomes.

The left team tugs towards the left, and the right team tugs towards the right. If the crowd runs out of gnomes and the two teams require one for the new round, then the left team is always favoured.

Output The output should consist of either the word right or the word left, indicating which team wins then followed by a space and an integer indicating at which round the team actually won (the provided starting state corresponds to round 0).

Sample Input 1

```
5 20
10101 01010
```

Sample Output 1

```
left 5
```

Sample Input 2

```
20 50
111000111 100111001
```

Sample Output 2

```
left 15
```

10 Executive Eileen's Effective Emails (CPU:1sec - RAM:256MB)

Eileen is the new head of the IT department. As a first policy, she would like to re-assign email addresses to all of the employees of the university (because why not). Email addresses have to respect a certain standard: they are made only of lowercase letters, cannot contain spaces or punctuation (for instance as in the Irish name O'Callaghan). All punctuation will simply be occulted in the email addresses, no capital letters should be used, and all spaces are to be replaced by dots.

As in many other places employing a lot of people, University College Cork has a certain number of employees with similar names, potentially leading to similar email addresses. In order to prevent conflicts, Eileen just decided that in case of a conflict, a number would be assigned to every following employee with a name already used. Hence, if John McLoughlin is the first one, he will get the email address `john.mcloughlin@ucc.ie`. The next employee with a similar name will be assigned the address `john.mcloughlin2@ucc.ie`.

As the newcomer to the IT department, you have been assigned with the task of implementing this simple policy.

Input The input will consist of a first line containing a single integer N, $4 <= N <= 50000$, then followed by N lines, each one containing a full name made of up to 128 characters. The provided names will not contain any special character (e.g. Róisín would be provided as Roisin).

Output The output should consist of N lines, the i-th line containing the email address to be assigned to the i-th name provided as an input.

Sample Input 1

```
4
Bastien Pietropaoli
Milan De Cauwer
Raffaele Baldassini
Eduardo Vyhmeister
```

Sample Output 1

```
bastien.pietropaoli@ucc.ie
milan.de.cauwer@ucc.ie
raffaele.baldassini@ucc.ie
eduardo.vyhmeister@ucc.ie
```

Sample Input 2

```
5
Aoife Murphy
Sean Kelly
Roisin Murphy
Iarfhlaith O'Sullivan
Sean Kelly
```

Sample Output 2

```
aoife.murphy@ucc.ie
sean.kelly@ucc.ie
roisin.murphy@ucc.ie
iarfhlaith.osullivan@ucc.ie
sean.kelly2@ucc.ie
```