Programming Competitions

IrlCPC - AIPO

How You Can Contribute
How We Created The Problem Set

Milan De Cauwer (Confirm@UCC)
Bastien Pietropaoli (Insight@UCC)

UCC-ACM, IrICPC, AIPO

What are they?

What is it?

UCC-ACM student chapter

IrICPC - Irish Collegiate Programming Contest

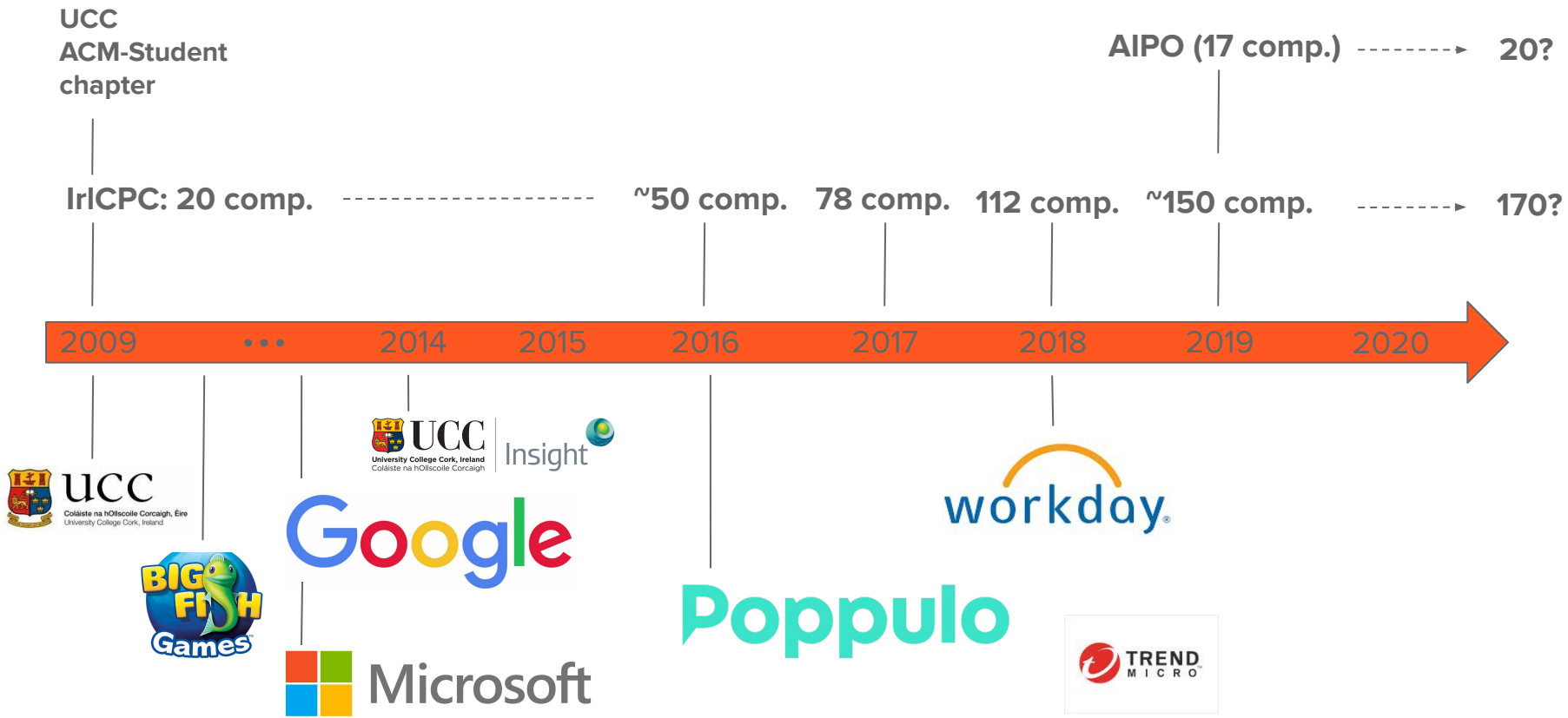
Created: 2009 **Audience:** College Students
By: UCC-ACM Student Chapter (Many should be thanked)
From about 20 competitors in 2010 to ~150 in 2019.

AIPO - All-Ireland Programming Olympiad

Created: ?? **Audience:** Secondary level
By: DCU before
Passed upon UCC (Sabin, Andrea, Federico) last year
Last year: 17 competitors

Competitors

Timeline


UCC
ACM-Student
chapter
IrICPC: 20 comp.
AIPO (17 comp.)
20?
~50 comp.
78 comp.
112 comp.
~150 comp.
170?
2009
2014
2015
2016
2017
2018
2019
2020
UCC
Coláiste na hOllscoile Corcaigh, Éire
University College Cork, Ireland
UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh
Insight
Google
workday
BIG FISH
Games
Poppulo
Microsoft
TREND
MICRO

Who were the sponsors? (2019)

Google

Microsoft

Poppulo

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Insight

workday.

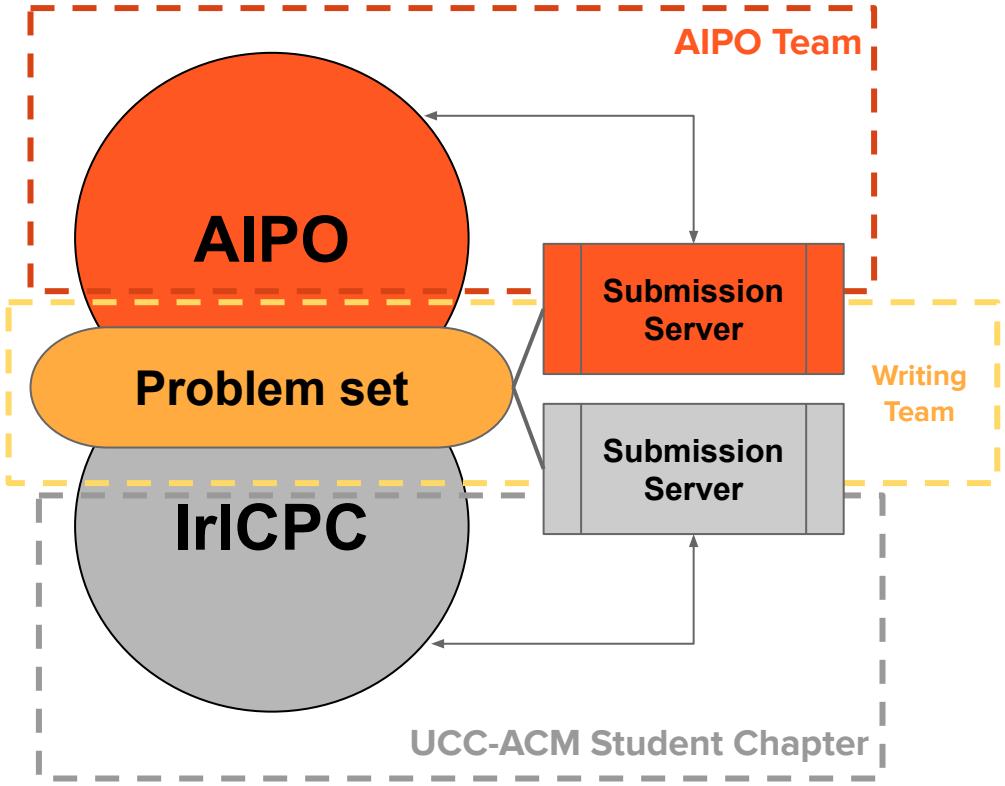
How does it work?

We create a mostly algorithmic problem set with test cases.

Teams/students submit their code to a server.

It gets automatically evaluated and they score points.

The highest scoring team wins **nice prizes ($$$)**.



How does it work? (Question Writer)

1

Find inspiration.

Wikipedia binges.

Reference books (Intro to algorithms *et al.*).

Exercice books.

Interview questions.

2

Nail the question.

Target algorithm?

Give it your own spin!

3

Create test cases.

Try to break that code.

Does it scale?

Does it have practical issues?

4

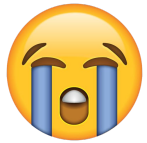
Tune the question.

Test it.

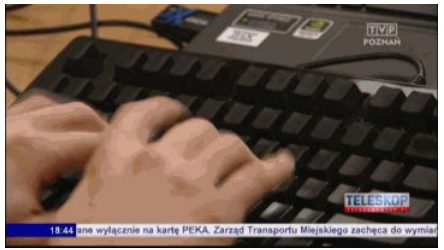
Tune the cases.

Fluff it up!

Embrace the rage.



How does it work? (Competitor)



1

Read the problems.

Read the **ENTIRE** problem set before starting!

What's easy?
What's hard?

2

Work as a team.

Share ideas.

Split work.

Who's coding?

3

Find the algorithms.

Try to break that code.

Does it scale?

Does it have practical issues?

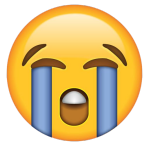
4

Submit code.

Test it.

Fail.

Retry!



How does it work? (Organiser)

1

UCC-ACM student chapter.

Annual General Meeting (TBA).

Take on a position.

Monthly meetings.

2

Do your f*ing job!**

Catering, room booking, networking with sponsors, branding and visual design, accounting, PR and social media, ...

Be active!

3

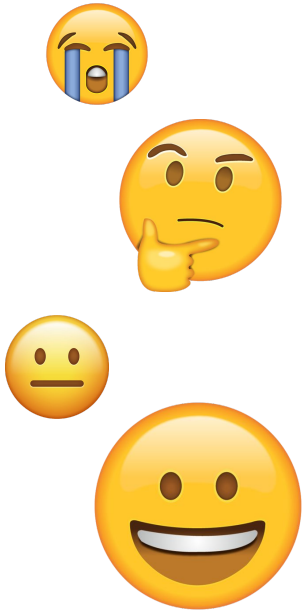
Enjoy the day.

No loitering!

Be active.

Assist competitors.

Get a nice t-shirt!



2019 Scoreboards

2019 scoreboard - IrlCPC

✓	Rank	First Name	Last Name	Team	Pando…	Caroli…	Taras_…	Aoifes…	Kieras…	Stellas…	Janine…	Shirle…	Laoise…	Eileen…	IrlCPC 20…	Global
	1	-=[B]ichael [B] [B]iggins=-			10	100	100	90	100	0	70	100	100	100	770	770
	2	Computational Triumph			40	60	90	100	10	0	100	100	100	100	700	700
	2	INSA Lyon 1			0	100	100	100	0	0	100	100	100	100	700	700
	4	O(-1)			0	40	90	100	40	0	0	100	100	100	570	570
	5	INSA Lyon 2			0	100	0	100	0	0	30	100	100	100	530	530
	6	Crypto			0	90	100	100	0	0	0	0	100	100	490	490
	6	YannWang			0	10	0	80	80	20	0	100	100	100	490	490
	8	Mysterious Net Dudes			0	0	100	80	0	0	0	100	100	100	480	480
	9	Byte_Me			0	0	0	100	40	0	0	100	100	100	440	440
	10	Darkline			0	30	0	60	10	20	0	100	100	100	420	420
	10	NUIG Runtime Errors			0	20	0	100	0	0	0	100	100	100	420	420
	10	OOPs			0	70	0	90	0	0	0	100	60	100	420	420
	13	Jambe de Bois			0	20	0	50	40	0	0	100	100	100	410	410
	14	Pycharmers			0	50	0	90	40	20	0	100	100	0	400	400
	15	MINDS			0	0	0	90	0	0	0	100	100	100	390	390
	15	succ(X)			0	10	0	50	30	0	0	100	100	100	390	390
	17	icuthere			0	0	0	100	40	0	0	100	10	100	350	350
	18	idk			40	0	0	0	40	20	0	100	70	50	320	320
	19	ThinkWinWin			0	0	0	100	0	0	0	0	100	100	300	300
	20	Crash Test Dummies			0	0	0	0	0	0	0	100	100	90	290	290
	21	Cloughjordan			0	0	0	20	0	0	0	70	80	100	270	270

2019 scoreboard - IrlCPC

✓	Rank	First Name	Last Name	Team	Pando…	Caroli…	Taras_…	Aoifes…	Kieras…	Stellas…	Janine…	Shirle…	Laoise…	Eileen…	IrlCPC 20…	Global
	1		-=[B]ichael [B] [B]iggins=-		10	100	100	90	100	0	70	100	100	100	770	770
	2		Computational Triumph		40	60	90	100	10	0	100	100	100	100	700	700
	2		INSA Lyon 1		0	100	100	100	0	0	100	100	100	100	700	700
	4		O(-1)		0	40	90	100	40	0	0	100	100	100	570	570
	5		INSA Lyon 2		0	100	0	100	0	0	30	100	100	100	530	530
	6		Crypto		0	90	100	100	0	0	0	0	100	100	490	490
	6		YannWang		0	10	0	80	80	20	0	100	100	100	490	490
	8		Mysterious Net Dudes		0	0	100	80	0	0	0	100	100	100	480	480
	9		Byte_Me		0	0	0	100	40	0	0	100	100	100	440	440
	10		Darkline		0	30	0	60	10	20	0	100	100	100	420	420
	10		NUIG Runtime Errors		0	20	0	100	0	0	0	100	100	100	420	420
	10		OOPs		0	70	0	90	0	0	0	100	60	100	420	420
	13		Jambe de Bois		0	20	0	50	40	0	0	100	100	100	410	410
	14		Pycharmers		0	50	0	90	40	20	0	100	100	0	400	400
	15		MINDS		0	0	0	90	0	0	0	100	100	100	390	390
	15		succ(X)		0	10	0	50	30	0	0	100	100	100	390	390
	17		icuthere		0	0	0	100	40	0	0	100	10	100	350	350
	18		idk		40	0	0	0	40	20	0	100	70	50	320	320
	19		ThinkWinWin		0	0	0	100	0	0	0	0	100	100	300	300
	20		Crash Test Dummies		0	0	0	0	0	0	0	100	100	90	290	290
	21		Cloughjordan		0	0	0	20	0	0	0	70	80	100	270	270

UCC

2019 scoreboard - AIPO

✓	Rank	First Name	Last Name	Team	Eileen…	Shirle…	Aoifes…	Laoise…	Taras …	Caroli…	Kieras…	Stellas…	Janine…	Pando…	AIPO 201…	Global
	1		Kieran Horgan		100	100	100	100	100	100	100	20	10	30	760	760
	2		Andrew Nash		100	100	100	100	100	100	40	0	0	0	640	640
	3		Caolan Fleming		100	100	100	100	100	90	0	0	0	0	590	590
	4		Harvey Brezina Conniffe		100	100	100	100	100	20	0	0	0	0	520	520
	5		Franciszek Lajszczak		100	100	90	100	100	10	0	0	0	0	500	500
	6		Oisin Davey		100	100	100	100	90	0	0	0	0	0	490	490
	7		Alan Panayotov		70	100	80	100	10	90	0	0	0	0	450	450
	8		Shang Lucey		100	100	90	90	0	40	0	0	0	0	420	420
	9		Brendan Alinquant		70	100	100	90	0	40	0	0	0	0	400	400
	10		Suneet Mahajan		50	100	100	60	10	40	0	0	0	0	360	360
	11		Jack Connolly		70	100	80	70	10	10	0	0	0	0	340	340
	12		Hubert Slapczynski		40	100	60	100	0	0	0	0	0	0	300	300
	13		Ayushi Mahajan		90	100	90	0	0	0	0	0	0	0	280	280
	14		David Brett		70	100	50	0	0	0	0	0	0	0	220	220
	15		kate forrest		0	100	40	50	0	0	0	0	0	0	190	190
	16		Luca Tuffy		60	100	0	0	0	0	0	0	0	0	160	160
	17		Dylan O' mahony		0	100	0	0	0	0	0	0	0	0	100	100
	17		Thomas Galligan		0	100	0	0	0	0	0	0	0	0	100	100
	19		Benjamin Murray		0	0	0	0	0	0	0	0	0	0	0	0
	19		David Selmeczy		0	0	0	0	0	0	0	0	0	0	0	0

2019 scoreboard - AIPO

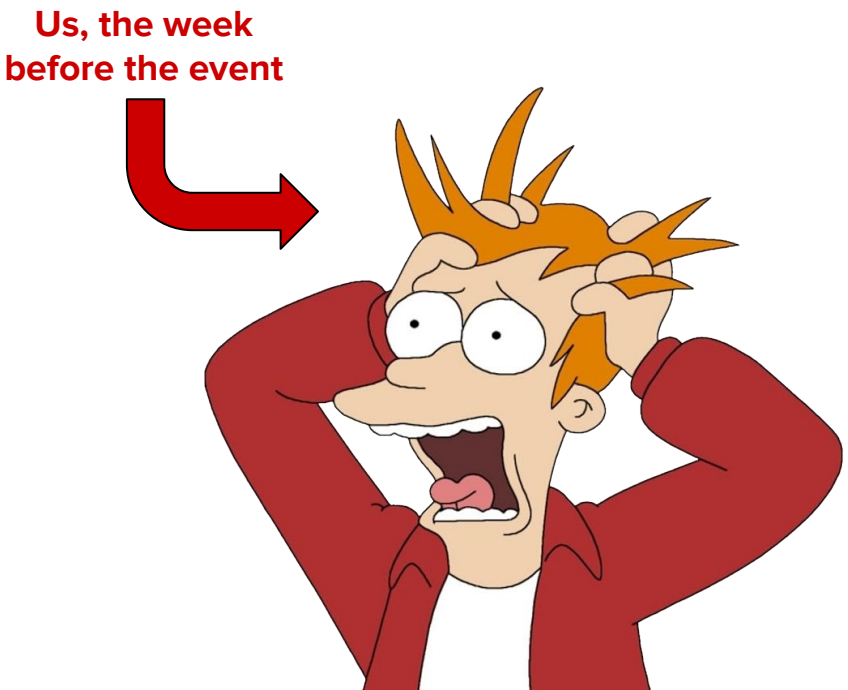
MPT

✓	Rank	First Name	Last Name	Team	Eileen…	Shirle…	Aoifes…	Laoise…	Taras …	Caroli…	Kieras…	Stellas…	Janine…	Pando…	AIPO 201…	Global
	1		Kieran Horgan		100	100	100	100	100	100	100	20	10	30	760	760
	2		Andrew Nash		100	100	100	100	100	100	40	0	0	0	640	640
	3		Caolan Fleming		100	100	100	100	100	90	0	0	0	0	590	590
	4		Harvey Brezina Conniffe		100	100	100	100	100	20	0	0	0	0	520	520
	5		Franciszek Lajszczak		100	100	90	100	100	10	0	0	0	0	500	500
	6		Oisin Davey		100	100	100	100	90	0	0	0	0	0	490	490
	7		Alan Panayotov		70	100	80	100	10	90	0	0	0	0	450	450
	8		Shang Lucey		100	100	90	90	0	40	0	0	0	0	420	420
	9		Brendan Alinquant		70	100	100	90	0	40	0	0	0	0	400	400
	10		Suneet Mahajan		50	100	100	60	10	40	0	0	0	0	360	360
	11		Jack Connolly		70	100	80	70	10	10	0	0	0	0	340	340
	12		Hubert Slapczynski		40	100	60	100	0	0	0	0	0	0	300	300
	13		Ayushi Mahajan		90	100	90	0	0	0	0	0	0	0	280	280
	14		David Brett		70	100	50	0	0	0	0	0	0	0	220	220
	15		kate forrest		0	100	40	50	0	0	0	0	0	0	190	190
	16		Luca Tuffy		60	100	0	0	0	0	0	0	0	0	160	160
	17		Dylan O' mahony		0	100	0	0	0	0	0	0	0	0	100	100
	17		Thomas Galligan		0	100	0	0	0	0	0	0	0	0	100	100
	19		Benjamin Murray		0	0	0	0	0	0	0	0	0	0	0	0
	19		David Selmeczy		0	0	0	0	0	0	0	0	0	0	0	0

How do we create the problem set?

Ingredients for a good problem set

A lot of work.
A cup of patience.
A pinch of fluff.
A zest of pedantism.
Sweat, tears, and potentially blood.
Booze and drugs (optional).



More seriously

What's needed:

- Questions ranging from "dead easy" to "wtf is this?!".
- Interesting/original problems to solve.
- Cunning corner cases.
- Scalability issues.

All of this creates a smooth distribution of scores.

Us, writing sneaky test cases

A dead-easy question

If you read it, it's actually easy.

A simple string manipulation.

Original format (everyone hated it tbh).

Everyone solved it in AIPO.

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.

A sneaky scalability-issue question

Not that hard if you think carefully.

Not easy either.

Symmetries can be exploited.

3^50 > 2^64 (someone reimplemented big numbers in C++...)

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.

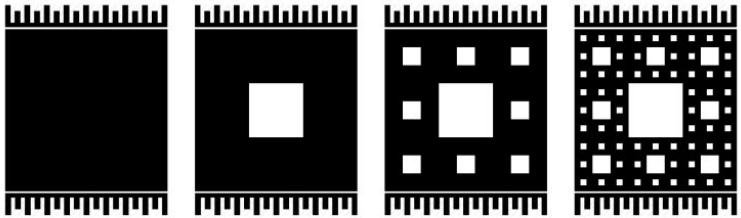


Figure 2: Carpet drawn for orders 0 to 3 for some arbitrary dimension $D \geq 3$.

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.

A fluffy question

Medium question, requires careful reading!

A simple 1D automaton.

Extra fluffy question.

Most questions we received were about this particular question. (I'm really proud of this one.)

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).

A “wtf is this” question

Super tough question.

Was first in the problem set (random order for the questions)

Doesn’t look that hard.

“Unusual” solution on our side (Ray tracing + Monte Carlo integration muahahaha)

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.

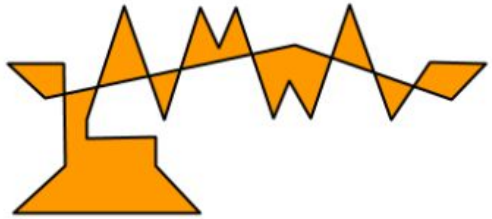


Figure 1: One of the latest "creations" of Pandora, named "Crippling Lamp".

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).

How can you contribute?

Submit your problem ideas to Milan and me.

milan.decauwer@ucc.ie
bastien.pietropaoli@ucc.ie

Write problems with us, join the writing team.

Review problems with us.

Solve our problems:

- Use various languages (Python, C, C++, Java, etc.).
- Confirm our solutions are correct (even with an invalid solution for the day!).
- Find potential issues.
- Find unpredicted/unforeseen corner cases.
- Evaluate the difficulty of the questions.

All of this is useful!

How we'd like to do it this year

Regular meetings starting in November:

- Every week? Every other week?
- Prevents having to panic at the end

Having 10-12 questions as last year (1 is already written!).

- 1-2 dead-easy one (so everyone can score something)
- 1-2 easy ones (to build up confidence)
- 2-4 medium ones (so there's some competition among the medium teams)
- 2-3 hard questions (three was probably overkill but some people are really good)
- 1 new type of question (to make sure they stay busy, it's already written)

Have a minimum of 20 test cases per question (instead of 10 last year)

We need you!