De eerste ronde Nederlandse Informatica Olympiade 2025-2026

De informatica olympiade is een wedstrijd voor leerlingen uit het voortgezet onderwijs in Nederland. Het is een wedstrijd die bestaat uit drie ronden. In de derde ronde wordt bepaald wie Nederland mogen vertegenwoordigen op de Internationale Informatica Olympiade in 2026 in Oezbekistan.

De eerste ronde

De eerste ronde van de Nederlandse Informatica Olympiade bestaat dit jaar uit 13 opgaven. Die hoef je niet allemaal te maken, al mag dat natuurlijk wel. Deelnemers die tenminste 200 punten halen krijgen een certificaat.

Heb je tussen de 200 en 349 punten dan staat op het Certificaat de vermelding **Brons**, tussen de 350 en 499 punten de vermelding **Zilver** en bij 500 punten of meer punten de vermelding **Goud**.

Soort	Omschrijving	Aantal	Punten per opgave	Totaal te behalen
A	Inleidende opgaven	5	40	200
B	Theoretische opgaven	4	25	100
C	Gevorderde opgaven	3	50, 50 en 100	200
D	Een spel programmeren	1	100	100

De beste 100 leerlingen worden uitgenodigd voor de tweede ronde, die in februari of maart 2026 wordt gehouden op de Universiteit Twente. Voor deelname aan de tweede ronde moet je wel in totaal minstens 250 punten hebben gehaald

Om deel te kunnen nemen moet je een account maken op submit.informaticaolympiade.nl

Bij de eerste keer aanmelden moet je enkele gegevens aanleveren die wij nodig hebben om de olympiade goed te kunnen organiseren. Als je deze gegevens niet wilt of kunt aanleveren, kun je helaas niet deelnemen. Je verklaart in de laatste stap dat je de gegevens naar waarheid hebt ingevuld; daarna staat deelname voor je open. Als je van vorige jaren al een

account hebt, zul je de gegevens ook eventueel eerst moeten aanvullen voor je verder kunt werken in het systeem. Wij gaan zeer zorgvuldig om met de gegevens die je ons aanlevert. Wij zullen deze gegevens niet met derden delen.

Je kunt je uitwerkingen uploaden naar submit.informaticaolympiade.nl wanneer je in het systeem bent ingelogd. In het systeem kun je ook een voorbeeldopgave insturen om uit te proberen hoe het werkt. De opgaven worden meteen geheel of gedeeltelijk nagekeken, voor de rest van de uitslag zul je moeten wachten op het resultaat. Je uitwerkingen voor de opgaven A, B en C moeten uiterlijk 22 januari worden geüpload. Op 24 januari wordt de eerste ronde gejureerd en kort daarna worden de uitslagen gepubliceerd.

Voor de spelopgave, opgave D, moet je je aanmelden op www.codecup.nl en kun je via die site ook je programma uploaden. De deelnemende programma's die meewerken met het jurysysteem komen op 24 januari 2026 tegen elkaar uit in een toernooi dat te volgen is op www.codecup.nl. Inzenden mag tot 24 januari 7.00 u. Dan begint het toernooi.

Voor alle opgaven geldt dat je ervan uit mag gaan dat je programma's alleen correcte invoer aangeboden krijgen.

Opgaven A1 tot en met A5

Deze opgaven zijn vooral bedoeld voor leerlingen die beginnen met programmeren.

De programma's die je moet schrijven lezen invoer van standard input (het toetsenbord) en schrijven uitvoer naar standard output (het beeldscherm). Je programma moet zich daarbij precies houden aan de beschrijvingen van de opdracht. Je programma krijgt een aantal testgevallen voorgeschoteld en voor ieder testgeval kun je punten krijgen. Tip: je mag absoluut geen extra teksten afdrukken zoals 'geef je invoer' of 'het antwoord is'. Het jurysysteem is heel streng en rekent extra teksten altijd fout.

Vanuit de olympiade bieden we lesmateriaal aan om te beginnen met programmeren met Python. Dat is de cursus CS Circles van de Universiteit van Waterloo in Canada. Er is een Nederlandse vertaling beschikbaar op cursus.informaticaolympiade.nl. Ook is er een introductiecursus beschikbaar die samen met EGOI (European Girls' Olympiad in Informatics) is ontwikkeld en door iedereen gebruikt mag worden: https://girls.gitbook.io/c++-cursus

Opgaven B1 tot en met B4

Deze opgave kun je één voor één downloaden uit het inzendsysteem. De opgave wordt speciaal voor jou gemaakt en jij moet het antwoord op de opgave die je vanuit het systeem krijgt inleveren. Het heeft dus geen zin om de antwoorden van iemand anders te gebruiken en die in te zenden.

Als je binnen een week na downloaden het goede antwoord instuurt krijg je 25 punten per opgave. Voor iedere dag later gaat er één punt van je score af. Inzendingen na 22 januari 2026 zullen niet worden verwerkt.

Als je een verkeerd antwoord hebt gegeven, verlies je meteen 5 punten, totdat er van de 25 punten geen punten meer over zijn.

Het gaat bij al deze opgaven om korte antwoorden, een getal of een korte tekst, die je op de betreffende pagina van het inzendsysteem kunt invoeren. Als je je antwoord hebt bevestigd, krijg je direct je score te zien.

Je mag allerlei hulpmiddelen gebruiken om de opgave op te lossen. Je zou er bijvoorbeeld een computerprogramma bij kunnen schrijven. Noodzakelijk is dat echter niet. Als voorbereiding op het vervolg van de informatica olympiade is het wel een mooie uitdaging om na te gaan hoe je een programma zou kunnen schrijven dat dit probleem, of problemen die erop lijken, kunt oplossen.

Opgaven C1 tot en met C3

Dit zijn complexere opgaven waarmee je een probleem moet oplossen door het schrijven van een computerprogramma.

Opgave D en de CodeCup

Bij deze opgave moet je een programma schrijven dat het spel Zero Point One kan spelen. Dat is het spel van de CodeCup 2026. Aan het toernooi om de CodeCup doen ook andere deelnemers mee, soms wel uit meer dan twintig verschillende landen. Zie hiervoor ook www.codecup.nl

De programma's voor het spel spelen op 24 januari een toernooi tegen elkaar. Om deel te kunnen nemen moet je programma kunnen samenwerken met onze jurysoftware; voor details verwijzen we naar www.codecup.nl

Opgave A1. Inoren (Inkorten)

Schrijf een programma dat van standard input een regel met daarop een woord inleest. Het woord is geschreven in hoofdletters en bestaat uit minstens 2 en hoogstens 80 letters. Een woord bestaat uitsluitend uit letters zonder trema of accentteken, en een woord bevat geen leestekens, spaties of andere toevoegingen.

Jouw programma schrijft naar standard output een ingekorte versie van dat woord. Daarbij gelden de volgende regels:

- Er zijn 26 letters: **A**BCD**E**FGH**I**JKLMN**O**PQRST**U**VWXYZ. Klinkers zijn de letters A, E, I, O en U. Alle andere letters zijn medeklinkers (dus ook de Y).
- De letters van een woord worden één voor één van links naar rechts verwerkt.
- De eerste letter blijft staan.
- Als er na een klinker een medeklinker volgt blijft die medeklinker staan; als er na een klinker een klinker volgt, wordt die klinker erna weggelaten.
- Als na een medeklinker een klinker volgt blijft die klinker staan; als na een medeklinker een medeklinker volgt, wordt die medeklinker erna weggelaten.
- Het kan gebeuren dat de uitvoer gelijk is aan de invoer.
- Voor je programma geldt een tijdslimiet van 1 seconde.

Voorbeelden

Invoer: AARDAPPEL	Invoer: ANGSTSCHREEUW	Invoer: LYNX	Invoer: MINIBANANEN
Uitvoer: ARAPEL	Uitvoer: ANEW	Uitvoer: L	Uitvoer: MINIBANANEN

Opgave A2. Boeken kopen

Voor de schoolbibliotheek mogen drie leerlingen een wensenlijst inleveren met de boeken die ze graag willen laten kopen. Er is een catalogus met 10000 boeken waaruit ze kunnen kiezen; elk boek heeft daarin een eigen boeknummer. De leerlingen leveren elk hun wensenlijst met boeknummers in bij de schoolbibliotheek. Alle boeken op de drie wensenlijsten worden gekocht, maar elk boek wordt maar eenmaal aangeschaft.

Schrijf een programma dat vier regels leest van standard input.

- Op de eerste regels staan drie getallen N_1, N_2 en N_3, telkens gescheiden door een spatie. Daarmee wordt aangegeven hoeveel boeken er op de wensenlijst van elke leerling staat.
- Vervolgens komen er drie regels met achtereenvolgens N_1, N_2 en N_3 boeknummers, telkens gescheiden door een spatie.
- Alle getallen in de invoer zijn groter dan 0 en niet groter dan 10000.
- In een wensenlijst kan elk boeknummer maar hoogstens eenmaal voorkomen.
- Het programma schrijft naar standard output één regel met daarop het aantal boeken dat de bibliotheek zal aanschaffen.
- Voor je programma geldt een tijdslimiet van 1 seconde.

Voorbeelden

Invoer:	Invoer:	Invoer:

`5 6 4`
`12 387 15 162 5`
`14 162 92 387 7 748`
`14 5 12 387`
	`3 2 6`
	`1100 822 17`
	`25 1100`
	`17 25 822 333 1 19`
		`1 1 1`
		`1`
		`2`
		`3`

Uitvoer:	Uitvoer:	Uitvoer:
`9`	`7`	`3`

Opgave A3. NIO-getal

In deze opgave wordt van een positief geheel getal n (n > 1) zijn NIO-getal `N(n)` bepaald. Dat gaat als volgt:

- Als het getal n zelf **geen** priemgetal is, schrijf je het getal n als een product van 2 of meer priemgetallen p.

> **De hoofdstelling van de getaltheorie zegt dat elk positief geheel getal dat geen priemgetal is op een unieke manier kan worden ontbonden als product van priemgetallen.**
>
> **Een priemgetal is een positief geheel getal, groter dan 1, dat slechtst twee verschillende positief gehele getallen als deler heeft: 1 en zichzelf (1 is dus geen priemgetal.)**

Voorbeeld: het getal 60 kan ontbonden worden in het product van de priemgetallen 60 = 2 x 2 x 3 x 5.

Om het NIO-getal te bepalen vervang je elk priemgetal p uit deze ontbinding in het getal `p+1`.

Het NIO-getal `N(n)` krijg je door deze nieuwe getallen met elkaar te vermenigvuldigen.

Voorbeeld `N(60) = (2+1) x (2+1) x (3+1) x (5+1) = 3 x 3 x 4 x 6 = 216`.

- Als het getal n **wel** een priemgetal p is, dan wordt het NIO-getal `N(n) = p + 1`

Voorbeeld het getal 7 is een priemgetal. Het NIO-getal `N(7) = 7 + 1 = 8`

Schrijf een programma dat een regel leest van standard input. Op die regel staat een getal n. Er geldt $2 \leq n \leq 10000$

Het programma schrijft één regel naar standard output met daarop het NIO-getal `N(n)`.

Voor je programma geldt een tijdslimiet van 1 seconde.

Voorbeelden:

Invoer: `60`	Invoer: `1001`	Invoer: `1024`
Uitvoer: `216`	Uitvoer: `1344`	Uitvoer: `59049`

Opgave A4. Terugrekenen

In deze opgave rekenen we weer aan NIO getallen, maar we rekenen nu de andere kant op dan bij opgave A3.
Als invoer krijg je een getal `m`, en je moet op zoek naar al die getallen n waarvoor er geldt `N(n) = m`.
Dat kunnen meerdere getallen n zijn, maar het kan ook zijn dat er geen getal n is waarvoor geldt `N(n) = m`.

Schrijf een programma dat een regel leest van standard input. Op die regel staat een getal `m`. Er geldt `2 ≤ m ≤ 10000.`

Het programma schrijft één of meer regels uitvoer naar standard output.

- Op iedere regel staat een getal n waarvoor geldt dat `N(n) = m`.
- De mogelijke getallen n staan oplopend in de uitvoer.
- Als er geen getal n bestaat waarvoor geldt dat `N(n) = m` dan bestaat de uitvoer uit één regel met daarop het getal `0`.
- Voor je programma geldt een tijdslimiet van 1 seconde.

Voorbeelden:

Invoer:

```
216
```

Uitvoer:

```
56
60
92
102
110
125
142
159
187
```

Invoer:

```
1344
```

Uitvoer:

```
546
585
747
806
861
897
1001
1115
1169
1271
```

Invoer:

```
1025
```

Uitvoer:

```
0
```

Opgave A5. Donut doolhof

o	o	o	#	#	#	o	#	#
#	M	o	#	o	o	#	o	#
#	o	#	o	#	o	#	#	o
#	o	o	o	o	#	o	o	o
o	#	o	o	o	#	#	D	o
o	o	o	#	#	o	#	o	o

In het diagram hierboven zie je een voorbeeld van een doolhof met `R` rijen en `K` kolommen.

De opdracht is om de muis `M` met zo min mogelijk stappen naar de donut `D` te laten gaan.

Spelregels:

- De muis maakt een verticale stap (een vak omhoog of omlaag) of een horizontale stap (een vak naar links of naar rechts).
- Vakken met een # zijn niet toegankelijk voor de muis.
- Vakken met een o zijn wel toegankelijk voor de muis.
- Het doolhof heeft iets bijzonders: je kunt altijd doorlopen en je kunt nooit uitstappen. Als je helemaal bovenaan staat en je stapt verder naar boven, dan kom je beneden in dezelfde kolom uit. En als je helemaal rechts staat en je stapt verder naar rechts, dan kom je aan de linkerkant in dezelfde rij uit. Dat geldt ook voor links en onderaan.

Schrijf een programma dat leest van standaard input
- eerst een regel met twee getallen `R (1 < R ≤ 900)` en `K (1 < K ≤ 900)`, gescheiden door een spatie.
- vervolgens `R` regels met elk `K` symbolen.

Je programma geeft als uitvoer één regel naar standaard output met daarop het aantal stappen van de kortste weg van de muis naar de donut. Als de donut niet bereikbaar is voert je programma `0` uit.

Voor je programma geldt een tijdlimiet van 1 seconde.

Voorbeeld

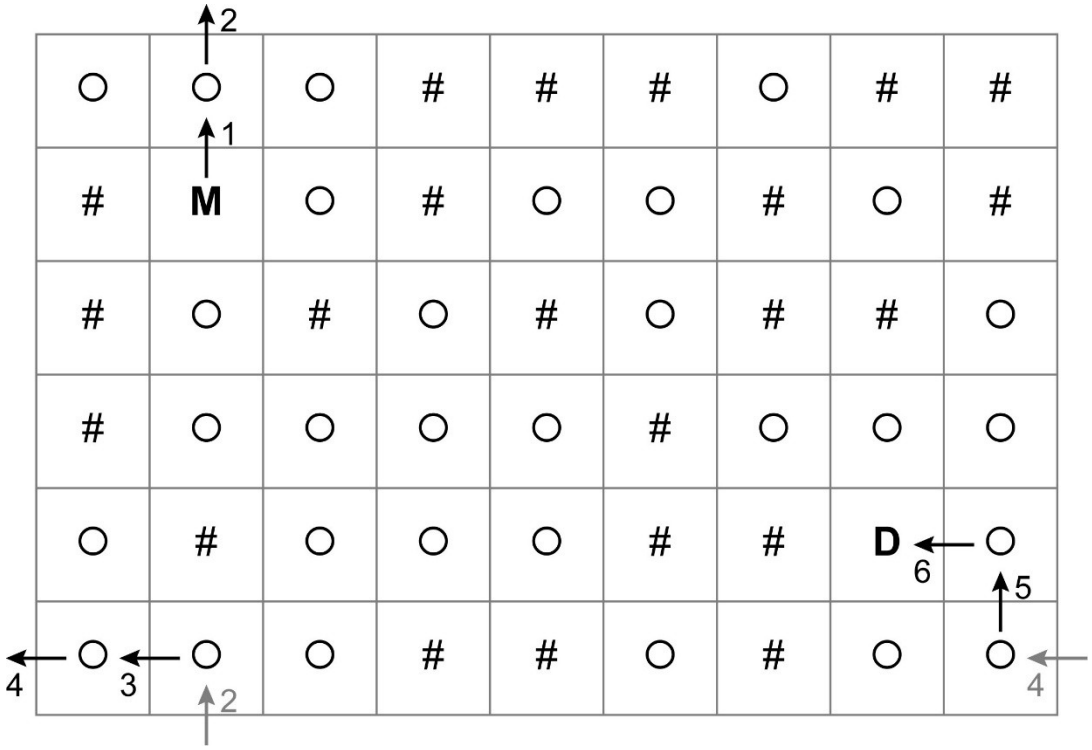
Invoer:

```
6 9
ooo###o##
#Mo#oo#o#
#o#o#o##o
#oooo#ooo
o#ooo##Do
ooo##o#oo
```

Uitvoer:

6

Toelichting:



In het diagram staat een mogelijke route van 6 stappen aangegeven. Stap 2 is de stap van de bovenste rij naar dezelfde kolom van de onderste rij. Stap 4 gaat van de meest linkse kolom naar de dezelfde rij van de meest rechtse kolom.

Opgave B1 tot en met B4:

Download deze van submit.informaticaolympiade.nl

B1. Pentominoverkaveling

B2. Woordslang

B3. Liftplan

B4. GN puzzel

Opgave C. Rooster Reductie

Inleiding

Alle versleuteling die vandaag gebruikt wordt op het internet, kan gekraakt worden met grote kwantumcomputers. Kwantumcomputers kunnen namelijk heel efficiënt getal ontbinden in een product van priemgetallen (zie opgave A3). Het zal waarschijnlijk nog jaren duren voordat er een kwantumcomputer zo groot en zo betrouwbaar kan worden gemaakt dat die getallen kan ontbinden die gebruikt worden in versleuteling, getallen die 2048 bits hebben.

Voordat grote kwantumcomputers beschikbaar zijn, zijn onderzoekers druk bezig om alternatieve versleutelingsmethodes te ontwikkelen, die niet te kraken zijn door computers én niet door kwantumcomputers. Veel van die alternatieven zijn gebaseerd op roosters, in het Engels "*lattice*", en deze zullen in de komende 10 jaar de huidige versleuteling vervangen.

In deze opgave bekijken we hoe roosters gebruikt kunnen worden voor versleuteling. De veiligheid van op roosters gebaseerde versleutelingsmethodes komt voort uit de lastigheid om in een rooster punten te vinden die dicht bij de oorsprong liggen. Hoewel dit in dimensie 1, 2 of 3 nog makkelijk is, is dit probleem extreem lastig in hoge dimensies.

Voor meer achtergrondinformatie over dit onderwerp, zie:
https://www.nist.gov/cybersecurity/what-post-quantum-cryptography (Engels)

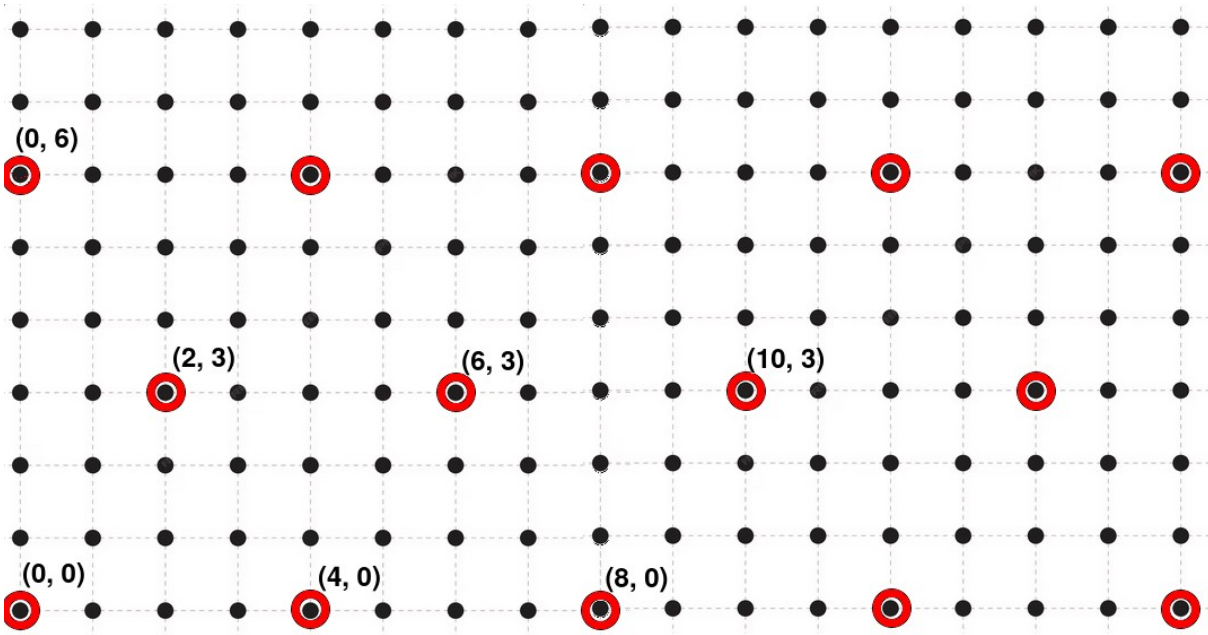
Roosters

Een rooster ***R*** is een verzameling van punten in een ***n*** dimensionale ruimte die aan deze eisen voldoet:

- De oorsprong ***(0, 0, …, 0)*** ligt in het rooster ***R***,
- Als ***x*** en ***y*** twee roosterpunten zijn, dan is ***x+y*** ook een roosterpunt in ***R***.
- Voor elk roosterpunt $\boldsymbol{x = (x_1, x_2, \ldots, x_n)}$ is ook $\boldsymbol{-x = (-x_1, -x_2, \ldots, -x_n)}$ een roosterpunt.
- De *afstand* tussen twee punten $\boldsymbol{x = (x_1, x_2, \ldots, x_n)}$ en $\boldsymbol{y = (y_1, y_2, \ldots, y_n)}$ wordt gegeven door:

$$\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

- Er is een getal ***d (> 0)*** zodanig dat elke twee roosterpunten in ***R*** minimaal een afstand ***d*** uit elkaar liggen.



Hierboven zie je een rooster in 2D. De zwarte stippen zijn alle punten met gehele coördinaten, de punten met een rode markering horen bij het rooster. Het rooster loopt met hetzelfde patroon alle kanten op oneindig ver door.

In de scheikunde kun je ook veel voorbeelden van 3D roosters vinden: kristalroosters.

Een rooster kun je in 2D beschrijven door slechts twee roosterpunten te geven: $\boldsymbol{b_1}$ en $\boldsymbol{b_2}$. We noemen $\boldsymbol{[b_1, b_2]}$ een basis voor het rooster. Elk roosterpunt is een lineaire combinatie van deze twee basispunten: $\boldsymbol{p\, b_1 + q\, b_2}$ waarbij dan $\boldsymbol{p}$ en $\boldsymbol{q}$ twee *gehele* getallen zijn.

Wanneer je als basis de punten $\boldsymbol{b_1}$ = (4, 0) en $\boldsymbol{b_2}$ = (2, 3) neemt kun je andere punten schrijven als zo'n combinatie van de basispunten. Zo geldt $\boldsymbol{2\, b_1 + 0\, b_2}$ = (8, 0) en $\boldsymbol{-b_1 + 2b_2}$ = (0, 6).

Maar een andere basis voor hetzelfde rooster is ook mogelijk. Neem bijvoorbeeld $\boldsymbol{b_1}$ = (6, 3) en $\boldsymbol{b_2}$ = (10, 3). Nu geldt $\boldsymbol{-2b_1 + 2b_2}$ = (8, 0) en $\boldsymbol{5b_1 - 3b_2}$ = (0, 6).

Er zijn oneindig veel basissen mogelijk voor een 2D rooster. Van de twee basissen bij het voorbeeld is de eerste basis beter dan de tweede, omdat de twee basispunten dichter bij de oorsprong liggen dan die in het tweede voorbeeld. In 2D is het makkelijk om voor elk rooster een beschrijving $\boldsymbol{b_1, b_2}$ te vinden waarbij $\boldsymbol{b_1}$ de kleinste afstand tot $\boldsymbol{0}$ heeft. In hogere dimensies is het echter enorm lastig om een basis te vinden met punten die in de buurt van de oorsprong liggen. Dit probleem maakt versleuteling mogelijk: het is makkelijk om een slechte basis vanuit een goede basis te maken, maar het is onmogelijk om vanuit een slechte basis weer een goede te vinden!

De opdrachten

Opgaven C1-C3 zijn gemaakt in oplopende moeilijkheid:

1. Opgave C1 (50 punten te halen) gaat over een rooster in 1 dimensie: de getallenlijn,
2. Opgave C2 (50 punten te halen) gaat over een rooster in het vlak, 2D,
3. Opgave C3 (100 punten te halen) gaat over een rooster van hogere dimensie.

De tijdslimiet voor elk testgeval is 1 seconde.

Opgave C1. Grootste Gemene Deler

In dimensie 1, dus op een (getallen)lijn, zijn er maar 2 mogelijke basissen voor een rooster: ***x*** en ***-x*** zijn allebei basissen voor het rooster dat bestaat uit alle (gehele) veelvouden van ***x***. In deze opgave krijg je geen basis, maar roosterpunten, en jij moet een basis voor het rooster geven.

Schrijf een programma dat van standard input op regel 1 een getal ***N*** leest, en dan op de tweede regel N gehele getallen leest: $x_1, x_2, \ldots, x_N$ gescheiden door een spatie. Het programma moet het grootst mogelijke positieve getal ***d*** naar standard output schrijven zodanig dat elke x_i deelbaar is door ***d***.

Er geldt:

- $1 \leq N \leq 1000$,
- $-10^9 \leq x_i \leq 10^9$.

Voorbeelden

Invoer:
```
5
4 -6 8 -10 12
```

Uitvoer:
```
2
```

Invoer:
```
3
48 24 30
```

Uitvoer:
```
6
```

Opgave C2. 2D-roosters

In deze opgave krijg je een rooster in 2D, en moet je een punt van het rooster vinden dat zo dicht mogelijk bij de oorsprong ligt..

Schrijf een programma dat van standard input leest: b_1 op regel 1, en b_2 op regel 2. Elke regel bestaat uit 2 gehele getallen: ***x*** en ***y***. Jij moet naar uitvoer een roosterpunt schrijven dat niet (0, 0) is en dat de kleinste afstand tot (0, 0) heeft.

Je programma schrijft naar standard output één regel met twee gehele getallen ***x*** en ***y***, gescheiden door een spatie. Het punt ***(x, y)*** moet in het rooster liggen, en niet ***(0, 0)*** zijn. Verder moet $x > 0$ gelden als $y = 0$ en als $x = 0$, dan moet $y > 0$ gelden.

Er geldt $-10^9 \leq x, y \leq 10^9$ voor beide invoerregels.

Voorbeelden

Invoer:
`7 7`
`8 9`
	Invoer:
	`5 25`
	`1 11`
		Invoer:
		`-24 -9`
		`82 29`

Uitvoer: `1 2`	Uitvoer: `3 3`	Uitvoer: `4 5`

Toelichting

In het voorbeeld 1 is het punt $b_2 - b_1 = (8 - 7, 9 - 7) = (1, 2)$ het gevraagde roosterpunt. Dit ligt op een afstand van $\sqrt{5}$ van de oorsprong.

Voorbeeld geval 2 heeft als roosterpunt het dichtst bij de oorsprong :
$b_1 - 2\,b_2 = (5 - 2*1, 25 - 2*11) = (3, 3)$.

Voorbeeld geval 3 heeft als roosterpunt het dichtst bij de oorsprong: $b_1 = (4, 5)$.

Opgave C3. Hogere dimensies

In deze opgave krijg je een rooster in een dimensie $d > 2$, en opnieuw zoeken we een roosterpunt dat zo dicht mogelijk bij de oorsprong ligt.

Schrijf een programma dat van standard input het volgende leest:

- Op regel 1, een geheel getal d,
- Op regel $i+1$ ($1 \le i \le d$) staan er d getallen $b_{i1}, b_{i2}, \ldots, b_{id}$ door spaties gescheiden. Deze regels beschrijven de basispunten $b_i = (b_{i1}, \ldots, b_{id})$.

Je programma schrijft naar standard output één regel met d gehele getallen: $x_1, \ldots, x_d$, gescheiden door spaties. Het punt $(x_1, \ldots, x_d)$ moet een roosterpunt zijn, en niet $(0, \ldots, 0)$ zijn. Verder moet de eerste coördinaat $x_i \neq 0$ die niet nul is, positief zijn: $x_i > 0$.

De jury garandeert voor elk testgeval:

- Er zijn precies twee kortste (niet-nul) roosterpunten;
- Er is geen lineaire vergelijking te vinden tussen de d basis vectoren $b_1, \ldots b_d$, oftewel de basispunten genereren geen rooster dat in een ruimte van $< d$ dimensies ligt.

Er geldt $3 \le d \le 20$ en voor elke coördinaat geldt $-10^9 \le b_{ij} \le 10^9$.

Voorbeelden

Invoer:
```
3
1 0 11
0 1 47
0 0 52
```

Uitvoer:
```
1 2 1
```

Invoer:
```
5
20 11 15 3 8
1 27 29 3 9
11 15 22 21 22
30 21 15 21 20
14 21 7 27 16
```

Uitvoer:
```
2 10 -7 0 3
```

Toelichting

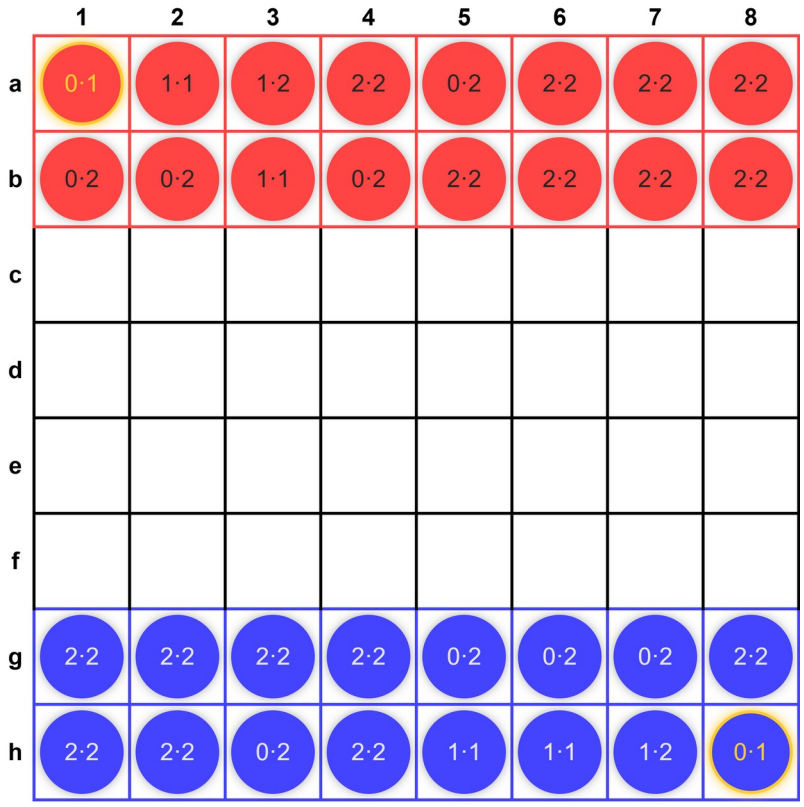
In het eerste voorbeeld kun je controleren: $(1, 2, 1) = b_1 + 2\,b_2 - 2\,b_3$. Dit is het kortste roosterpunt.

In het tweede voorbeeld kun je controleren: $(2, 10, -7, 0, 3) = -4\,b_1 + b_2 - b_3 + 4b_4 - 2b_5$. Dit is het kortste roosterpunt.

Opgave D. Zero Point One (0.1)

Zero Point One is een strategiespel voor twee spelers, aangeduid met ROOD en blauw. Het spel is in 2012 ontworpen en geïllustreerd door Jim Wickson en is gepubliceerd op BoardGameGeek.com.

Het spel wordt gespeeld op een vierkant bord van 8x8 vakjes.



Het spelbord met een mogelijke beginopstelling

Alle stukken hebben twee kanten met hetzelfde opschrift maar ze zijn aan de ene kant rood en aan de andere kant blauw.

Aan het begin van het spel hebben de spelers ROOD en blauw ieder de volgende 16 stukken:

- 1 x Wazir (`w` of `W`)
- 1 x Knight (`n` of `N`)
- 2 x Ferzes (`f` of `F`)
- 4 x Dabbabas (`d` of `D`)
- 8 x Alfils (`a` of `A`)

De stukken waarmee ROOD speelt worden aangegeven als: `WNFDA`
De stukken waarmee blauw speelt worden aangegeven als: `wnfda`
Let op: de Knight wordt aangeduid met een `n` of `N`.

Het maken van de opstelling
ROOD plaatst zijn stukken in een zelfgekozen volgorde in de vakjes op de rijen a en b. Daarna plaatst blauw zijn stukken op een zelfgekozen volgorde in de vakjes op de rijen g en h.

Het spel
Spelers doen om beurten een zet. ROOD begint. Een speler verplaatst één eigen stuk op basis van de specifieke bewegingsregels, die ook op het stuk staan aangegeven. Je mag een stuk niet zetten op een veld waar al een stuk staat van je eigen kleur.

Bewegingsregels
Op alle stukken staan twee cijfers. Die cijfers geven aan hoeveel stappen het stuk horizontaal (H) en/of verticaal (V) je bij elke zet moet doen, de volgorde maakt niet uit.

naam	zet
Wazir (0-1)	verplaats 1 stap H **of** verplaats 1 stap V.
Knight (1-2)	verplaats 1 stap H en 2 stappen V **of** verplaats 2 stappen H en 1 stap V. De Knight mag over andere stukken springen.
Ferz (1-1)	verplaats 1 stap H en 1 stap V.
Dabbaba (0-2)	verplaats 2 stappen H **of** verplaats 2 stappen V. De Dabbaba mag over een ander stuk springen.
Alfil (2-2)	verplaats 2 stappen H en 2 stappen V. De Alfil mag over andere stukken springen.

Slaan van stukken
Als je zet op een vakje terecht komt waar een stuk van de tegenstander staat wordt dat stuk geslagen. Het geslagen stuk wordt van het bord genomen; het maakt nu deel uit van de stukken van degene die de steen geslagen had (en neemt dus ook die kleur aan).
Als een speler de Wazir (0-1) van de tegenstander slaat heeft hij het spel gewonnen.

Inzetten van geslagen stukken
In plaats van het bewegen van een stuk kan de speler ook een stuk dat hij van de tegenstander had geslagen weer op het bord zetten. Dat stuk mag op ieder leeg veld van het bord worden geplaatst.

Winst
Het spel eindigt als een speler de Wazir (0-1) van zijn tegenstander slaat. Daarmee heeft hij het spel gewonnen.

Meer informatie over het spel kun je vinden op:
https://boardgamegeek.com/boardgame/114307/01-zero-point-one.

Protocol:
Om met de jurysoftware te kunnen communiceren moet je het protocol volgen. De speler leest informatie van standard input en voert zaken uit naar standard output. Voor meer informatie hierover raadpleeg je de Technical Rules op www.codecup.nl. Vergeet niet om je uitvoer te flushen.

ROOD krijgt als eerste invoer "`Start`", waardoor hij weet dat hij als ROOD moet spelen. Daarop moet hij zijn startopstelling uitvoeren.
Bij de startopstelling van ROOD worden de 16 hoofdletters van de stukken op één regel ingevoerd. De eerste 8 stukken worden geplaatst op rij `a`, `a1` t/m rij `a8`, de laatste 8 stukken worden geplaatst op `b1` t/m `b8`.
De regel met de rode startopstelling is de eerste invoer voor blauw. Deze speler begrijpt hieruit dat hij blauw speelt en hij moet nu de blauwe startopstelling uitvoeren.
Bij de startopstelling van blauw worden de 16 kleine letters van de stukken ingevoerd. De eerste 8 stukken worden geplaatst op rij `g`, in de vakjes `g1` t/m `g8`, de laatste 8 stukken worden geplaatst op `h1` t/m `h8`.

ROOD krijgt die startopstelling weer te zien als invoer en voert vervolgens zijn eerste zet uit. Blauw leest die zet in, voert zelf een zet uit en zo gaat het verder.

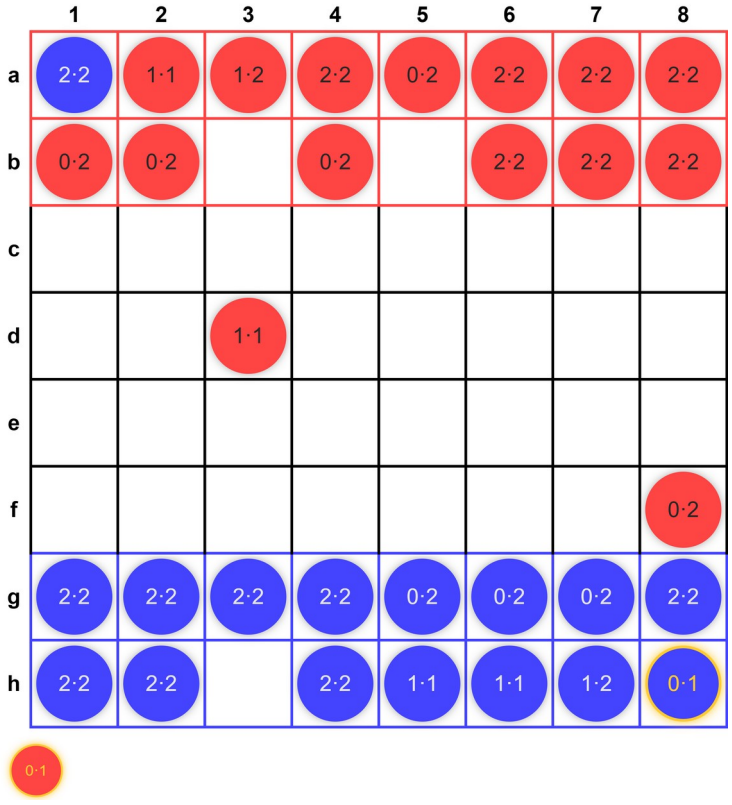
Een gewone zet heeft de vorm `b5d3`, dat betekent dat het stuk van `b5` naar `d3` wordt geplaatst. Als een geslagen stuk opnieuw wordt ingezet gebruik je voor een rode Dabbaba (0-2) een `D` plus de naam van het vakje, iets als `Df8`: een rode Dabbaba wordt nu op veld `f8` geplaatst; en voor een blauwe Alfil (2-2) bijvoorbeeld `ac3`, waarmee de Alfil op veld `c3` terecht komt.

Als er na 100 zetten, dat wil zeggen 50 zetten per speler, geen winnaar is eindigt het spel onbeslist. Beide programma's hebben 30 seconden bedenktijd voor hun wedstrijd; de klok staat stil als de speler niet aan de beurt van spelen is.

Voorbeeld:

ROOD:		**blauw:**	
Invoer	**Uitvoer**	**Invoer**	**Ultvoer**
"Start"			
	WFNADAAADDFDAAAA		
		"WFNASAAADDFDAAAA"	
			aaaadddaaadaffnw
"aaaadddaaadaffnw"			
	b5d3		
		"b5d3"	
			h3f3
"h3f3"			
	b3c2		
		"b3c2"	
			f3d3
"f3d3"			
	c2d3		
		"c2d3"	
			ac3
"ac3"			
	Df8		
		"Df8"	
			c3a1
"Quit"			
		"Quit"	

De beginpositie bij dit voorbeeldspel staat bij de inleiding op de spelregels aangegeven.



Dit is de eindpositie. Blauw wint dit spel (maar er werd niet slim gespeeld).

Puntentelling voor de CodeCup-competitie
De winnaar van een spel krijgt 21 punten, de verliezer krijgt 1 punt. Als de wedstrijd onbeslist blijft krijgen beide spelers 11 punten.

Als een speler een fout maakt, crasht, het spel te vroeg verlaat, of de bedenktijd overschrijdt krijgt hij van de jurysoftware een “`Quit`” als invoer, verliest hij het spel en ontvangt hij 0 punten. In dat geval krijgt de tegenstander ook een “`Quit`” als invoer en wint hij het spel. De uitslag is in zo’n geval 21-0 (in plaats van 21-1).

Als je programma als invoer de regel "`Quit`" krijgt moet je programma worden afgesloten.

Je programma mag informatie wegschrijven naar standard error. Je kunt, als het spel gespeeld is, na een testcompetitie gebruik maken van deze informatie.

Competitie
In de competitie tref je iedere tegenstander tweemaal, eenmaal als ROOD en eenmaal als blauw. De competitie wordt gewonnen door de speler die het meeste aantal punten verzamelt tijdens de competitie. Meer informatie hierover is te vinden op www.codecup.nl in de Competition rules.

Punten voor de olympiade

Als je programma samenwerkt met onze jurysoftware wordt het toegelaten tot het toernooi. In dat geval verdien je 20 punten voor deze opgave. Als je programma zonder fouten speelt kun je daarmee nog eens 50 punten verdienen.

De uitslag van dat toernooi is bepalend voor de laatste 30 punten; hoe hoger je programma eindigt, des te meer punten krijg je.

Inzenden van je programma gaat op www.codecup.nl waar je eerst een account moet aanmaken.