be-OI 2018 **Finale - BELOFTEN** zaterdag 17 maart 2018	**Invullen in HOOFDLETTERS en LEESBAAR aub** VOORNAAM : ..... NAAM : ..... SCHOOL :	O **Gereserveerd**

Belgische Informatica-olympiade (duur : 2u maximum)

Dit is de vragenlijst van de finale van de Belgische Informatica-olympiade 2018. Ze bevat 7 vragen, en je krijgt **maximum 2u** de tijd om ze op te lossen.

Algemene opmerkingen (lees dit aandachtig voor je begint)

1. Controleer of je de juiste versie van de vragen hebt gekregen (die staat hierboven in de hoofding).
   - De categorie **beloften** is voor leerlingen tot en met het 2e middelbaar,
   - de categorie **junior** is voor het 3e en 4e middelbaar,
   - de categorie **senior** is voor het 5e middelbaar en hoger.
2. Vul duidelijk je voornaam, naam en school in, **alleen op dit eerste blad**.
3. Jouw antwoorden moet je invullen op de daarop voorziene antwoordbladen, die je achteraan vindt.
4. Als je door een fout buiten de antwoordkaders moet schrijven, schrijf dan alleen verder op hetzelfde blad papier (desnoods op de achterkant).
5. Schrijf **duidelijk leesbaar** met blauwe of zwarte **pen of balpen**.
6. Je mag alleen schrijfgerief bij je hebben. Rekentoestel, GSM, ... zijn **verboden**.
7. Je mag altijd extra kladpapier vragen aan de toezichthouder of leerkracht.
8. Wanneer je gedaan hebt, geef je deze eerste bladzijde terug (met jouw naam erop), en de pagina's met jouw antwoorden. Al de rest mag je bijhouden.
9. Voor alle code in de opgaven werd **pseudo-code** gebruikt. Op de volgende bladzijde vind je een **beschrijving** van de pseudo-code die we hier gebruiken.
10. Als je moet antwoorden met code, mag dat in **pseudo-code** of in eender welke **courante programmeertaal** (zoals Java, C, C++, Pascal, Python, ...). We trekken geen punten af voor syntaxfouten.

Veel succes!

Overzicht pseudo-code

Gegevens worden opgeslagen in variabelen. Je kan de waarde van een variabele veranderen met $\leftarrow$. In een variabele kunnen we gehele getallen, reële getallen of arrays opslaan (zie verder), en ook booleaanse (logische) waarden : waar/juist (**`true`**) of onwaar/fout (**`false`**). Op variabelen kan je wiskundige bewerkingen uitvoeren. Naast de klassieke operatoren $+$, $-$, $\times$ en $/$, kan je ook % gebruiken: als a en b allebei gehele getallen zijn, dan zijn a/b en $a\%b$ respectievelijk het quotiënt en de rest van de gehele deling (staartdeling). Bijvoorbeeld, als $a = 14$ en $b = 3$, dan geldt: $a/b = 4$ en $a\%b = 2$. In het volgende stukje code krijgt de variabele *leeftijd* de waarde 17.

```
geboortejaar ← 2001
leeftijd ← 2018 − geboortejaar
```

Als we een stuk code alleen willen uitvoeren als aan een bepaalde voorwaarde (conditie) is voldaan, gebruiken we de instructie **`if`** . We kunnen eventueel code toevoegen die uitgevoerd wordt in het andere geval, met de instructie **`else`**. Het voorbeeld hieronder test of iemand meerderjarig is, en bewaart de prijs van zijn/haar cinematicket in een variabele *prijs*. De code is bovendien voorzien van commentaar.

```
if (leeftijd ≥ 18)
{
  prijs ← 8    // Dit is een stukje commentaar
}
else
{
  prijs ← 6    // Goedkoper!
}
```

Soms, als een voorwaarde onwaar is, willen we er nog een andere controleren. Daarvoor kunnen we **`else if`** gebruiken, wat neerkomt op het uitvoeren van een andere **`if`** binnen in de **`else`** van de eerste **`if`**. In het volgende voorbeeld zijn er 3 leeftijdscategorieën voor cinematickets.

```
if (leeftijd ≥ 18)
{
    prijs ← 8 // Prijs voor een volwassene.
}
else if (leeftijd ≥ 6)
{
    prijs ← 6 // Prijs voor een kind van 6 of ouder.
}
else
{
    prijs ← 0 // Gratis voor kinderen jonger dan 6.
}
```

Wanneer we in één variabele tegelijk meerdere waarden willen stoppen, gebruiken we een array. De afzonderlijke elementen van een array worden aangeduid met een index (die we tussen vierkante haakjes schrijven achter de naam van de array). Het eerste element van een array arr heeft index 0 en wordt genoteerd als $arr[0]$. Het volgende element heeft index 1, en het laatste heeft index $N - 1$ als de array N elementen bevat. Dus als de array arr de drie getallen 5, 9 en 12 bevat (in die volgorde) dan is $arr[0] = 5$, $arr[1] = 9$ en $arr[2] = 12$. De lengte van arr is 3, maar de hoogst mogelijke index is slechts 2.

Voor het herhalen van code, bijvoorbeeld om de elementen van een array af te lopen, kan je een **for**-lus gebruiken. De notatie **for** ($i \leftarrow a$ **to** b **step** k) staat voor een lus die herhaald wordt zolang $i \leq b$, waarbij i begint met de waarde a en telkens verhoogd wordt met k aan het eind van elke stap. Het onderstaande voorbeeld berekent de som van de elementen van de array arr, veronderstellend dat de lengte ervan N is. Nadat het algoritme werd uitgevoerd, zal de som zich in de variabele sum bevinden.

```
sum ← 0
for (i ← 0 to N − 1 step 1)
{
   sum ← sum + arr[i]
}
```

Een alternatief voor een herhaling is een **while**-lus. Deze herhaalt een blok code zolang er aan een bepaalde voorwaarde is voldaan. In het volgende voorbeeld delen we een positief geheel getal N door 2, daarna door 3, daarna door 4 ... totdat het getal nog maar uit 1 decimaal cijfer bestaat (d.w.z., kleiner wordt dan 10).

```
d ← 2
while (N ≥ 10)
{
   N ← N/d
   d ← d + 1
}
```

We tonen algoritmes vaak in een kader met wat extra uitleg. Na **Input**, definiëren we alle parameters (variabelen) die gegeven zijn bij het begin van het algoritme. Na **Output**, definiëren we de staat van bepaalde variabelen nadat het algoritme is uitgevoerd, en eventueel de waarde die wordt teruggegeven. Een waarde teruggeven doe je met de instructie **return**. Zodra **return** wordt uitgevoerd, stopt het algoritme en wordt de opgegeven waarde teruggegeven.

Dit voorbeeld toont hoe je de som van alle elementen van een array kan berekenen.

```
Input  : arr, een array van N getallen.
         N, het aantal elementen van de array.
Output : sum, de som van alle getallen in de array.

for (i ← 0 to N − 1 step 1)
{
    sum ← sum + arr[i]
}
return sum
```

Opmerking: in dit laatste voorbeeld wordt de variabele i enkel gebruikt om de tel bij te houden van de **for**-lus. Er is dus geen uitleg voor nodig bij **Input** of **Output**, en de waarde ervan wordt niet teruggegeven.

Vraag 1 – Voortplanting

Uitleg

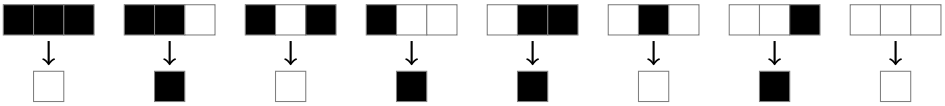
De array hieronder toont de staat van een rijtje eencellige organismen naast elkaar, op een bepaald moment. Een cel wordt weergegeven door een zwart vakje als ze leeft, of door een wit vakje als ze dood is.

De staat van een cel (levend of dood) evolueert van de ene generatie naar de volgende via eenvoudige regels, die enkel afhangen van haar huidige staat en die van haar twee buurcellen (de linker en de rechter) in de vorige generatie. Voorbij de uiteinden aan de linker- en rechterkant, mag je aannemen dat alle cellen altijd dood blijven.

Een regel kan bijvoorbeeld zijn:

- Een cel met exact één levende buurcel wordt (of blijft) levend.
- Anders sterft de cel of blijft ze dood.

Die regel kunnen we als volgt weergeven:

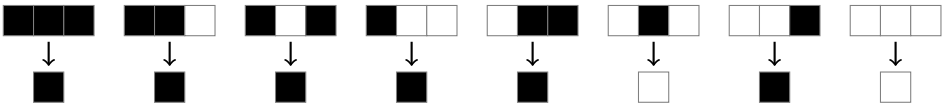


Als we die regel toepassen, dan wordt de 2e generatie van ons voorbeeld:

Q1(a) [2 ptn]	**Als je dezelfde regel volgt, hoe ziet de 3e generatie er dan uit?** **(Zoek hier naar de oplossing voordat je ze definitief invult op het antwoordblad.)**

Een andere regel

Stel dat je de volgende regel gebruikt:



en de situatie van vertrek (de 1e generatie) is:

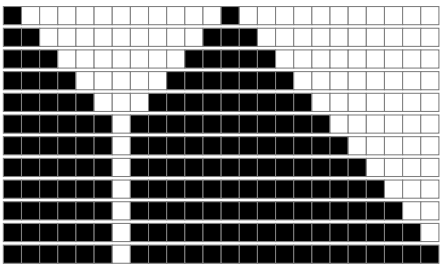
Q1(b) [2 ptn]	**Wat zal de 3e generatie zijn met deze nieuwe regel?** **(Zoek hier naar de oplossing voordat je ze definitief invult op het antwoordblad.)**

Q1(c) [2 ptn]	**En wat zal de 12e generatie zijn met deze nieuwe regel?** **(Zoek hier naar de oplossing voordat je ze definitief invult op het antwoordblad.)**

Vind de regel

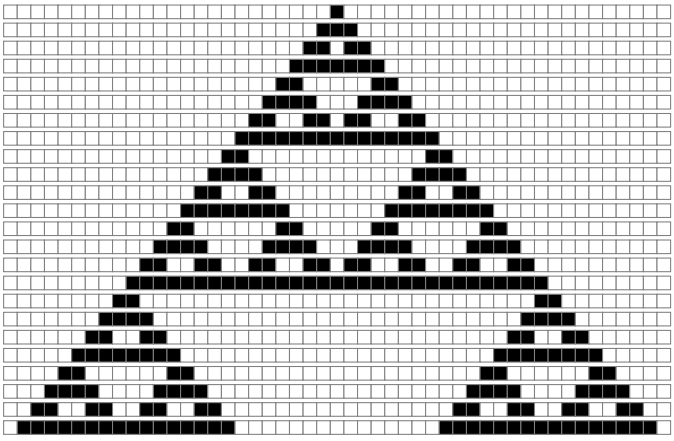
We geven de generaties onder elkaar weer. De 1e lijn is de vertreksituatie, de lijnen daaronder komen overeen met de daaropvolgende generaties. Zo krijgen we een mooie grafische voorstelling van een evolutie.

Bijvoorbeeld:



Q1(d) [2 ptn]	**Welke regel komt overeen met dit resultaat?** **(Zoek hier naar de oplossing voordat je ze definitief invult op het antwoordblad.)**

Hier is nog een grafische voorstelling van een evolutie die een andere regel volgt:



Q1(e) [2 ptn]	**Welke regel komt overeen met dit resultaat?** **(Zoek hier naar de oplossing voordat je ze definitief invult op het antwoordblad.)**

Vraag 2 – Voetbal

Bij een voetbaltoernooi speelt elke ploeg 2 keer tegen elke andere ploeg: 1 keer thuis en 1 keer op verplaatsing bij de tegenstander. Er zijn n ploegen, genummerd van 0 tot $n-1$, en de resultaten van de matches worden opgeslagen in 2 arrays met grootte $n \times n$.

Als ploeg i ploeg j ontvangt, dan wordt het aantal goals gescoord door ploeg i opgeslagen in $A[i][j]$, en het aantal goals gescoord door ploeg j opgeslagen in $B[i][j]$. Dat heeft natuurlijk enkel zin als $i \neq j$ (want een ploeg kan niet tegen zichzelf spelen), maar de arrays worden geïnitialiseerd zodat voor elke i geldt: $A[i][i] = B[i][i] = 0$.

Dit zijn bijvoorbeeld de resultaten van een toernooi met $n = 4$ ploegen.

A[ ][ ]

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

B[ ][ ]

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

$A[2][3] = 4$ en $B[2][3] = 0$, dus ploeg 2 heeft met 4 - 0 gewonnen toen ze ploeg 3 ontving.
$A[0][1] = B[0][1] = 2$, dus ploeg 1 heeft met 2 - 2 gelijk gespeeld op verplaatsing bij ploeg 0.

Het toernooi is afgelopen en alle resultaten van alle matches zijn bekend. Vervolledig de volgende algoritmes door de `...` in te vullen, zodat ze de gevraagde resultaten geven. Je mag vrij gebruik maken van de variabele n die het aantal ploegen bevat, en van de arrays $A[\,][\,]$ en $B[\,][\,]$ met daarin de resultaten.

Algoritme 1X2 : Analyseer de resultaten van een match en geef `"1"` terug als de thuisploeg heeft gewonnen, of geef `"2"` terug als de bezoekende ploeg heeft gewonnen. Bij gelijkspel, geef je `"X"` terug.

```
Input  : i het nummer van de thuisploeg, 0 ≤ i < n.
         j het nummer van de bezoekende ploeg, 0 ≤ j < n.
Output : "1", "2" of "X" al naargelang de winnaar.

if ( ... )              // (a)
{
    return "1"
}
else if ( ... )         // (b)
{
    return "X"
}
else
{
    return "2"
}
```

Q2(a) [2 ptn]	**Wat moet expressie (a) zijn in algoritme 1X2 ?**

Q2(b) [2 ptn]	**Wat moet expressie (b) zijn in algoritme 1X2 ?**

Algoritme Goals : Bereken het totaal aantal doelpunten gescoord door een ploeg, minus het totaal aantal doelpunten dat de ploeg tegen heeft gekregen, in de loop van het toernooi.

```
Input  : k, het nummer van de ploeg, 0 ≤ k < n.
Output : g, het aantal gescoorde doelpunten minus het aantal doelpunten tegen,
          voor ploeg k in het hele toernooi.

g ← 0
for (i ← 0 to n − 1 step 1)
{
    g ← ...                    //(c)
}
return g
```

Q2(c) [4 ptn]	**Wat moet expressie (c) zijn in algoritme Goals ?**

Algoritme Overwinningen : Bereken het aantal overwinningen van een ploeg in het hele toernooi.

```
Input  : k het nummer van de ploeg, 0 ≤ k < n.
Output : r, het aantal overwinningen van ploeg k in het hele toernooi.

r ← 0
for (i ← 0 to n − 1 step 1)
{
    if ( A[k][i] ... )          //(d)
    {
        r ← r + 1
    }
    if ( ... )                  //(e)
    {
        r ← r + 1
    }

}
return r
```

Q2(d) [2 ptn]	**Vervolledig expressie (d) in algoritme Overwinningen.**

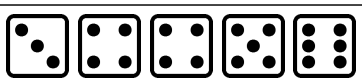
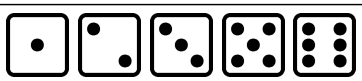
Q2(e) [2 ptn]	**Wat moet expressie (e) zijn in algoritme Overwinningen ?**

Vraag 3 – Dobbelspel

Een programma "gooit" 5 dobbelstenen en controleert of er een speciale combinatie voorkomt. Het kiest willekeurig 5 getallen tussen 1 en 6 en zet ze, van klein naar groot gesorteerd, in de vakjes met indices 1 tot 5 van array $d[\,]$.

Bijvoorbeeld, als de gegenereerde dobbelstenen zijn: 5, 1, 6, 5 en 1 , dan is $d[1] = 1$, $d[2] = 1$, $d[3] = 5$, $d[4] = 5$ en $d[5] = 6$. Er geldt dus altijd: $1 \leq d[1] \leq d[2] \leq d[3] \leq d[4] \leq d[5] \leq 6$.

De volgende tabel toont de combinaties waarop we willen testen:

Combinatie	**Beschrijving**	**Voorbeeld**
Yahtzee	5 identieke dobbelstenen	
Carré	Minstens 4 identieke dobbelstenen	
Full House	3 identieke dobbelstenen + de andere 2 ook identiek	
Trio	Minstens 3 identieke dobbelstenen	
Paar	Minstens 2 identieke dobbelstenen	
Dubbel Paar	2 niet-overlappende Paren	
Grote straat	5 opeenvolgende waarden	
Kleine straat	Minstens 4 opeenvolgende waarden	
Niets	Geen van alle vorige	

Let op: voor hetzelfde resultaat zijn er soms verschillende combinaties geldig. Een *Yahtzee* is ook een *Carré*, een *Full House*, een *Trio*, een *Dubbel Paar* en een *Paar*. Op dezelfde manier is een *Grote straat* ook een *Kleine straat*, en in het voorbeeld hierboven bevat de *Kleine straat* ook een *Paar*.

In de meeste programmeertalen bestaan de instructies uit Engelse woorden. We gebruiken **true** (waar), **false** (niet waar), **or** (of), **and** (en).
Als we verschillende voorwaarden testen met **or**, dan is de test **true** als minstens één van de voorwaarden **true** is.
Als we verschillende voorwaarden testen met **and**, dan is de test **true** als alle voorwaarden **true** zijn.

Bijvoorbeeld, als $lengte = 180$ en $leeftijd = 16$, dan:

$\big((lengte = 155)$ **or** $(leeftijd = 12)\big)$ is **false**	$\big((lengte = 155)$ **and** $(leeftijd = 12)\big)$ is **false**
$\big((lengte = 155)$ **or** $(leeftijd = 16)\big)$ is **true**	$\big((lengte = 155)$ **and** $(leeftijd = 16)\big)$ is **false**
$\big((lengte = 180)$ **or** $(leeftijd = 16)\big)$ is **true**	$\big((lengte = 180)$ **and** $(leeftijd = 16)\big)$ is **true**

Zijn er haakjes, dan werk je zoals gewoonlijk eerst de binnenste haakjes uit, en daarna de buitenste.

Om punten te halen op de vragen hieronder, moet je ALLE mogelijke combinaties aankruisen waarvoor de gegeven test altijd **true** is, en ZEKER GEEN combinatie aankruisen waarvoor de test soms **false** is.

Als bijvoorbeeld een test **true** is voor elke *Trio* en elke *Full House*, dan kruis je de vakjes *Trio* en *Full House* aan. Als de test **false** is voor minstens één *Carré* dan mag je het vakje *Carré* niet aankruisen.

Q3(a) [1 pt]

$(d[1] = d[5])$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q3(b) [1 pt]

$(d[1] = d[4])$ **or** $(d[2] = d[5])$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q3(c) [1 pt]

$(d[1] = d[2])$ **or** $(d[2] = d[3])$ **or** $(d[3] = d[4])$ **or** $(d[4] = d[5])$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q3(d) [1 pt]

$(d[1] = d[3])$ **or** $(d[2] = d[4])$ **or** $(d[3] = d[5])$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q3(e) [2 ptn]

$\Big((d[1] = d[2])$ **and** $(d[3] = d[5])\Big)$ **or** $\Big((d[1] = d[3])$ **and** $(d[4] = d[5])\Big)$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q3(f) [2 ptn]

$(d[1] \neq d[2])$ **and** $(d[2] \neq d[3])$ **and** $(d[3] \neq d[4])$ **and** $(d[4] \neq d[5])$ **and** $(d[5] - d[1] = 4)$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Vraag 4 – De mier van Langton

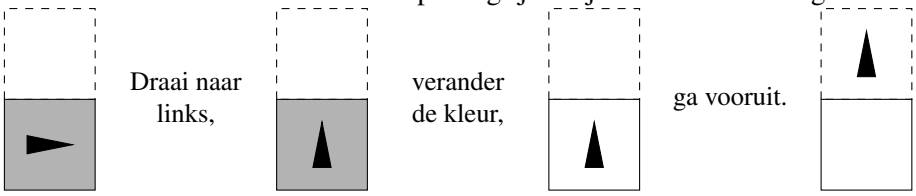
De vakjes van een rooster kunnen wit of grijs zijn. Een mier staat op een vakje en verplaatst zich, één vakje per keer, naar links, rechts, boven of onder, volgens deze regels:

- Als de mier op een wit vakje staat, draait ze een kwartslag naar rechts, verandert ze de kleur van het vakje (naar grijs), en gaat ze een vakje vooruit.
- Als de mier op een grijs vakje staat, draait ze een kwartslag naar links, verandert ze de kleur van het vakje (naar wit), en gaat ze een vakje vooruit.

In de volgende voorbeelden stellen we de mier voor als zwarte driehoek, met de scherpste hoek als kop die de richting aangeeft. Het vakje in stippellijnen kan wit of grijs zijn, de kleur ervan is niet belangrijk.

Eerste voorbeeld: de mier staat op een wit vakje en is naar rechts gericht.

Tweede voorbeeld: de mier staat op een grijs vakje en is naar rechts gericht.



Vervolledig de code op de volgende pagina die het gedrag van de mier simuleert, voor k stappen op een rooster met grootte $n \times m$ dat we opslaan in een array $grid[\,][\,]$ waarvan de elementen enkel 0 (= wit) of 1 (= grijs) kunnen zijn.

- De eerste index van de array is het nummer van de kolom, gaande van 0 links tot $n-1$ rechts.
- De tweede index van de array is het nummer van de rij, gaande van 0 aan de onderkant tot $m-1$ aan de bovenkant.

Dus als $grid[3][7] = 1$, dan is het vakje op kolom 3 (de vierde van links) en rij 7 (de achtste vanaf beneden) grijs.

We slaan de positie van de mier op in variabelen x voor de kolom en y voor de rij. De richting van de mier wordt opgeslagen in variabelen dx en dy die elk een waarde kunnen hebben van 0, 1 of -1. De variabele dx geeft aan hoe het nummer van de kolom van de mier verandert als zij vooruitgaat. Op dezelfde manier geeft dy aan hoe het nummer van de rij verandert als zij vooruitgaat.

- als de mier naar boven gaat dan is $dx = 0$ en $dy = 1$,
- als de mier naar rechts gaat dan is $dx = 1$ en $dy = 0$
- als de mier naar onder gaat dan is $dx = 0$ en $dy = -1$,
- als de mier naar links gaat dan is $dx = -1$ en $dy = 0$

Het rooster is een cylinder:

- De mier kan links en rechts voorbij de grens van het rooster gaan.
  Als ze dat doet, verschijnt ze terug op dezelfde rij aan de andere kant van het rooster.
- Maar: de mier kan niet voorbij de boven- en onderkant van het rooster.
  Als ze dat probeert, blijft ze geblokkeerd op hetzelfde vakje en maakt ze rechtsomkeert (180 graden).

Vervolledig het algoritme door de ontbrekende expressies en instructies in te vullen.

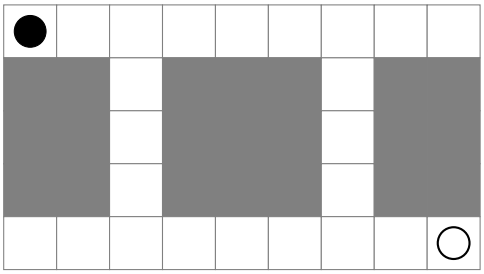
```
Input  : k, een positief geheel getal, het aantal te simuleren stappen.
         grid, een tweedimensionale array die enkel nullen en eenen bevat.
         n, het aantal kolommen van array grid.
         m, het aantal rijen van array grid.
         x, de kolom waarin de mier start, 0 ≤ x ≤ n−1.
         y, de rij waarin de mier start, 0 ≤ y ≤ m−1.
         dx en dy, 2 waarden (0, 1 of −1), de richting waarin de mier start.
Output : de array grid is gewijzigd door k stappen van de mier.

for (i ← 1 to ... step 1)                    //(a)
{
    temp ← dx
    if (grid[x][y] = ... )                   //(b)
    {
        dx ← dy
        dy ← -temp
        grid[x][y] ← ...                     //(c)
    }
    else
    {
        dx ← ...                             //(d)
        dy ← ...                             //(e)
        ...                                  //(f)
    }

    x ← x + dx
    if (x < 0 )
    {
        x ← ...                              //(g)
    }
    else if (x = ... )                       //(h)
    {
        ...                                  //(i)
    }

    ...                                      //(j)
    if ((y < 0 ) or (y = ... ))              //(k)
    {
        y ← y - ...                          //(l)
        dy ← ...                             //(m)
    }
}
return ← grid
```

Q4(a) [1 pt]	**Wat is expressie (a) ?**
Q4(b) [1 pt]	**Wat is expressie (b) ?**
Q4(c) [1 pt]	**Wat is expressie (c) ?**
Q4(d) [1 pt]	**Wat is expressie (d) ?**
Q4(e) [1 pt]	**Wat is expressie (e) ?**
Q4(f) [1 pt]	**Wat is instructie (f) ?**
Q4(g) [1 pt]	**Wat is expressie (g) ?**
Q4(h) [1 pt]	**Wat is expressie (h) ?**
Q4(i) [1 pt]	**Wat is instructie (i) ?**
Q4(j) [1 pt]	**Wat is instructie (j) ?**
Q4(k) [1 pt]	**Wat is expressie (k) ?**
Q4(l) [1 pt]	**Wat is expressie (l) ?**
Q4(m) [1 pt]	**Wat is expressie (m) ?**

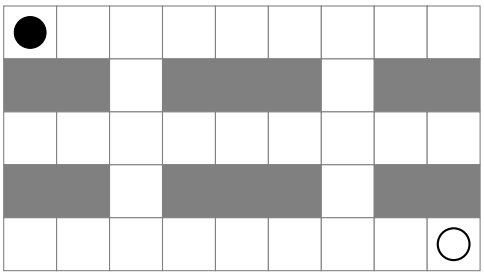
Vraag 5 – Alle wegen leiden naar ...

De tekeningen hieronder zijn tonen je kamers met vierkantjes vrije ruimte (witte vakjes) en obstakels (grijze vakjes). Een robot (zwarte cirkel) staat op het vakje linksboven en moet zich verplaatsen naar zijn doel (witte cirkel) rechtsonder. De robot is beschadigd en kan zich alleen naar rechts of naar onder bewegen. Hij kan nooit naar links of naar boven gaan. De robot moet op de witte vakjes blijven. Hij kan niet over de grijze obstakels lopen. Hoeveel verschillende routes kan de robot nemen om zijn doel te bereiken?

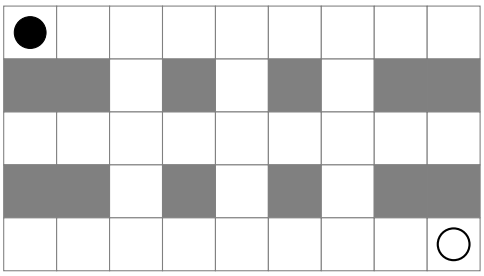
Q5(a) [1 pt]	**Hoeveel verschillende routes kan de robot nemen?**



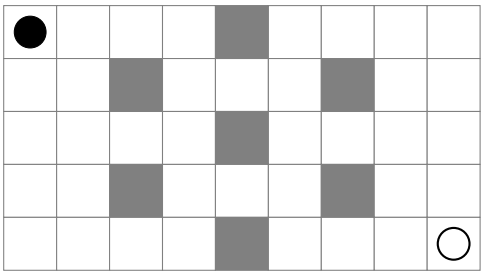
Q5(b) [1 pt]	**Hoeveel verschillende routes kan de robot nemen?**



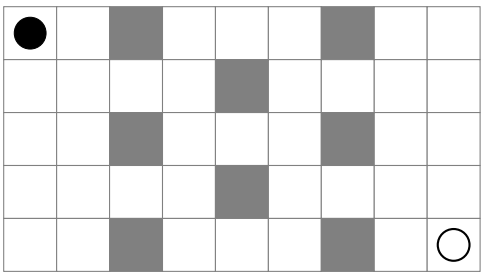
Q5(c) [2 ptn]	**Hoeveel verschillende routes kan de robot nemen?**



Q5(d) [2 ptn]	**Hoeveel verschillende routes kan de robot nemen?**



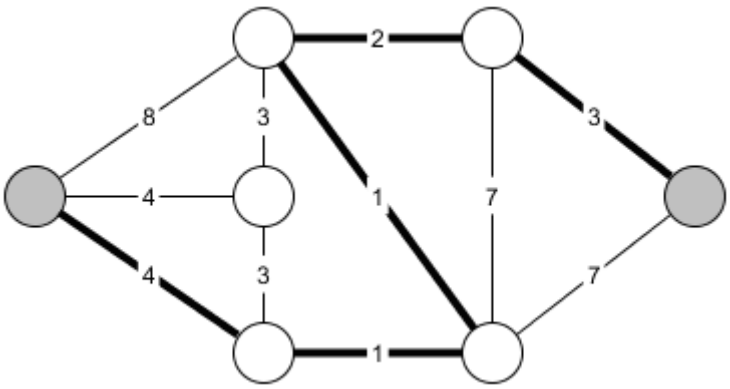
Q5(e) [2 ptn]	**Hoeveel verschillende routes kan de robot nemen?**



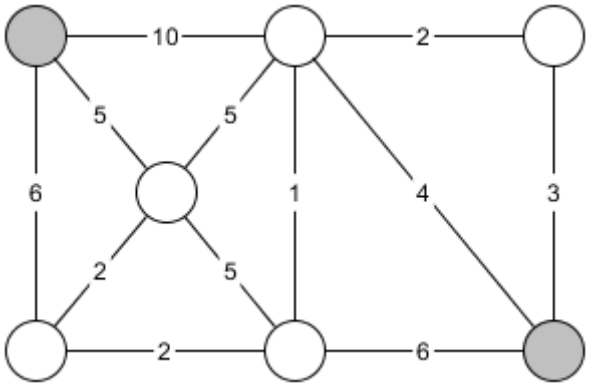
Vraag 6 – De kortste weg

Een *graaf* is een verzameling nodes (de cirkels) met verbindingen (de lijnen) ertussen. In deze vraag kijken we naar grafen waarbij elke verbinding een lengte heeft (genoteerd op de verbinding). De lengte van een verbinding heeft niets te maken met de lengte van de lijn waarmee ze getekend is.

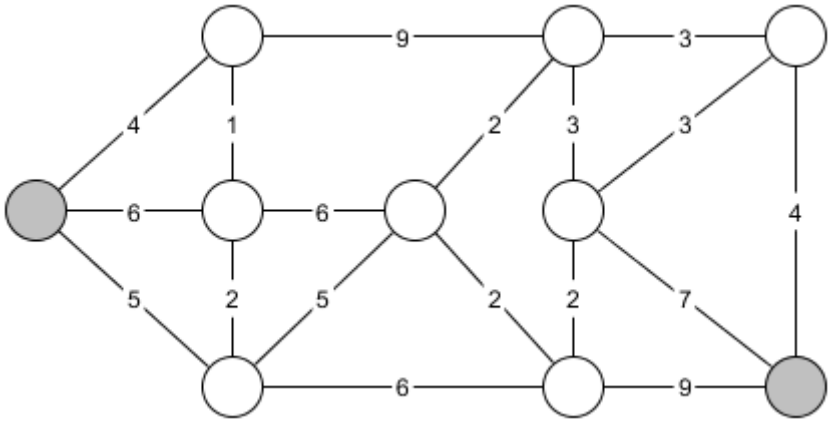
Een pad tussen 2 nodes is een reeks verbindingen die we kunnen volgen om van de ene naar de andere node te gaan. De lengte van een pad is de som van de lengtes van alle verbindingen die deel uitmaken van het pad. De kortste afstand tussen twee nodes is de lengte van het kortste pad ertussen.



In het voorbeeld hierboven, hebben we het kortste pad tussen de twee grijze nodes vet gemarkeerd. De kortste afstand tussen die twee nodes is de som van alle lengtes op dit pad, dus gelijk aan $4 + 1 + 1 + 2 + 3 = 11$.

Q6(a) [4 ptn]	**Wat is de kortste afstand tussen de 2 grijze nodes op deze graaf met 7 nodes?** 

Q6(b) [4 ptn]	**Wat is de kortste afstand tussen de 2 grijze nodes op deze graaf met 10 nodes?**



Vraag 7 – Barcodes kopiëren

Je werkt op de etiketten-afdeling van een supermarkt, en jouw taak is om barcodes te kopiëren. Die bestaan uit een aantal witte en zwarte vakjes naast elkaar, bijvoorbeeld:

De mishandelde kopieermachine ziet helaas niet meer zo scherp. Ze kan de volgende fouten maken: als in de originele barcode een vakje is omgeven door twee vakjes van de andere kleur, kan dat vakje die kleur krijgen in de kopie. De vakjes aan de uiteinden kunnen een andere kleur krijgen als het enige naastliggende vakje de andere kleur heeft in het origineel. In alle andere gevallen blijft de originele kleur behouden.

In het voorbeeld van hierboven, kan het 2e vakje, dat wit is, zwart worden, want het 1e en 3e zijn allebei zwart:

maar het 4e kan niet zwart worden want het 5e is wit.

Het 6e vakje, dat zwart is, kan wit worden:

Let op: verschillende fouten kunnen tegelijk voorkomen, zodat ons eerste voorbeeld ook zou kunnen worden:

Voorspellen

In de volgende vragen geven we je een originele barcode. De vraag is altijd: hoeveel verschillende kopies zijn er mogelijk?

Bijvoorbeeld, als dit de originele barcode is:

Dan zijn de enige mogelijke kopies:

 en 

dus is het antwoord 2.

Q7(a) [1 pt]	**Gegeven dit origineel, hoeveel verschillende kopies zijn er mogelijk?** 

Q7(b) [1 pt]	**Gegeven dit origineel, hoeveel verschillende kopies zijn er mogelijk?** 

Q7(c) [2 ptn]	**Gegeven dit origineel, hoeveel verschillende kopies zijn er mogelijk?** 

Q7(d) [2 ptn]	**Gegeven dit origineel, hoeveel verschillende kopies zijn er mogelijk?** 

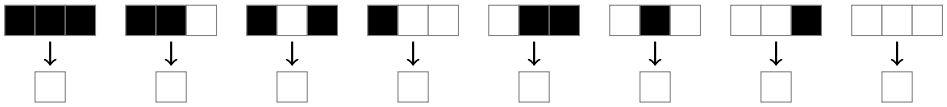
Vul hier je antwoorden in!

Q1(a) **Kleur de cellen zwart die leven in de 3e generatie.** /2

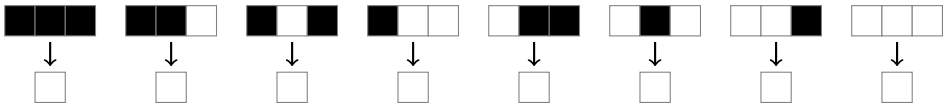
Q1(b) **Kleur de cellen zwart die leven in de 3e generatie met de nieuwe regel.** /2

Q1(c) **Kleur de cellen zwart die leven in de 12e generatie met de nieuwe regel.** /2

Q1(d) **Kleur de onderste vakjes zodat de juiste regel gevormd wordt.** /2



Q1(e) **Kleur de onderste vakjes zodat de juiste regel gevormd wordt.** /2



Q2(a) **Expressie (a) in algoritme 1X2** /2

. . . . . .

Q2(b) **Expressie (b) in algoritme 1X2** /2

. . . . . .

Q2(c) **Expressie (c) in algoritme Goals** /4

. . . . . .

Q2(d) **Het vervolg van expressie (d) in algoritme Overwinningen** /2

. . . . . .

Q2(e) **Expressie (e) in algoritme Overwinningen** /2

. . . . . .

Q3(a) /1

$(d[1] = d[5])$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q3(b) /1

$(d[1] = d[4])$ **or** $(d[2] = d[5])$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q3(c) /1

$(d[1] = d[2])$ **or** $(d[2] = d[3])$ **or** $(d[3] = d[4])$ **or** $(d[4] = d[5])$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q3(d) /1

$(d[1] = d[3])$ **or** $(d[2] = d[4])$ **or** $(d[3] = d[5])$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q3(e) /2

$\Big((d[1] = d[2])$ **and** $(d[3] = d[5])\Big)$ **or** $\Big((d[1] = d[3])$ **and** $(d[4] = d[5])\Big)$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q3(f) /2

$(d[1] \neq d[2])$ **and** $(d[2] \neq d[3])$ **and** $(d[3] \neq d[4])$ **and** $(d[4] \neq d[5])$ **and** $(d[5] - d[1] = 4)$ is altijd **true** voor ...

☐ *Yahtzee* ☐ *Carré* ☐ *Full House* ☐ *Trio* ☐ *Paar*
☐ *Dubbel Paar* ☐ *Grote straat* ☐ *Kleine straat* ☐ *Niets*

Q4(a) **Een expressie** /1

. . . . . .

Q4(b) **Een expressie** /1

. . . . . .

Q4(c) **Een expressie** /1

. . . . . .

Q4(d) **Een expressie** /1

. . . . . .

Q4(e) **Een expressie** /1

. . . . . .

Q4(f) **Een instructie** /1

. . . . . .

Q4(g)	**Een expressie**	/1
Q4(h)	**Een expressie**	/1
Q4(i)	**Een instructie**	/1
Q4(j)	**Een instructie**	/1
Q4(k)	**Een expressie**	/1
Q4(l)	**Een expressie**	/1
Q4(m)	**Een expressie**	/1
Q5(a)	**Een aantal routes**	/1
Q5(b)	**Een aantal routes**	/1
Q5(c)	**Een aantal routes**	/2
Q5(d)	**Een aantal routes**	/2
Q5(e)	**Een aantal routes**	/2
Q6(a)	**Een getal, de kortste afstand tussen de grijze nodes in de graaf met 7 nodes.**	/4
Q6(b)	**Een getal, de kortste afstand tussen de grijze nodes in de graaf met 10 nodes.**	/4
Q7(a)	**Een getal**	/1

Q7(b) **Een getal**	/1
Q7(c) **Een getal**	/2
Q7(d) **Een getal**	/2