be-OI 2016 **Finale - BELOFTEN** zaterdag 19 maart 2016	**Invullen in HOOFDLETTERS en LEESBAAR aub** VOORNAAM : ..... NAAM : ..... SCHOOL :	100 **Gereserveerd**

Belgische Informatica-olympiade (duur : 2u maximum)

Dit is de vragenlijst van de finale van de Belgische Informatica-olympiade 2016. Ze bevat 10 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.
11. Meerkeuzevragen werken als volgt : bij een goed antwoord krijg je de punten, bij een fout antwoord verlies je de punten. Geef je geen antwoord, dan krijg je geen punten. Je kan geen negatieve score halen. Om op een meerkeuzevraag te antwoorden, kruis je het hokje aan (☒) dat met het juiste antwoord overeenkomt. Om een antwoord te annuleren, maak je het hokje volledig zwart (■).
12. De toezichthouder of leerkracht zal geen vragen beantwoorden. Elke fout in de opgave moet je beschouwen als onderdeel van de proef.

Veel succes !

De Belgische Informatica-olympiade wordt mogelijk gemaakt door de steun van deze sponsors en leden :

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 20.

```
geboortejaar ← 1994
leeftijd ← 2014 − 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!
}
```

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
}
```

Vraag 1 – Opwarming

Je krijgt het volgende algoritme. Stel dat **n=2** :

```
count ← 0
for (i ← 0 to n − 1 step 1)
{
    for (j ← 0 to n − 1 step 1)
    {
        if (i < j) { count ← count − 1 }
        else { count ← count + 1 }
    }
}
```

Q1(a) [3 ptn]	**Wat is de waarde van de variable** $count$ **na het uitvoeren van dit algoritme ?**

Bekijk nu de volgende code :

```
i ← 0
while (i ≤ 3)
{
    i ← 2 ∗ i + 1
}
```

Q1(b) [3 ptn]	**Wat is de waarde van de variabele** i **na het uitvoeren van dit algoritme ?**

De code $x\%2 = 0$ betekent dat x even is.

Q1(c) [3 ptn]	**Welke code betekent dan** ***"*** a ***en*** b ***zijn oneven"*** **?**
☐	`a%2 = 0 or b%2 = 0`
☐	`a%2 = 0 and b%2 = 0`
☐	`not (a%2 = 0 or b%2 = 0)`
☐	`not (a%2 = 0 and b%2 = 0)`

Vraag 2 – De verborgen berekening ...

Bij het opruimen van de zolder bots je op een oud wiskundeboek. Daarin vind je een bladzijde met eem mysterieus algoritme dat je hieronder ziet.

```
Input : a en b, twee strikt positieve natuurlijke getallen
Output : ?

while ( a ≠ b )
{
  if ( a > b )
  {
    a ← a − b
  }
  else
  {
    b ← b − a
  }
}
return a
```

Dit lijkt iets interessants te doen, maar de pagina met de uitleg is al opgegeten door de muizen.

Q2(a) [10 ptn]	**Wat berekent dit algoritme ?**

(Hint : probeer dit eens verschillende keren uit met andere getallen, bijvoorbeeld 9 en 6, of 10 en 4 !)

Vraag 3 – Haakjes

Bart Simpson moet als straf 100 wiskundesommen op de computer typen, en wil er snel vanaf zijn. Zijn lerares, mevr. Krabappel, houdt hem echter scherp in de gaten. Ze heeft alvast een algoritme bedacht om te controleren of hij de haakjes wel juist gebruikt.

Een rekensom wordt getypt als een reeks karakters. Bij juiste haakjes is er voor elk open haakje "(" ook een gesloten haakje ")" verderop. Bovendien moet je voor elk gesloten haakje ")" dat je tegenkomt, eerder al een open haakje "(" zijn tegengekomen.

Bijvoorbeeld, de reeks karakters "$(a+2+(c/4)-(1+e))$" gebruikt de haakjes juist. De reeks karakters "$)a+(2-x)/(1$" gebruikt haakjes fout, en ook "$a+(2-3)/((3+a)+3$" heeft foute haakjes.

Het algoritme van mevr. Krabappel staat hieronder, maar je moet het eerst nog aanvullen !

```
Input  : s, een array van karakters.
         n, een natuurlijk getal, is de lengte van array s.
Output : true als de rekensom voorgesteld in array s op een juiste manier
         haakjes gebruikt, en anders false.

num ← 0
for (i ← 0 to [...] step 1)                                          // (a)
{
    if ( s[i] = '(' )
    {
        [...]                                                        // (b)
    }
    if ( s[i] = ')' )
    {
        [...]                                                        // (c)
    }
    if ( [...] )                                                     // (d)
    {
        return false
    }
}
return (num = 0) // dit geeft true terug als num gelijk is aan nul, anders false
```

Q3(a) [2 ptn]	**Vervolledig (a)**

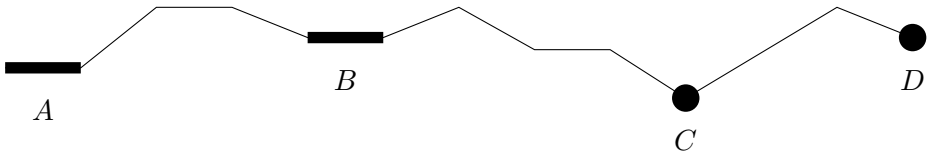
Q3(b) [2 ptn]	**Vervolledig (b)**

Q3(c) [2 ptn]	**Vervolledig (c)**

Q3(d) [2 ptn]	**Vervolledig (d)**

Vraag 4 – Bergen en Dalen

Je vertrekt op wetenschappelijke expeditie naar het Oeralgebergte. Je wil je tent graag op de bodem van een vallei opzetten, want daar is zeker stromend water. De figuur hieronder geeft weer hoe jouw wandelroute stijgt en daalt :



Een *bodem van een vallei* is iedere plaats die je alleen maar kan verlaten door omhoog te gaan, in iedere richting. Dat kunnen zowel punten als lijnstukken zijn. Op de tekening hierboven zijn er 4 *bodems*, aangeduid met A, B, C en D. (Het platte stukje tussen B en C is geen bodem want je kan verder dalen tot C.) Een volledig plat wandelpad heeft 1 bodem.

Je kan de hoogtes langs je wandelroute uit je GPS halen als een array van gehele getallen. Dan krijg je bijvoorbeeld een array zoals `[5,4,3,3,3,9,9,8]`. Die heeft 2 bodems : het platte stuk op hoogte 3, en het laatste punt op hoogte 8. De array `[4]` heeft 1 bodem. Ook de array `[3,1,8]` heeft er 1. De array `[5,5,8,5,5]` heeft er 2.

Q4(a) [2 ptn]	**Hoeveel bodems heeft de array** `[2,5]` **?**

Q4(b) [2 ptn]	**Hoeveel bodems heeft de array** `[1,1,1,5,5,9,9,9,8,6,12,17,17,4,4,1,-12]` **?**

Je wil nu snel een overzicht krijgen van alle kampeermogelijkheden. Dit algoritme telt alle *bodems* in gegeven array.

```
Input  : arr, een array van gehele getallen.
         n, een strikt positief natuurlijk getal, is de lengte van array arr.
Output : Het aantal "bodems" in arr wordt teruggegeven.

aantal ← 0
gedaald ← true
for (i ← 1 to n − 1 step 1)
{
    if ( gedaald and ( [...] ) )                          // (a)
    {
        aantal ← aantal + 1
        gedaald ← false
    }
    gedaald ← gedaald or ( [...] )                        // (b)
}
if ( gedaald )                          // (dit stukje verwerkt het laatste element nog)
{
    aantal ← aantal + 1
}
return aantal
```

Q4(c) [3 ptn]	**Wat ontbreekt er bij** `(a)` **?**

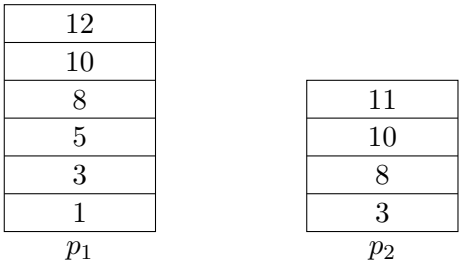
Q4(d) [3 ptn]	**Wat ontbreekt er bij** `(b)` **?**

Vraag 5 – De Ruilkaarten...

Je bent een verwoed speler van het ruilkaartenspel *Hassle : The Dorkening*. Je moet zoveel mogelijk unieke kaarten bezitten om veel spelletjes te kunnen winnen. Alle kaarten hebben een nummer. Jouw verzameling kaarten is gesorteerd van klein naar groot en op een stapel p_1 gelegd. De kaart met het grootste nummer ligt bovenaan. Je hebt geen kaarten dubbel.

Daarnet heb je de volledige verzameling van iemand anders overgekocht. Die is op dezelfde manier gesorteerd en op een andere stapel p_2 gelegd. In p_2 zitten ook geen kaarten dubbel, maar er is misschien wel overlap met jouw eigen verzameling !

Natuurlijk wil je alleen de kaarten houden die je nog niet had. Kaarten die je nu plots dubbel hebt, wil je weer doorverkopen (met een mooie winst).



Je kan van de stapel alleen de bovenste kaart zien. En natuurlijk kan je geen kaart weghalen uit het midden van de stapel zonder eerst de kaarten weg te halen die daarbovenop liggen.

Samengevat, kan je dus de volgende 4 bewerkingen doen :

— `top(`p`)` : Lees het nummer van de bovenste kaart op stapel p ;

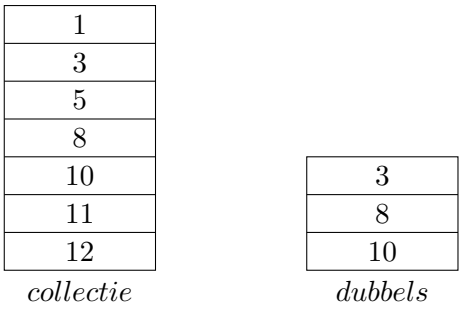
— `pop(`p`)` : Haal de bovenste kaart van stapel p af ;

— `push(`p`,`e`)` : Voeg kaart nummer e toe aan stapel p ;

— `isEmpty(`p`)` : Kijk of stapel p leeg is of niet.

Je wil nu alle kaarten sorteren in twee nieuwe stapels : *collectie* en *dubbels*. De stapel *collectie* moet een zo volledig mogelijke verzameling van kaarten zijn. Er mag slechts één exemplaar van elk kaart in zitten. De stapel *dubbels* bevat dan enkel de kaarten die je dubbel hebt.

Die twee nieuwe stapels moeten ook zo gesorteerd zijn dat het grootste nummer nu onderaan de stapel zit, en het kleinste nummer bovenaan. Als we dat doen met de tekening hierboven, geeft dat het volgende resultaat :



Het algoritme hieronder voert die hele operatie uit. Vervolledig het ! (Ter verduidelijking : de functie **not** (x) berekent het tegengestelde van x.)

```
Input  : p1, p2 : twee stapels, stijgend gesorteerd, met het hoogste element boven,
          en waarop je de eerder gedefinieerde bewerkingen kan uitvoeren.
Output : De stapel collectie bevat de meest volledige verzameling, andersom gesorteerd,
         De stapel dubbels bevat alle dubbels, omgekeerd gesorteerd.

collectie ← een nieuwe lege stapel
dubbels ← een nieuwe lege stapel

while ([...])                                    // (a)
{
    if ([...])                                   // (b)
    {
        m ← pop(p1)
        push(collectie, m)
    }
    else if (top(p1) = top(p2))
    {
        m ← pop(p1)
        push([...], m)                           // (c)
    }
    else
    {
        m ← pop(p2)
        push(collectie, m)
    }
}

while ( not(isEmpty(p1)) )
{
    m ← pop(p1)
    [...]                                        // (d)
}

while ( not(isEmpty(p2)) )
{
    m ← pop(p2)
    [...]                                        // (d)
}
```

Q5(a) [2 ptn]	**Welke voorwaarde zoeken we bij** (a) **?**

Q5(b) [2 ptn]	**Welke voorwaarde zoeken we bij** (b) **?**

Q5(c) [2 ptn]	**Welke stapel moet je invullen bij** (c) **?**

Q5(d) [2 ptn]	**Welke instructie zoeken we bij** (d) **(tweemaal herhaald in dit programma) ?**

Vraag 6 – De dolle informaticus

De stoppen zijn doorgeslagen bij de arme informaticus die de volgende code schreef. Je mag aannemen dat de variabele v een geheel getal bevat :

```
Input : een variabele v

if ( (v > 1 and v ≤ 2) or (v ≥ 4 and v ≤ 6) )
{
   if ( (v > 2 and v < 4) or (v > 5 and v ≤ 9) )
   {
      Print('Luke, I am your father!')
   }
   else
   {
       if (v ≥ 3 and v ≤ 4)                    // (a)
       {
           Print('May the force be with you!')
       }
   }
}
else
{
   Print('Hhhrrrrrrrrrrraaaaaaaaaaaaeeeerrrrrrr')
}
```

Gegeven de volgende waarden van v. Zal dit programma dan *Hhhrrrrrrrrrrraaaaaaaaaaaaeeeerrrrrrr* printen ? (Als je dit niet had herkend, dit is Chewbacca die zingt onder de douche).

Q6(a) [1 pt]	Ja / Nee	$v = 1$
Q6(b) [1 pt]	Ja / Nee	$v = 2$
Q6(c) [1 pt]	Ja / Nee	$v = 3$
Q6(d) [1 pt]	Ja / Nee	$v = 4$

Q6(e) [2 ptn]	**Wat zal het programma printen voor $v = 2$?**

Q6(f) [2 ptn]	**Wat zal het programma printen voor $v = 4$?**

Vraag 7 – Wie zoekt die vindt

Je krijgt een array van gehele getallen, gesorteerd van klein naar groot. Je krijgt ook een geheel getal N. Je moet een algoritme schrijven dat N opzoekt in de array en de plaats ervan teruggeeft. Je mag ervan uitgaan dat het getal zich in de array bevindt.

Als getalenteerd informaticus ken je natuurlijk de beste manier om dat te doen. Die staat hieronder, vul ze aan !

```
Input : arr, een array van gehele getallen, gesorteerd van klein naar groot
        len, een strict positief geheel getal, de lengte van arr
        N, een geheel getal dat voorkomt in arr.

begin ← 0
eind ← len − 1

while(begin ≤ eind)
{
    mid ← [...]                    // (a)
    if(arr[mid] < N)
    {
        [...]                      // (b)
    }
    if(arr[mid] > N)
    {
        [...]                      // (c)
    }
    if(arr[mid] = N)
    {
        return mid
    }
}
```

Q7(a) [3 ptn]	**Wat ontbreekt er bij** `(a)` **?**

Q7(b) [3 ptn]	**Wat ontbreekt er bij** `(b)` **?**

Q7(c) [3 ptn]	**Wat ontbreekt er bij** `(c)` **?**

Vraag 8 – Crazy Arrays

Als je een array V van getallen doorloopt, dan begin je meestal bij het eerste element $V[0]$, daarna $V[1]$, $V[2]$, enzovoort tot het laatste element. We maken hierop een variant, die we *Crazy Array* noemen. De volgorde waarin we de elementen bezoeken wordt bepaald door een tweede array $volg$ en een variabele t.

Let nu goed op : het eerste element dat we bezoeken is het element op index t. (Ter herinnering, als $t = 6$ dan is dat dus het 7e element, want we nummeren de elementen vanaf 0.) Het tweede element dat we bezoeken is het element op index $volg[t]$. Het derde element dat we bezoeken is het element op index $volg[volg[t]]$, enzovoort. Als een element op index i het laatste is van de array, dan is $volg[i] = -1$.

Hier zijn drie voorbeelden :

Voorbeeld 1

$t = 3$

$V =$	2	4	5	1	3
$volg =$	4	2	-1	0	1

Voorbeeld 2

$t = 4$

$V =$	1	2	3	4	5
$volg =$	-1	2	3	0	1

Voorbeeld 3

$t = 5$

$V =$	89	19	34	57	71	16	43	33	30	88	66	82	93	85	17	34	43	29	37	78
$volg =$	12	17	18	10	19	14	3	15	7	0	4	13	-1	9	1	2	6	8	16	11

Als je V doorloopt op de *Crazy Array* manier, welke van deze voorbeelden geven je dan een reeks getallen die gesorteerd is van klein naar groot ?

Q8(a) [2 ptn]	Gesorteerd / Niet gesorteerd	Voorbeeld 1
Q8(b) [2 ptn]	Gesorteerd / Niet gesorteerd	Voorbeeld 2
Q8(c) [2 ptn]	Gesorteerd / Niet gesorteerd	Voorbeeld 3

We willen deze Crazy Arrays nu sorteren. Dat kan zonder V te veranderen : het is genoeg om de volgorde die bepaald wordt door $volg$ en t aan te passen. Dit is een Crazy Array die niet gesorteerd is :

$t = 0$

$V =$	1	3	4	7	2	6	5
$volg =$	4	2	6	5	1	-1	3

Je moet nu zo weinig mogelijk elementen van $volg$ veranderen zodat je de array hierboven kan doorlopen van klein naar groot.

Q8(d) [3 ptn]	**Welke elementen moet je veranderen, en welke waarde moet je ze geven ? Geef als antwoord één of meer toekenningen van de vorm $volg[i] = j$.**

Verderop vind je een algoritme om een Crazy Array te sorteren. Het werkt als volgt : er is een variabele $sorted$. Als je V volgt beginnend vanaf $sorted$ (op de *Crazy Array*-manier : eerst komt positie $sorted$, dan positie $volg[sorted]$, dan $volg[volg[sorted]]$, etc, totdat je -1 tegenkomt), dan krijg je een reeks getallen die van klein naar groot gesorteerd is. De variabele $sorted$ geeft je dus toegang tot het deel van V dat al gesorteerd is. Vanaf variabele t loop je dan het deel van V af dat nog niet gesorteerd is.

In het begin is $sorted = -1$, omdat er nog niets gesorteerd is. Bij elke iteratie van buitenste **`while`**-lus, halen we een element uit het niet-gesorteerde deel (toegankelijk via t) en zetten we het op de juiste plaats in het gesorteerde deel.

```
Input : V, een array van gehele getallen
        volg, een array van gehele getallen
        N, een strikt positief getal, gelijk aan de lengte van V en volg
        t, een geheel getal tussen 0 en N − 1 (inclusief)

sorted ← −1

while (t ≠ -1)                              // Buitenste while-lus
{
   j ← sorted
   while ( (j ≠ -1) and (V[t] > V[j]) )
   {
      jprev ← [...]                         // (1)
      j ← volg[j]                           // Lijn ⋆
   }

   if ([...])                               // (2)
   {
       sorted ← t
   }
   else
   {
       volg[jprev] ← t
   }
   tmp ← volg[t]
   volg[t] ← j
   [...] ← tmp                               // (3)
}
t ← sorted
```

Q8(e) [2 ptn]	**Vervolledig expressie (1)**

Q8(f) [2 ptn]	**Geef de ontbrekende voorwaarde (2)**

Q8(g) [2 ptn]	**Welke variabele ontbreekt bij (3) ?**

Vraag 9 – Doe het licht uit !

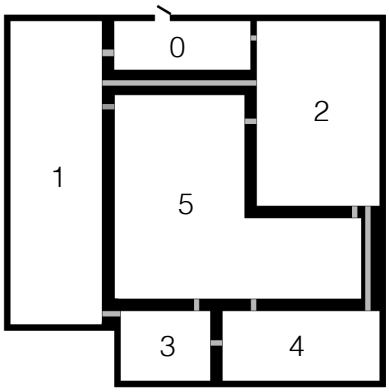
Auteur : *Jorik Jooken, Gilles Geeraerts*

Pieter en zijn vrienden gaan op vakantie. Voordat ze vertrekken, ziet Pieter dat ze vergeten zijn alle lichten in huis uit te doen. In het huis is er een lamp in elke gang, die bediend wordt door een schakelaar in het midden van die gang. Eens Pieter het licht in een gang heeft uitgedaan, kan hij er geen tweede keer doorheen omdat het te donker is.

Pieter moet dus een manier vinden om alle kamers en alle gangen in het huis te doorlopen, *en mag daarbij elke gang maar 1 keer doorlopen.* Hij kan de kamers wel meerdere keren bezoeken, maar kan niet omkeren in het midden van een gang.

Pieter begint zijn ronde van het huis in kamer 0 (waar de deur is) en moet eindigen in diezelfde kamer, zodat hij het huis uit kan. Hieronder staat een kaart van het huis waarbij de muren zwart zijn, en de gangen grijs. De vraag is of het mogelijk is op die manier het hele huis rond te gaan.

Hier is een kaart van het huis :



Q9(a) [5 ptn]	**Geef, indien mogelijk, een route die gaat van kamer 0 naar kamer 0, waarbij elke gang exact 1 keer doorlopen wordt. Schrijf « onmogelijk » als je denkt dat zo'n route niet bestaat.**

Vraag 10 – The Usual Suspects

Auteur : *Niels Joncheere*

Er wordt een misdaad gepleegd in de stad, maar de politie weet niet wie de daders zijn. Gelukkig zijn er in de stad maar 5 misdadigers : Alex, Bernard, Camille, David, en Eva. De politie arresteert ze alle 5 als verdachten. Ze worden ondervraagd, en ze zeggen :

1. Alex zegt : “De misdaad werd gepleegd door Bernard, met hulp van Camille en/of Eva.”
2. Bernard zegt : “Als Camille bij de misdaad betrokken was, dan heeft Alex meegeholpen.”
3. Camille zegt : “Ofwel heeft Bernard niets met de misdaad te maken, ofwel heeft Eva er niets mee te maken.”
4. David zegt : “Als Camille niet bij de misdaad betrokken was, dan heeft Alex meegeholpen.”

Als er gezegd wordt dat iemand bij de misdaad betrokken *zou kunnen zijn*, en niemand anders spreekt dat tegen, dan mag de politie die persoon een nachtje opsluiten. Om uit te zoeken wie ze mogen opsluiten, bekijkt de politie alle verklaringen apart.

Q10(a) [2 ptn]	**Hoeveel personen kunnen een dag lang worden opgesloten als de politie alleen let op verklaring ?? ?**

Q10(b) [2 ptn]	**Welke verdachten zijn er *zeker* bij de misdaad betrokken als de politie alleen let op verklaring ?? ?**

Q10(c) [3 ptn]	**Hoeveel personen kunnen een dag lang worden opgesloten als de politie alleen let op verklaringen ?? en ?? ?**

Vul hier uw antwoorden in !

Q1(a) **een getal**	/3
Q1(b) **een getal**	/3

Q1(c)		**Er is maar EEN juist antwoord!**	/3
	☐	`a%2 = 0 or b%2 = 0`	
	☐	`a%2 = 0 and b%2 = 0`	
	☐	`not (a%2 = 0 or b%2 = 0)`	
	☐	`not (a%2 = 0 and b%2 = 0)`	

Q2(a) **Leg uit wat het algoritme berekent**	/10
Q3(a) **Een expressie**	/2
Q3(b) **Een instructie**	/2
Q3(c) **Een instructie**	/2
Q3(d) **Een expressie**	/2
Q4(a) **Een getal**	/2
Q4(b) **Een getal**	/2
Q4(c) **Een voorwaarde**	/3
Q4(d) **Een voorwaarde**	/3
Q5(a) **Een voorwaarde**	/2

Q5(b) **Een voorwaarde** /2

. . . . . .

Q5(c) **De naam van een stapel** /2

. . . . . .

Q5(d) **Een instructie** /2

. . . . . .

	Ja	Nee		/4
Q6(a) /1	☐	☐	$v = 1$	
Q6(b) /1	☐	☐	$v = 2$	
Q6(c) /1	☐	☐	$v = 3$	
Q6(d) /1	☐	☐	$v = 4$	

Q6(e) **Een zin geprint door het programma, of "Niets".** /2

. . . . . .

Q6(f) **Een zin geprint door het programma, of "Niets".** /2

. . . . . .

Q7(a) **Een expressie** /3

. . . . . .

Q7(b) **Een instructie** /3

. . . . . .

Q7(c) **Een instructie** /3

. . . . . .

	Gesorteerd	Niet gesorteerd		/6
Q8(a) /2	☐	☐	Voorbeeld 1	
Q8(b) /2	☐	☐	Voorbeeld 2	
Q8(c) /2	☐	☐	Voorbeeld 3	

Q8(d) **Toekenningen aan array** $volg$ /3

Q8(e) **Een expressie**	/2
Q8(f) **Een voorwaarde**	/2
Q8(g) **Een variabele**	/2
Q9(a) **Een route, of « onmogelijk ».**	/5
Q10(a) **Een getal**	/2
Q10(b) **Eén of meerdere namen**	/2
Q10(c) **Een getal**	/3