be-OI 2014 **Halve Finale** woensdag 27 november 2013	**Invullen in HOOFDLETTERS en LEESBAAR aub** VOORNAAM : ..... NAAM : ..... SCHOOL :	O **Gereserveerd**

Belgische Informatica-olympiade (duur : 3u maximum)

Dit is de vragenlijst van de halve finale van de Belgische Informatica-olympiade 2014. Ze bevat acht vragen, en je krijgt **maximum 3u** de tijd om ze op te lossen. Naast elke vraag vind je een schatting van de benodigde tijd om ze op te lossen, en de maximale score die je kan behalen.

Algemene opmerkingen (lees dit aandachtig voor je begint)

1. Vul je voornaam, naam en school in, **alleen op het eerste blad**. Jouw antwoorden moet je invullen op de daarop voorziene antwoordbladen, die zich achteraan in deze bundel papier bevinden. Als je door een fout toch buiten de antwoordkaders moet schrijven, schrijf dan alleen verder op hetzelfde blad papier. Anders kunnen we je antwoord niet verbeteren.
2. Wanneer je gedaan hebt, geef je aan de toezichthoud(st)er(s) deze eerste bladzijde terug (met jouw naam erop), en de pagina's met jouw antwoorden. De andere pagina's mag je bijhouden.
3. Je mag alleen schrijfgerief bij je hebben. Rekentoestel, GSM, ... zijn **verboden**.
4. Schrijf je antwoorden met blauwe of zwarte **pen of balpen**. Laat geen antwoorden staan in potlood. Als je kladbladen wil, vraag ze dan aan een toezichthouder.
5. Voor alle opgaven werd **pseudo-code** gebruikt. Op de volgende bladzijde vind je een **beschrijving** van de pseudo-code die we hier gebruiken.

   Op open vragen mag je zelf antwoorden in **pseudo-code** of in één van de **toegestane programmeertalen** (Java, C, C++, Pascal, Python, PHP). We trekken geen punten af voor syntaxfouten. Tenzij het anders vermeld staat, mag je geen voorgedefinieerde functies gebruiken, met uitzondering van `max(a,b)`, `min(a,b)` en `pow(a,b)`, waarbij deze laatste a^b berekent.
6. Voor elke meerkeuzevraag (waarbij hokjes moeten worden aangekruist) doet een fout antwoord je evenveel punten *verliezen* als je met een correct antwoord zou winnen. Als je de vraag niet beantwoordt dan win je noch verlies je punten. Behaal je op die manier een negatieve score voor een volledige vraag, dan wordt de score teruggezet naar nul. 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 (■).
7. Je mag **op geen enkel moment met iemand communiceren**, behalve met de toezichthouders. Je mag hen bijvoorbeeld wel om kladpapier vragen, maar je mag geen vragen stellen over de inhoud van de proef. Zo garanderen wij gelijke behandeling tussen deelnemers in alle regionale centra. Elke fout in de opgave moet je beschouwen als onderdeel van de proef.
8. Je mag in principe je **plaats niet verlaten** tijdens de proef. Moet je dringend naar het toilet, meldt dit dan aan een toezichthouder. Hij of zij kan dit toelaten of weigeren, afhankelijk van de mogelijkheid om je te laten vergezellen door een van de toezichters. Afhankelijk van het regionaal centrum kan het ook verboden zijn om te eten of te drinken in het lokaal.

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 ← 1993
leeftijd ← 2013 − 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    // Verlaagde prijs
}
```

Wanneer we in één variabele tegelijkertijd 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.

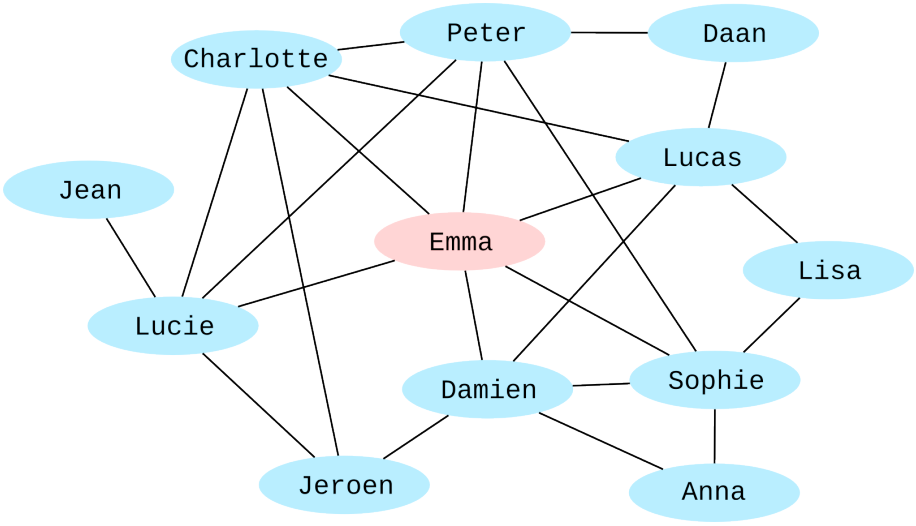
```
sum ← 0
for (i ← 0 to N − 1 step 1)
{
  sum ← sum + arr[i]
}
```

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 $4 \ldots$ 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 – Sociale Netwerken (5 min – 6 ptn)

Emma is lid van een sociaal netwerk waar zij *bevriend* kan zijn met andere personen die ook lid zijn van dat netwerk. Het diagram hieronder toont de *vrienden* van Emma, en ook de *vrienden van vrienden* van Emma.



Een lijn tussen twee personen wil zeggen dat ze *bevriend* zijn op het sociale netwerk. Bijvoorbeeld, Sophie is vriend van Lisa en Lisa is vriend van Lucas, maar Sophie is geen vriend van Lucas.

Elke persoon kan zijn/haar foto's alleen delen met zijn/haar vrienden. Zodra iemand een *vriend* toegang geeft tot een foto, kunnen die vrienden commentaar geven op de foto. Zodra iemand commentaar geeft op een foto, kunnen *diens vrienden* de foto en die commentaar zien, maar ze kunnen zelf geen nieuwe commentaar geven, tenzij ze al in het begin toelating hadden om commentaar te geven.

Emma heeft een foto van haar huisdier gepost (een chihuahua genaamd Paris), maar ze wil niet dat Damien die foto te zien krijgt. Ze is er zeker van dat Damien niet plots bevriend zal worden met een van haar eigen vrienden (dus: het diagram hierboven zal niet veranderen).

Beschouw de volgende personen. Heeft Emma het recht om ze toegang te geven tot de foto, als ze ook de garantie wil behouden dat Damien de foto nooit te zien zal krijgen?

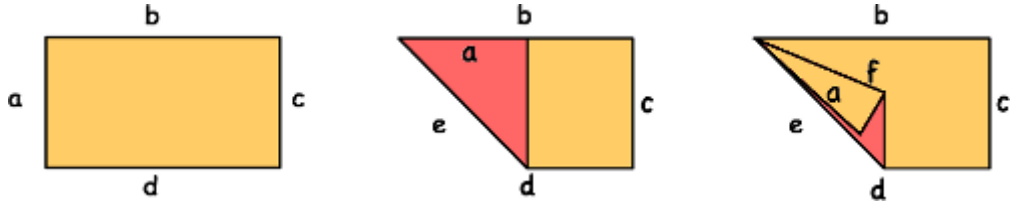
		Persoon
Q1(a) [1 pt]	JA / NEE	Charlotte
Q1(b) [1 pt]	JA / NEE	Lucas
Q1(c) [1 pt]	JA / NEE	Damien
Q1(d) [1 pt]	JA / NEE	Lucie
Q1(e) [1 pt]	JA / NEE	Daan
Q1(f) [1 pt]	JA / NEE	Peter

Vraag 2 – Origami (5 min – 6 ptn)

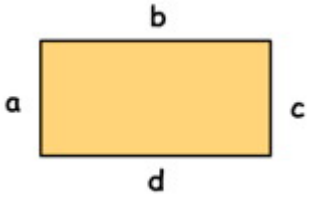
Je klasgenote Sakura is heel begaafd in het vouwen van papieren bloemen. Je vindt dat best amusant en interessant, en dus wil je het zelf ook leren. Maar omdat de Japanse Sakura nog niet zo lang in België woont bots je tegen een grote taalbarrière. Om snel van start te kunnen gaan verzin je een eenvoudige codetaal, waarmee je vlot haar voorbeeld-instructies kunt noteren en instuderen.

Deze codetaal bestaat uit een instructie $fold$ die je toelaat om één zijde van een stuk papier naar een andere zijde te vouwen. De instructie $z = fold(x, y)$ wil zeggen dat we de zijde x vouwen naar de zijde y toe. De nieuwe zijde die daardoor ontstaat noemen we z.

Hieronder staat bijvoorbeeld afgebeeld wat er gebeurt als we achtereenvolgens de instructies $e = fold(a, b)$ en $f = fold(a, e)$ uitvoeren.



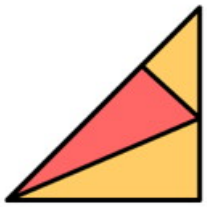
Stel dat we het volgende stuk papier nemen, waarbij de zijde b tweemaal langer is dan de zijde a:



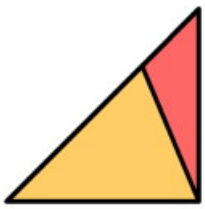
Als we daarop de volgende instructies uitvoeren: $e = fold(c, a)$, $f = fold(c, d)$ en $g = fold(a, f)$

Q2(a) [6 ptn]	**Welke van de vier figuren hieronder komt overeen met het eindresultaat van deze reeks instructies?**

1.



2.



3.

4.

Vraag 3 – Leve de generaal! Een man van staal! (15 min – 12 ptn)

De revolutie slaat toe op San Theodoros. De beruchte generaal Tapioca heeft de goede generaal Alcazar omvergeworpen. Alcazar kon zich maar op het nippertje terugtrekken in de jungle, dankzij zijn trouwe Picaro's.

Om zijn overwinning meer glans te geven, heeft generaal Tapioca beslist om alle referenties naar zijn voorganger te laten verwijderen uit alle wetteksten en andere archieven van San Theodoros. Hij wil bovendien dat iedere vermelding van de woorden « generaal Alcazar » wordt vervangen door « grote en geniale generaal Tapioca »!

De archieven van San Theodoros zijn gelukkig geïnformatiseerd. Iedere tekst is opgeslagen in een array, karakter per karakter (de eerste letter in het eerste element van de array, de tweede in het tweede element, etc). Bijvoorbeeld:

G	e	n	e	r	a	a	l		A	l	c	a	z	a	r		v	e	r	k	l	a	a	r	t		…

Voordat ze beginnen met het schrijven van een programma dat de nodige veranderingen aanbrengt, stellen de informatici van San Theodoros zich eerst de vraag: hoeveel opslagruimte zullen ze gebruiken eens die wijzigingen zijn doorgevoerd? Stel dat het originele document N karakters bevat; dat de reeks karakters die vervangen moet worden (bijvoorbeeld « generaal Alcazar ») lengte L_{oud} heeft; dat de nieuwe reeks waardoor die vervangen moet worden als lengte L_{nieuw} heeft; en dat de reeks die vervangen moet worden k keer in het originele document voorkomt.

Q3(a) [3 ptn]	**Wat is, in het algemeen, de nieuwe lengte (het aantal karakters) van het document na de vervanging? Geef een expressie in functie van N, k, L_{nieuw} en L_{oud}.**

Om de nodige veranderingen door te voeren, hebben de informatici van San Theodoros het onderstaande algoritme geschreven. Zoals je kan lezen in de commentaren, zal deze code de originele wettekst (de array $tekst$) karakter per karakter kopiëren naar de array $resultaat$. Ondertussen wordt elk karakter vergeleken met de array oud. Telkens wanneer de array oud volledig werd gedetecteerd in $tekst$, wordt deze vervangen door $nieuw$ in de array $resultaat$. Dit algoritme gaat ervan uit dat $resultaat$ groot genoeg is om het resultaat van alle vervangingen te bevatten.

Een cyberaanval van de Picaro's heeft deze plannen echter in de war gestuurd, en enkele expressies zijn uitgewist. Aan jou om die expressies in te vullen:

Q3(b) [3 ptn]	**Wat is expressie** `(b)` **?**

Q3(c) [2 ptn]	**Wat is expressie** `(c)` **?**

Q3(d) [2 ptn]	**Wat is expressie** `(d)` **?**

Q3(e) [2 ptn]	**Wat is expressie** `(e)` **?**

```
Input : een reeks karakters tekst , met lengte N
        een reeks karakters oud , met lengte L_oud
        een reeks karakters nieuw , met lengte L_nieuw
Output : een reeks karakters resultaat, bekomen door elk voorkomen van de reeks
        oud te vervangen door de reeks nieuw in de reeks tekst


it ← 0      // Om tekst te doorlopen
id ← 0      // Om resultaat te doorlopen
j ← 0       // Om oud te doorlopen

while (it < N)
{
  resultaat[id] ← tekst[it]

  // Hebben we een volgend karakter van oud gevonden in tekst ?

  if (...)          // (b)
  {
    j ← j + 1
  }
  else
  {
    j ← 0
    if (resultaat[id] = oud[0])
    {
      j ← 1
    }
  }

  it ← it + 1
  id ← id + 1

  // We gaan na of het nodig is om de L_oud laatste karakters die
  // werden toegevoegd aan resultaat te vervangen door de reeks nieuw.

  if (...)            // (c)
  {
    for (i ← 0 to L_nieuw − 1 step 1)
    {
      resultaat[...] ← nieuw[i] // (d)
    }
    id ← ...          // (e)
    j ← 0
  }
}
```

Vraag 4 – Min-Max (30 min – 20 ptn)

Twee vaak gebruikte functies bij het programmeren zijn de min- en max-functie. Beide functies krijgen twee argumenten mee. Als a en b twee getallen zijn, dan geeft $min(a, b)$ het kleinste getal terug en $max(a, b)$ het grootste getal terug.

Je kan de min- en max-functies combineren met elkaar en met andere operatoren ($+$,$-$,$\times$,$/$) om lange *expressies* te maken. min en max zijn daarbij vooral interessant omdat zij gebruikt kunnen worden om een selectie maken: ze selecteren het kleinste of grootste getal, om dan daarmee verder te gaan. Op die manier kan je soms een `if`-instructie vervangen.

Hieronder zie je een voorbeeld van een expressie waarin min, max, en de operatoren $+$ en $-$ voorkomen (deze doet trouwens niets specifiek nuttigs):

```
max(a-c, min(b+a, c+a))
```

We vragen je nu om het omgekeerde te doen: we willen een bepaalde bewerking uitvoeren op 1 of meer getallen, en jij moet één expressie samenstellen die deze bewerking volledig kan uitvoeren. Jouw expressies mogen alleen de basisoperatoren ($+$,$-$,$\times$,$/$) en de functies min en max gebruiken.

a) Grootste van drie getallen

```
Input : drie getallen a, b en c
Output : het grootste van de getallen a, b en c

return [...] // (a)
```

Q4(a) [3 ptn]	**Geef een zo kort mogelijke, equivalente expressie.**

b) Absolute waarde

De absolute waarde van een getal, is het getal zonder zijn teken (de absolute waarde van 3 is 3, de absolute waarde van -5 is 5).

```
Input : een getal a
Output : de absolute waarde van a

return [...] // (b)
```

Q4(b) [3 ptn]	**Geef een zo kort mogelijke, equivalente expressie.**

c) Dichtstbijzijnde getal

```
Input : drie getallen a, b en c
Output : het absolute verschil (de afstand) tussen a en het getal dat
          het dichtst bij a ligt van de getallen b en c

return [...] // (c)
```

Q4(c) [5 ptn]	**Geef een zo kort mogelijke, equivalente expressie.**

d) Alles of niets

Deze opgave kan je voorstellen als volgt: je wil een voorwerp verplaatsen met gewicht b, maar je kunt maximaal a dragen. Het is niet mogelijk om een gedeelte van het voorwerp mee te nemen (je moet dus alles of niets meenemen). Hoeveel gewicht zal je moeten torsen?

We zijn hier op zoek naar het antwoord dat correct is voor alle *reële* getallen a en b (m.a.w., a en b kunnen kommagetallen zijn)

```
Input : twee positieve, reële, getallen a ≠ b
Output : 0 als a < b, anders b

return [...] // (d)
```

Q4(d) [5 ptn]	**Geef een zo kort mogelijke, equivalente expressie.**

e) Alles of niets, deel 2

Als we a en b beperken tot *gehele* getallen, bestaat er een nog kortere expressie. Let op voor de *gehele deling*, die werkt zoals beschreven staat op de overzichtspagina over pseudocode aan het begin van deze proef.

Kan je de kortste expressie vinden die geldt voor gehele getallen?

```
Input : twee positieve, gehele, getallen a ≠ b
Output : 0 als a < b, anders b

return [...] // (e)
```

Q4(e) [4 ptn]	**Geef een zo kort mogelijke, equivalente expressie.**

Vraag 5 Knipperlichtjes (15 min – 14 ptn)

Om het Chinese nieuwjaar gepast te vieren, hebben de Taiwanese autoriteiten beslist om knipperende lichtsnoeren te installeren in de stad. Deze verlichting is *programmeerbaar*: je kan de verschillende lampen aan- en uitzetten volgens een gegeven programma. Om de mogelijkheden van de verlichting te testen, wordt er je gevraagd om een simulator te schrijven. Die houdt een array G bij van lengte N (het aantal lampen), zodat $G[i] = 1$ als de i^e lamp brandt, en $G[i] = 0$ als de i^e lamp gedoofd is.

Het programma van het lichtsnoer is opgeslagen in een array Act. Elk element van de array Act kan één van de volgende instructies bevatten:

1. SL: voert een *circulaire rotatie naar links* uit. Na uitvoering van deze instructie, is de inhoud van elk element i (voor $1 \leq i \leq N-1$) gekopieerd naar element $i-1$ (het element links ervan). De inhoud van element 0 is gekopieerd naar element $N-1$.
2. SR: het tegenovergestelde van operatie SL: een *circulaire rotatie naar rechts*.
3. ON: zet alle lampen aan.
4. OFF: zet alle lampen uit.
5. INV: inverteer alle lampen (zet de lampen die aan staan uit, en de lampen die uit staan aan).

Bijvoorbeeld, als we de volgende arrays G en Act hebben:

$G =$ | 0 | 0 | 1 | 0 | 1 | $Act =$ | SL | SR | SL | INV |

Dan zal de array G bij de uitvoering van elke instructie in Act, achtereenvolgens de volgende inhoud hebben:

Na het uitvoeren van $Act[0]$: $G =$ | 0 | 1 | 0 | 1 | 0 |

Na het uitvoeren van $Act[1]$: $G =$ | 0 | 0 | 1 | 0 | 1 |

Na het uitvoeren van $Act[2]$: $G =$ | 0 | 1 | 0 | 1 | 0 |

Na het uitvoeren van $Act[3]$: $G =$ | 1 | 0 | 1 | 0 | 1 |

Nu is de vraag wat het effect zal zijn van enkele programma's op het volgende lichtsnoer G:

$G =$ | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

Q5(a) [1 pt]	**Hoe ziet G eruit na uitvoering van $Act =$** | SL | INV | SR | **?**
Q5(b) [1 pt]	**Hoe ziet G eruit na uitvoering van $Act =$** | SL | SL | SL | **?**
Q5(c) [1 pt]	**Hoe ziet G eruit na uitvoering van $Act =$** | SL | INV | SR | SR | INV | SR | INV | ON | **?**
Q5(d) [1 pt]	**Hoe ziet G eruit na uitvoering van $Act =$** | INV | SR | SR | SR | INV | SR | OFF | INV | **?**

Om een meer realistische simulator te maken, voegen we de mogelijkheid toe om de instructie $\mathsf{Loop}(T)$ uit te voeren, waarbij T een strikt positief geheel getal is. Deze instructie moet verplicht aan het eind van de array Act komen. T stelt het aantal keren voor dat de voorgaande instructies herhaald moeten worden. Op die manier wordt het programma SL SR $\mathsf{Loop}(3)$ uitgevoerd als SL SR SL SR SL SR

We vragen je opnieuw om het effect van enkele programma's na te gaan op het lichtsnoer G dat hierboven werd gegeven.

Q5(e) [1 pt]	**Hoe ziet G eruit na uitvoering van** $Act =$ SL | SR | $\mathsf{Loop}(30)$ **?**

Q5(f) [1 pt]	**Hoe ziet G eruit na uitvoering van** $Act =$ SL | INV | SR | INV | $\mathsf{Loop}(6205)$ **?**

Q5(g) [1 pt]	**Hoe ziet G eruit na uitvoering van** $Act =$ SL | SL | SL | $\mathsf{Loop}(36)$ **?**

Q5(h) [1 pt]	**Hoe ziet G eruit na uitvoering van** $Act =$ SL | INV | SR | $\mathsf{Loop}(245)$ **?**

Nu willen we de simulator zelf programmeren. Vul het volgende programma aan, zodat het het effect simuleert van een bepaalde actie op de array G. Dit kan met behulp van de volgende **`for`**-lus, mogelijk voorafgegaan en/of gevolgd door twee bijkomende instructies:

```
[...] // (1)
for (i ← 0 to [...] step 1) // (2)
{
  G[i] ← [...] // (3)
}
[...] // (4)
```

Bijvoorbeeld, om de operatie OFF te simuleren, moet je bij (2) $N - 1$ invullen, bij (3) de waarde 0, en (1) en (4) leeg laten.

Q5(i) [3 ptn]	**Welke expressies moet je invullen bij `(1)`, `(2)`, `(3)` en `(4)` om de operatie INV te simuleren? (Schrijf « leeg » indien nodig. Je mag bijkomende variabelen gebruiken).**

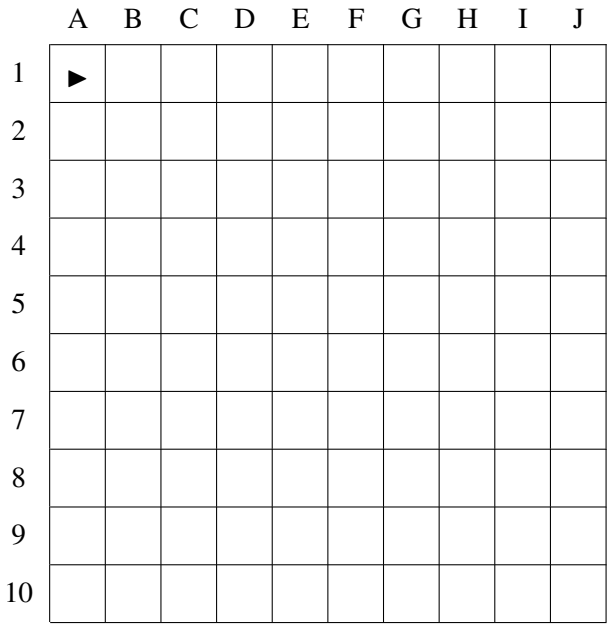
Q5(j) [3 ptn]	**Welke expressies moet je invullen bij `(1)`, `(2)`, `(3)` en `(4)` om de operatie SL te simuleren? (Schrijf « leeg » indien nodig. Je mag bijkomende variabelen gebruiken).**

Vraag 6 – Volg de robot (10 min – 8 ptn)

Hieronder vind je een reeks 2D-ruimtes waarin telkens een robot staat afgebeeld. De robot heeft een bepaalde positie in de ruimte, en ook een oriëntatie (noord = ▲, oost = ►, zuid = ▼, west = ◄).

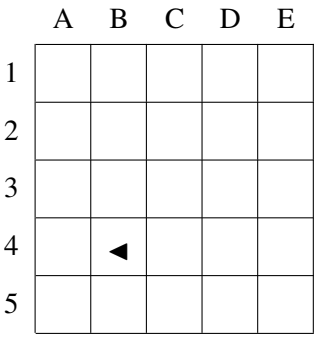
De robot wordt aangestuurd met behulp van een computerprogramma. Telkens wanneer het programma de procedure `Stap` aanroept, beweegt de robot 1 vakje volgens zijn huidige oriëntatie. Telkens wanneer het programma de procedure `Draai` aanroept, draait de robot ter plekke 90° in wijzerzin. De robot kan zich niet buiten de ruimte bewegen: wanneer hij tegen de rand botst, blijft hij gewoon op hetzelfde vakje staan (de `Stap` operatie heeft dan geen effect).

Gegeven zijn drie ruimtes waarin telkens de startpositie en -oriëntatie van de robot worden aangeduid. Naast elke ruimte staat een programma. Gevraagd wordt om te voorspellen wat de **positie** van de robot is **na** het uitvoeren van het programma. De positie wordt aangeduid met een letter en een cijfer (de startpositie van de robot in de eerste ruimte is bijvoorbeeld **A1**).



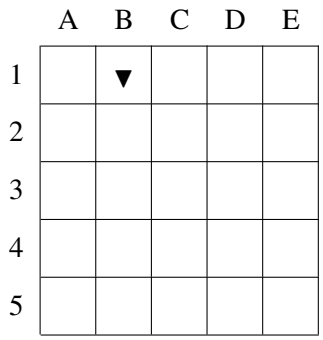

```
x ← 1
while (x < 9)
{
  Stap()
  Draai()
  Stap()
  for (y ← 0 to 2 step 1)
  {
    Draai()
  }
  x ← x + 1
}
```

Q6(a) [2 ptn]	**Wat is de positie van de robot na het uitvoeren van bovenstaand programma ?**




```
x ← 0
while (x < 66)
{
  Draai()
  Stap()
  Stap()
  x ← x + 1
}
```

Q6(b) [3 ptn]	**Wat is de positie van de robot na het uitvoeren van bovenstaand programma ?**




```
x ← 1
y ← 1
z ← 0

while (x + y < 10)
{
  x ← x + 1
  y ← y + x
  Draai()
  z ← y
  while (z > 0)
  {
    z ← z − 1
    Stap()
  }
}
```

Q6(c) [3 ptn]	**Wat is de positie van de robot na het uitvoeren van bovenstaand programma ?**

Vraag 7 – De indringer (20 min – 20 ptn)

De stad Taipei is erg trots om de gaststad te mogen zijn voor de Internationale Informatica-olympiade. Het stadsbestuur heeft beslist om speciaal voor de gelegenheid een nieuwe noedelfabriek te bouwen, om dit traditionele gerecht te kunnen aanbieden aan de vele deelnemers. De machine in de fabriek moet noedelslierten produceren die allemaal een verschillende lengte hebben. De lengte van een noedelsliert kan uitgedrukt worden in mm als een *geheel getal*, waarbij we beginnen vanaf 0 (dat is dan een microscopisch korte sliert). Helaas is de machine ergens vastgelopen, zodat er per ongeluk twee slierten van dezelfde lengte in de portie noedels terechtkomen. Je moet zo snel mogelijk vinden waar de fout zit.

Je moet een algoritme schrijven om dit op te lossen. Als input krijgt jouw programma een array arr van lengte n. Daarin komen alle getallen van 0 tot en met $n-2$ éénmaal voor in een willekeurige volgorde, behalve één getal dat tweemaal voorkomt. Jouw programma moet dat getal teruggeven. Bijvoorbeeld, als $arr = [2, 4, 0, 4, 1, 3]$ en $n = 6$, moet jouw algoritme 4 teruggeven.

De stadsbestuurders hebben in paniek al een schets van een programma op papier gezet. Het loopt de hele array af en controleert voor elk element van de array of het getal dat erin staat nogmaals voorkomt in een van de daaropvolgende elementen. Help ze om het programma te vervolledigen:

```
Input: een array arr van lengte n die gehele getallen bevat
          van 0 tot en met n−2, waarbij één getal tweemaal voorkomt.

Output: Na uitvoering, moet sol het getal bevatten dat tweemaal voorkomt in arr.

  sol ← −1

  for (i ← 0 to n − 1 step 1)
  {
    for (j ←[...] to n − 1 step 1)       // (a)
    {
      if (arr[i] = arr[j])
      {
        sol ← arr[i]
      }
    }
  }
```

Q7(a) [4 ptn]	**Welke expressie moet ingevuld worden bij** `(a)` **?**

Al snel merk je op dat, omdat er meer dan 100 000 slierten van verschillende lengte zijn, dit algoritme meer dan 10 miljard keer de "`if`"-instructie moet uitvoeren. Op die manier zal het programma niet op tijd klaar zijn voor de start van de olympiade. Een van de organisatoren stelt voor om het programma te optimaliseren, door gebruik te maken van een tussentijdse array. Vervolledig het volgende algoritme:

```
Input: een array arr van lengte n die gehele getallen bevat
          van 0 tot en met n − 2, waarbij één getal tweemaal voorkomt.

Output: Na uitvoering, moet sol het getal bevatten dat tweemaal voorkomt in arr.

arr2 ← een array van lengte n, geïnitialiseerd met waarde 0 op elke index.
sol ← −1

for (i ← 0 to n − 1 step 1)
{
  if ([...])                    // (b)
  {
    [...]                       // (c)
  }
  else
  {
    sol ← arr[i]
  }
}
```

Q7(b) [4 ptn]	**Welke expressie zoeken we bij** `(b)` **?**

Q7(c) [4 ptn]	**Welke expressie zoeken we bij** `(c)` **– het gaat om een toewijzing ?**

Helaas komt een verantwoordelijk van de fabriek je vertellen dat de noedelmachine erg weinig RAM-geheugen heeft, en dat die grote tussentijdse array van lengte n daarom niet gebruikt kan worden. Er bestaat een oplossing voor dit probleem waarbij slechts één getal (dat willekeurig groot kan worden) gebruikt wordt als variabele, en dat tegelijk de twee in elkaar geneste "`for`"-lussen ontwijkt. Kan je vinden hoe?

```
Input: een array arr van lengte n die gehele getallen bevat
          van 0 tot en met n − 2, waarbij één getal tweemaal voorkomt.

Output: Na uitvoering, moet sol het getal bevatten dat tweemaal voorkomt in arr.

sol ← −1
x ← 0

for (i ← 0 to n − 1 step 1)
{
  x ← [...]                     // (d)
}
sol ← [...]                     // (e)
```

Q7(d) [4 ptn]	**Welke expressie zoeken we bij** `(d)` **?**

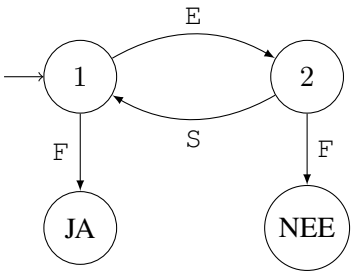
Q7(e) [4 ptn]	**Welke expressie zoeken we bij** `(e)` **?**

Vraag 8 – De machine van Alan (25 min – 22 ptn)

Om fraude bij examens te vermijden, hebben veel universiteiten (en scholen) de volgende regel: niemand mag het examenlokaal nog binnenkomen, zodra er één student is buitengegaan. Je goede vriend Alan wil een systeem op punt stellen om dit soort fraude feilloos te detecteren. Het plan van Alan is als volgt: je zet jezelf aan de ingang van de examenlokaal, en je noteert op een lint de volgorde van binnenkomen en buitengaan: een `E` wanneer iemand binnenkomt, een `S` wanneer iemand buitengaat, en één enkele `F` aan het einde van jouw observaties, dit alles in de volgorde waarin het gebeurt. Bijvoorbeeld, als een student binnenkomt, dan een student buitengaat, en daarna opnieuw een student binnenkomt, noteer je het volgende aan het begin van het lint:

E	S	E	F	...

Alan steekt dit lint in een machine die hij speciaal hiervoor heeft uitgevonden. Die machine leest het lint, en zal je zeggen of de volgorde van letters op het lint geldig is. Alan legt uit dat je de machine kan programmeren, en dat het programma wordt voorgesteld op een diagram zoals dat hieronder:



Op dit diagram noemen we elke cirkel een *staat*. Er zijn drie speciale staten: de staten JA en NEE waarmee de machine aangeeft dat een reeks letters op het lint wel of niet geldig is, en de beginstaat waar de machine is als hij begint te werken. Op dit voorbeeld is de beginstaat staat 1, en we geven dit aan met de kleine pijl links die uit het niets vertrekt. De pijlen noemen we *transities*, en ze laten de machine toe om een letter van het lint te lezen en naar een andere staat te gaan. De machine kan zich slechts in 1 staat tegelijk bevinden.

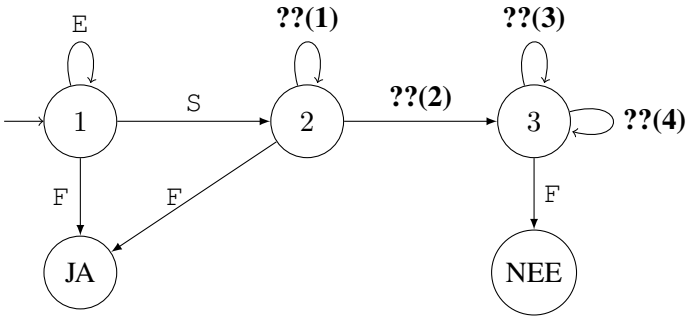
Om letters te lezen van het lint, heeft de machine een leeskop. Die bevindt zich op het eerste (het meeste linkse) element van het lint als de machine begint te werken. Als de machine zich in een bepaalde staat bevindt, kan ze een transitie uitvoeren die vertrekt uit deze staat, op voorwaarde dat de letter die verbonden is aan die transitie zich op dat moment onder de leeskop bevindt. Als dat het geval is, wordt de leeskop een element verder geplaatst (ze gaat naar de volgende letter op het lint), en komt de machine terecht in de staat aan het einde van de pijl die ze heeft gevolgd.

Bijvoorbeeld, voor het lint dat hierboven is gegeven, bevindt de machine zich eerst in staat 1, met de leeskop op de eerste letter `E`. Ze kan dus de transitie uitvoeren van staat 1 naar staat 2, waarna de leeskop zich bevindt op de letter `S`. Daarop kan ze de transitie naar staat 1 volgen, waarna de leeskop zich bevindt op de tweede `E`, enzovoort. De uitvoering eindigt in de staat NEE.

In welke staat zal de machine eindigen als we ze de volgende linten geven?

		Lint
Q8(a) [2 ptn]	JA / NEE	E S E S F ...
Q8(b) [2 ptn]	JA / NEE	F ...

Om het binnenkomen en buitengaan van de studenten te controleren, stelt Alan het volgende programma voor. Helaas zijn enkele letters uitgewist (ze zijn vervangen door vraagtekens in het diagram). Alan wil dat de machine in staat NEE terecht komt *enkel en alleen als een student binnenkomt nadat een andere reeds is buitengegaan*.

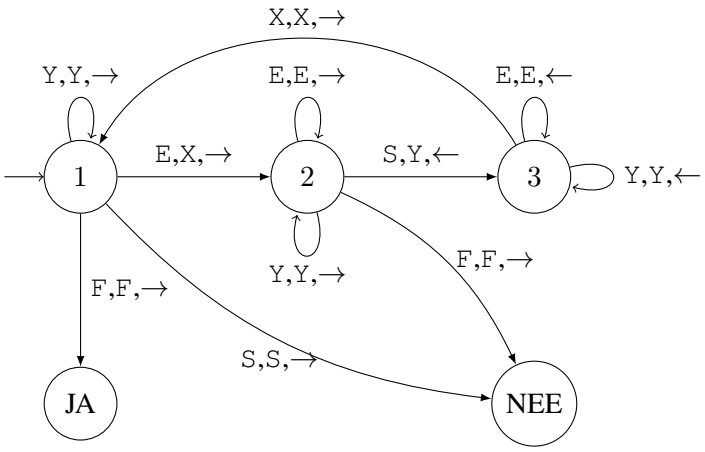


Help Alan door de 4 letters te geven die **??(1)**, **??(2)**, **??(3)** en **??(4)** moeten vervangen (in die volgorde)

Q8(c) [4 ptn]	**Geef de vier letters waardoor ??(1), ??(2), ??(3) en ??(4) vervangen moeten worden (in die volgorde).**

Alan stelt je nu het laatste snufje van zijn machine voor. Die kan nu, bij elke transitie, het karakter dat op het lint wordt gelezen ook vervangen (de leeskop kan nu ook schrijven). Bovendien wordt nu bij elke transitie ook aangegeven in welke richting de leeskop nadien moet bewegen (de leeskop kan nu dus ook naar links, terug naar delen van het lint die voordien al zijn gelezen). De aanwijzingen bij de transities zijn nu van de vorm $\texttt{A}, \texttt{B}, d$. Om een transitie uit te voeren, moet de machine `A` lezen op de plaats van de leeskop, waarna ze die letter moet vervangen door `B` (het is mogelijk dat $\texttt{A} = \texttt{B}$), en tot slot de leeskop in de richting d moet bewegen. Die richting kan $\leftarrow$ (naar links) en $\rightarrow$ (naar rechts) zijn.

Voor deze upgrade van de machine, stelt Alan voor het volgende programma te gebruiken:



Stel dat het lint nu de volgende reeks letters bevat:

E	S	F	...

Q8(d) [2 ptn]	**In welke staat zal de machine dan eindigen?**

Q8(e) [2 ptn]	**Wat zal de inhoud van het lint zijn nadat de machine alles heeft uitgevoerd?**

Dezelfde vragen voor het volgende lint:

E	E	S	F	...

Q8(f) [2 ptn]	**In welke staat zal de machine nu eindigen?**

Q8(g) [2 ptn]	**Wat zal de inhoud van het lint zijn nadat de machine alles heeft uitgevoerd?**

Je geeft de machine een stel linten die bestaan uit: een reeks `E`'s, gevolgd door een reeks `S`'en, gevolgd door één enkele `F`. Onder welke voorwaarde(n) zal de machine zo'n lint gegarandeerd altijd geldig verklaren?

		Voorwaarde
Q8(h) [2 ptn]	Gegarandeerd / Niet gegarandeerd	Het aantal `E`'s is gelijk aan het aantal `S`'en
Q8(i) [2 ptn]	Gegarandeerd / Niet gegarandeerd	Het aantal `E`'s is groter dan of gelijk aan het aantal `S`'en
Q8(j) [2 ptn]	Gegarandeerd / Niet gegarandeerd	Het aantal `E`'s is strikt groter dan het aantal `S`'en

Vul hier uw antwoorden in !

	JA	NEE	Persoon	/6
Q1(a) /1	☐	☐	Charlotte	
Q1(b) /1	☐	☐	Lucas	
Q1(c) /1	☐	☐	Damien	
Q1(d) /1	☐	☐	Lucie	
Q1(e) /1	☐	☐	Daan	
Q1(f) /1	☐	☐	Peter	

Q2(a) **Een nummer tussen 1 en 4**	/6
Q3(a) **Een expressie**	/3
Q3(b) **Een expressie**	/3
Q3(c) **Een expressie**	/2
Q3(d) **Een expressie**	/2
Q3(e) **Een expressie**	/2
Q4(a) **Een expressie**	/3
Q4(b) **Een expressie**	/3
Q4(c) **Een expressie**	/5
Q4(d) **Een expressie**	/5
Q4(e) **Een expressie**	/4

Q5(a) **De inhoud van** G	/1
Q5(b) **De inhoud van** G	/1
Q5(c) **De inhoud van** G	/1
Q5(d) **De inhoud van** G	/1
Q5(e) **De inhoud van** G	/1
Q5(f) **De inhoud van** G	/1
Q5(g) **De inhoud van** G	/1
Q5(h) **De inhoud van** G	/1
Q5(i) **De expressies op plaatsen (1), (2), (3) en (4)**	/3
Q5(j) **De expressies op plaatsen (1), (2), (3) en (4)**	/3
Q6(a) **Een positie**	/2
Q6(b) **Een positie**	/3
Q6(c) **Een positie**	/3
Q7(a) **Een expressie**	/4
Q7(b) **Een expressie**	/4

Q7(c) **Een instructie (toewijzing)**	/4
Q7(d) **Een expressie**	/4
Q7(e) **Een expressie**	/4

	JA	NEE	Lint	
Q8(a) /2	☐	☐	E S E S F …	/4
Q8(b) /2	☐	☐	F …	

Q8(c) **Vier letters**	/4
Q8(d) **Een staat**	/2
Q8(e) **De inhoud van het lint**	/2
Q8(f) **Een staat**	/2
Q8(g) **De inhoud van het lint**	/2

	Gegarandeerd	Niet gegarandeerd	Voorwaarde	
Q8(h) /2	☐	☐	Het aantal `E`'s is gelijk aan het aantal `S`'en	/6
Q8(i) /2	☐	☐	Het aantal `E`'s is groter dan of gelijk aan het aantal `S`'en	
Q8(j) /2	☐	☐	Het aantal `E`'s is strikt groter dan het aantal `S`'en	