be-OI 2015 **Halve Finale** woensdag 19 november 2014	**Invullen in HOOFDLETTERS en LEESBAAR aub** VOORNAAM : .... NAAM : .... SCHOOL :	100 **Gereserveerd**

Belgische Informatica-olympiade (duur : 3u maximum)

Oplossingen

Dit is de vragenlijst van de halve finale van de Belgische Informatica-olympiade 2015. Ze bevat negen vragen, en je krijgt **maximum 3u** de tijd om ze op te lossen. Naast elke vraag vind je 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 (desnoods op de achterkant). 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, JavaScript, (Visual) Basic). 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, onder begeleiding, zolang er toezicht verzekerd blijft in het lokaal. 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 ← 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    // 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 … 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 – De Goeden en de Valsen (16 ptn)

Op het eiland Sergio Leone zijn er twee kampen : de Goeden, die altijd de waarheid zeggen, en de Valsen, die altijd liegen. Je kan enkel weten tot welk kamp iemand behoort, als je de waarheid van hun uitspraken onderzoekt. Duid van de volgende stellingen aan of je *zeker* kan zijn dat ze waar zijn. Als je met de gegeven informatie niet kan bepalen of een stelling waar of niet waar is, kruis dan « *niet zeker* » aan.

Arthur zegt : *Ik ben een Goede*.

	Zeker	Niet zeker	Stelling
Q1(a) [1 pt]	☐	☒	Arthur is een Goede
Q1(b) [1 pt]	☐	☒	Arthur is een Valse

Candice zegt : *Ik zit in hetzelfde kamp als Brice*.

	Zeker	Niet zeker	Stelling
Q1(c) [1 pt]	☒	☐	Brice is een Goede
Q1(d) [1 pt]	☐	☒	Candice is een Goede

Danaë zegt : *Slechts één van deze twee klopt : ofwel is er een schat op het eiland, ofwel ben ik een Valse*.

	Zeker	Niet zeker	Stelling
Q1(e) [1 pt]	☐	☒	Danaë is een Valse
Q1(f) [1 pt]	☒	☐	Er ligt een schat op het eiland

Ethan zegt : *Francis is een Valse*. Francis antwoordt : *dat is fout*.

	Zeker	Niet zeker	Stelling
Q1(g) [1 pt]	☐	☒	Francis is een Valse
Q1(h) [1 pt]	☐	☒	Ethan is een Valse
Q1(i) [1 pt]	☒	☐	Een van de twee is een Goede, de ander is een Valse

Gaëlle zegt : *Hubert en Irène zitten in hetzelfde kamp*.

	Zeker	Niet zeker	Stelling
Q1(j) [1 pt]	☒	☐	Minstens één van de drie is een Goede
Q1(k) [1 pt]	☐	☒	Minstens twee van de drie zijn Goeden
Q1(l) [1 pt]	☒	☐	Een oneven aantal van deze drie zijn Goeden
Q1(m) [1 pt]	☐	☒	Minstens één van deze drie is een Valse

Louis, die geen Dorpeling is, zegt : *alle Dorpelingen zijn Valsen*. Een Dorpeling zegt : *Louis is een Goede*. Een andere Dorpeling zegt : *Louis is een Valse*.

	Zeker	Niet Zeker	Stelling
Q1(n) [1 pt]	☐	☒	Louis is een Goede
Q1(o) [1 pt]	☒	☐	Louis is een Valse
Q1(p) [1 pt]	☒	☐	Sommige Dorpelingen zijn Valsen

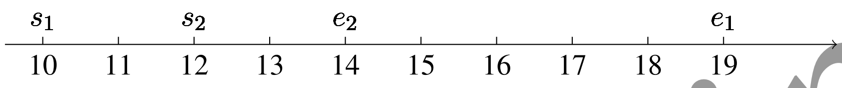
Vraag 2 – De technicus (12 ptn)

Een technicus moet reparaties uitvoeren bij klant 1 en klant 2. De technicus werkt in shiften van een uur, die telkens beginnen op klokslag het uur (dus : om 10u00 begint een shift die duurt tot 10u59, en om 11u begint de volgende shift). Voor de reparatie bij klant 1 heeft hij d_1 shiften nodig ($d_1 > 0$). Bij klant 2 heeft hij d_2 shiften nodig ($d_2 > 0$). Zodra hij begint aan een reparatie, werkt hij die in één keer af. Hij doet ook geen twee reparaties tegelijkertijd.

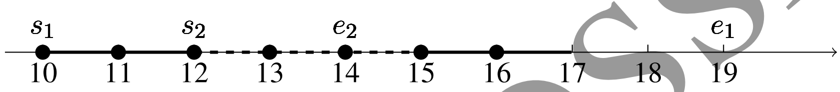
Klant 1 geeft de technicus een tijdsvenster waarbinnen die mag langskomen voor de reparatie : een interval $[s_1, e_1]$, waarbij s_1 en e_1 volle uren (gehele getallen) zijn. Dit wil zeggen dat de technicus een shift kan beginnen ten vroegste op uur s_1 en ten laatste op uur e_1. Bijvoorbeeld, als $[s_1, e_1] = [10, 12]$, kan de technicus ten hoogste 3 shiften werken : een uur vanaf 10u, een vanaf 11u, en een vanaf 12u. Ook klant 2 geeft op deze manier een tijdsvenster mee, dat we met $[s_2, e_2]$ aanduiden.

Je mag aannemen dat de tijdsvensters groot genoeg zijn om de volledige reparatie te kunnen doen : $e_1 - s_1 + 1 \geq d_1$ en $e_2 - s_2 + 1 \geq d_2$. We nemen ook aan dat er geen extra tijd nodig is om van klant 1 naar klant 2 te gaan.

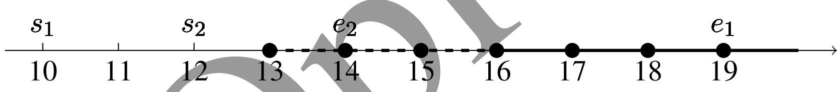
We willen weten of het mogelijk is de twee reparaties uit te voeren, gegeven de beperkingen die de klanten opleggen. Bijvoorbeeld, als $[s_1, e_1] = [10, 19]$, $[s_2, e_2] = [12, 14]$, $d_1 = 4$ en $d_2 = 3$:



Als de technicus werkt bij klant 1 (volle lijn) om 10u en 11u, daarna bij klant twee (stippellijn) om 12u, 13u en 14u, en dan opnieuw bij klant 1 om 15u en 16u, dan :

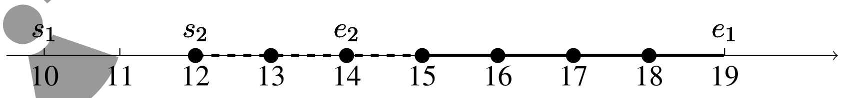


Dit is geen goede oplossing, omdat de technicus zo niet kan werken aan één stuk door bij klant 1. Ook de volgende situatie is geen goede oplossing :



omdat de technicus de beperkingen van klant 2 zo niet respecteert.

De volgende oplossing respecteert wel alle opgelegde beperkingen :



Er bestaat dus een oplossing voor het voorbeeld dat hierboven werd beschreven.

Bij de volgende vragen moet je altijd exact 1 hokje aankruisen (er is steeds slechts 1 correct antwoord)

Q2(a) [6 ptn]		**Onder welke voorwaarde bestaat er een oplossing om de reparaties uit te voeren binnen de tijdsvensters bepaald door de klanten ?**
(1)	☐	$(s_1 + d_1 > e_2 - d_2)$ of $(s_2 + d_2 > e_1 - d_1)$
(2)	☐	$s_1 \neq s_2$ en $e_1 \neq e_2$
(3)	☒	$(s_1 + d_1 - 1 \leq e_2 - d_2)$ of $(s_2 + d_2 - 1 \leq e_1 - d_1)$
(4)	☐	$(e_2 - d_1 - d_2) \geq s_1$

Q2(b) [6 ptn]		**Als er een oplossing bestaat, wat is dan het vroegste moment dat de technicus kan beginnen aan de reparatie bij klant 1 ?**
(1)	☐	$e_2 + 1$
(2)	☐	$s_2 + d_2$
(3)	☐	$e_2 - d_2 - d_1 + 1$
(4)	☐	$\max(s_2 + d_2, s_1)$
(5)	☐	s_1
(6)	☒	als $(s_1 + d_1 - 1 \leq e_2 - d_2)$ dan s_1 , anders $\max(s_2 + d_2, s_1)$

Vraag 3 – Black & white (22 ptn)

Voor jou staat een reeks gesloten, ondoorzichtige dozen. Elke doos bevat ofwel een witte bal, ofwel een zwarte bal. De exacte beginconfiguratie is onbekend : je weet in het begin niet welke doos welke kleur bevat.

We vragen je om de best mogelijke strategieën te vinden om een reeks problemen op te lossen. De beste strategieën zijn diegene die je toelaten het probleem op te lossen in zo weinig mogelijk stappen in het allerslechtste geval : dat wil zeggen, in het geval dat de beginsituatie zo ongunstig mogelijk is.

Voorbeeld :

Configuratie : Er zijn 9 dozen en slechts één zwarte bal in totaal.
Probleem : Bepaal de positie van de zwarte bal, op zo'n manier dat je ook in het slechtste geval zo weinig mogelijk dozen moeten openen.
Vraag : Hoeveel dozen moet je openen, met de beste mogelijke strategie, in het slechtste geval ?
Antwoord : 8
Verklaring : Het is nodig alle dozen één voor één te openen, bijvoorbeeld van links naar rechts. Dat is de beste strategie. In het slechtste geval bevindt de zwarte bal zich dan in één van de laatste twee dozen. Als ze niet in de 8ste doos zit, moet je de 9de niet meer open doen : je weet immers al zeker dat ze zich daar in moet bevinden.

Probleem 1 :

Configuratie : Er zijn 9 dozen en 7 witte ballen in totaal.
Probleem : Bepaal de positie van één van de zwarte ballen, op zo'n manier dat je ook in het slechtste geval zo weinig mogelijk dozen moeten openen.

Q3(a) [3 ptn]	Hoeveel dozen moet je openen, met de beste strategie, in het slechtste geval ?
Oplossing: 7	

Probleem 2 :

Configuratie : De dozen bevatten afwisselend een witte en zwarte bal (bijvoorbeeld $WZWZW$ of ZWZ). Er zijn n dozen.
Probleem : Bepaal het totaal aantal zwarte ballen, op zo'n manier dat je ook in het slechtste geval zo weinig mogelijk dozen moeten openen.

Q3(b) [2 ptn]	Als $n = 3$, hoeveel dozen moet je dan openen, met de beste strategie, in het slechtste geval ?
Oplossing: 1	

Q3(c) [2 ptn]	Als $n = 8$, hoeveel dozen moet je dan openen, met de beste strategie, in het slechtste geval ?
Oplossing: 0	

Q3(d) [2 ptn]	Als $n = 85$, hoeveel dozen moet je dan openen, met de beste strategie, in het slechtste geval ?
Oplossing: 1	

Probleem 3 :

Configuratie : Van links naar rechts, bevatten de dozen eerst enkel witte ballen, en dan enkel zwarte ballen. (bijvoorbeeld $WZZZZ$, $WWWZZ$ of ZZ). Er zijn n dozen.
Probleem : Bepaal het totaal aantal zwarte ballen, op zo'n manier dat je ook in het slechtste geval zo weinig mogelijk dozen moeten openen.

Q3(e) [2 ptn]	**Als $n = 3$, hoeveel dozen moet je dan openen, met de beste strategie, in het slechtste geval ?**
Oplossing: 2	

Q3(f) [2 ptn]	**Als $n = 15$, hoeveel dozen moet je dan openen, met de beste strategie, in het slechtste geval ?**
Oplossing: 4	

Q3(g) [3 ptn]	**Als $n = 82$, hoeveel dozen moet je dan openen, met de beste strategie, in het slechtste geval ?**
Oplossing: 7	

Probleem 4 :

Configuratie : Er zijn n dozen.
Probleem : Groepeer alle witte ballen aan de linkerkant en alle zwarte ballen aan de rechterkant (zoals in probleem 3). Je mag enkel de twee ballen uit dozen die vlak naast elkaar liggen met elkaar omwisselen. Doe dit op zo'n manier dat je ook in het slechtste geval zo weinig mogelijk omwisselingen moet doen.

Q3(h) [2 ptn]	**Als $n = 4$, hoeveel omwisselingen moet je dan doen, met de beste strategie, in het slechtste geval ?**
Oplossing: 4	

Q3(i) [2 ptn]	**Als $n = 20$, hoeveel omwisselingen moet je dan doen, met de beste strategie, in het slechtste geval ?**
Oplossing: 100	

Q3(j) [2 ptn]	**Als $n = 21$, hoeveel omwisselingen moet je dan doen, met de beste strategie, in het slechtste geval ?**
Oplossing: 110	

Vraag 4 – Fraudebestrijding (18 ptn)

De nieuwe regering van Palombië wil jacht maken op fraudeurs om de nationale economie te stimuleren. In hun vizier : bedrijfjes die hun werknemers teveel vakantiedagen geven, want dat haalt de productiviteit onderuit, die zo al niet hoog is. . .

Zodus worden alle bedrijven verplicht om aan de overheid hun aantal werknemers n door te geven, en ook een tabel `vakantie` van n rijen en 366 kolommen waarin elk element een Booleaanse waarde bevat (**`true`** of **`false`**). De tabel `vakantie` geeft aan wanneer een werknemer (met een uniek identificatienummer i dat tussen 0 en $n-1$ ligt) een verlofdag krijgt : element `vakantie[`i`][`j`]` bevat **`true`** als werknemer i verlof heeft op de j^{de} dag van het jaar, en anders **`false`**.

De overheid wil de volgende drie regels opleggen :

1. een werknemer heeft recht op 40 dagen (!) verlof per jaar ;
2. op ieder moment moet minstens de helft van de werknemers aanwezig zijn in het bedrijf ;
3. een vakantieperiode mag niet langer duren dan 15 dagen.

Een ambtenaar kreeg de taak een programma te schrijven om zo'n tabel te analyseren en te ontdekken of er werd gefraudeerd. Jammer genoeg is de ambtenaar nog snel op verlof vertrokken voor drie maanden, voordat hij zijn werk kon afmaken. . . De regering van Palombië rekent op jouw hulp om de ontbrekende stukjes aan te vullen, en belooft je drie weken betaald verlof in het paradijselijke Santa Banana als je daarin slaagt !

(Opmerking : in pseudocode wordt de tabel voorgesteld met een tweedimensionale array (een array van arrays).)

```
Input : een tweedimensionale array vakantie[n][366], een werknemersnummer e
Output: true als werknemer e niet meer dan 40 dagen verlof heeft per jaar,
        anders false.

sum ← [...]           // (a)
i ← 0
while (i < 366)
{
  if (vakantie[e][i] = true)
  {
     [...]                        //  (b)
  }
  i ← i+1
}
if (sum > 40)
{
  return [...]                    // (c)
}
else
{
  return [...]                    // (d)
}
```

Q4(a) [2 ptn]	Welke expressie zoeken we bij `(a)` ?
Oplossing: 0	

Q4(b) [2 ptn]	Welke expressie zoeken we bij (b) ?
Oplossing: sum ← sum+1	

Q4(c) [1 pt]	Welke expressie zoeken we bij (c) ?
Oplossing: **false**	

Q4(d) [1 pt]	Welke expressie zoeken we bij (d) ?
Oplossing: **true**	

```
Input : een tweedimensionale array vakantie[n][366]
Output: true als op elke dag van het jaar steeds minstens de helft van de werknemers
        werkt, anders false

for (i ← 0 to 365 step 1)
{
  [...]                           // (e)
  for (j ← [...] to [...] step 1) // (f), (g)
  {
    if ([...] = true)             // (h)
    {
      sum ← sum + 1
    }
  }
  if (sum > n/2)
  {
    return false
  }
}
return true
```

Q4(e) [2 ptn]	Welke instructie zoeken we bij (e) ?
Oplossing: sum ← 0	

Q4(f) [1 pt]	Welke expressie zoeken we bij (f) ?
Oplossing: 0	

Q4(g) [1 pt]	Welke expressie zoeken we bij (g) ?
Oplossing: $n-1$	

Q4(h) [2 ptn]	Welke expressie zoeken we bij (h) ?
Oplossing: `vakantie[j][i]`	

```
Input : een tweedimensionale array vakantie[n][366], een werknemersnummer e
```

```
Output: true als werknemer e nooit langer dan 15 aaneengesloten dagen met verlof is,
        anders false

i ← 0
while(i < 366)
{
  if (vakantie[e][i] = true)
  {
    [...]                       // (i)
    while ([...])               // (j)
    {
      i ← i+1
    }
    if (i-begin > 15)
    {
      return false
    }
  }
  else
  {
    i ← i+1
  }
}
return true
```

Q4(i) [3 ptn]	Welke instructie zoeken we bij `(i)` ?
	Oplossing: `begin←i`

Q4(j) [3 ptn]	Welke voorwaarde zoeken we bij `(j)` ?
	Oplossing: `i < 366 en vakantie[e][i] = true`

Vraag 5 – Recursieve functies (14 ptn)

Je herinnert je misschien de Fibonacci-getallen uit de lessen wiskunde. Het 0-de Fibonacci-getal is 0 en het 1-ste Fibonacci-getal is 1. Voor alle getallen $n > 1$ is het n-de Fibonacci-getal de som van het $(n-1)$-de en $(n-2)$-de Fibonacci-getal. De eerste acht Fibonacci-getallen zijn dus $0, 1, 1, 2, 3, 5, 8, 13$. De Fibonacci-getallen kunnen wiskundig als volgt gedefinieerd worden :

$$\mathrm{Fib}(0) = 0$$
$$\mathrm{Fib}(1) = 1$$
$$\mathrm{Fib}(n) = \mathrm{Fib}(n-1) + \mathrm{Fib}(n-2), \text{ voor } n > 1$$

Dit is wat men een *recursieve definitie* noemt : de functie Fib wordt gedefinieerd in functie van zichzelf. In een programmeertaal kunnen we zo'n recursieve definitie makkelijk neerschrijven als een functie die zichzelf oproept :

```
Input  : n, een natuurlijk getal waarvoor we het Fibonacci-getal willen berekenen
Output : het n-de Fibonacci-getal

Fib(n)
{
  if (n = 0)
  {
    return 0
  }
  else if (n = 1)
  {
    return 1
  }
  else
  {
    return Fib(n−1) + Fib(n−2)
  }
}
```

Q5(a) [1 pt]	Wat is het resultaat van de functie-oproep Fib(9) ?
Oplossing: 34	

Q5(b) [2 ptn]	Hoeveel keer roept de functie Fib zichzelf op nadat we Fib(2) oproepen ?
Oplossing: 2 keer	

Q5(c) [3 ptn]	Hoeveel keer roept de functie Fib zichzelf op nadat we Fib(5) oproepen ?
Oplossing: 14 keer	

Slimme informatici hebben een andere (hopelijk betere) manier gevonden om Fibonacci-getallen te berekenen. We kunnen deze in een programmeertaal neerschrijven als de volgende (recursieve) functie BetereFib :

```
Input  : n, een natuurlijk getal waarvoor we het Fibonacci-getal willen berekenen
         a, een natuurlijk getal dat initieel 0 is
         b, een natuurlijk getal dat initieel 1 is
```

```
        i, een natuurlijk getal dat initieel 0 is
Output : het n-de Fibonacci-getal

BetereFib(n, a, b, i) {
  if (i = n)
  {
    return a
  }
  else
  {
    return BetereFib(n, b, a+b, i+1)
  }
}
```

Om het n-de Fibonacci-getal te berekenen, roep je $\mathrm{BetereFib}(n, 0, 1, 0)$ op. Wellicht begrijp je de bovenstaande code beter wanneer je beseft dat, bij elke oproep van BetereFib, a steeds het i-de Fibonacci-getal zal bevatten, en b steeds het $(i + 1)$-de Fibonacci-getal zal bevatten.

Q5(d) [4 ptn]	**Hoeveel keer roept de functie** BetereFib **zichzelf op nadat we** $\mathrm{BetereFib}(2, 0, 1, 0)$ **oproepen ?**
Oplossing: 2 keer	

Q5(e) [4 ptn]	**Hoeveel keer roept de functie** BetereFib **zichzelf op nadat we** $\mathrm{BetereFib}(5, 0, 1, 0)$ **oproepen ?**
Oplossing: 5 keer	

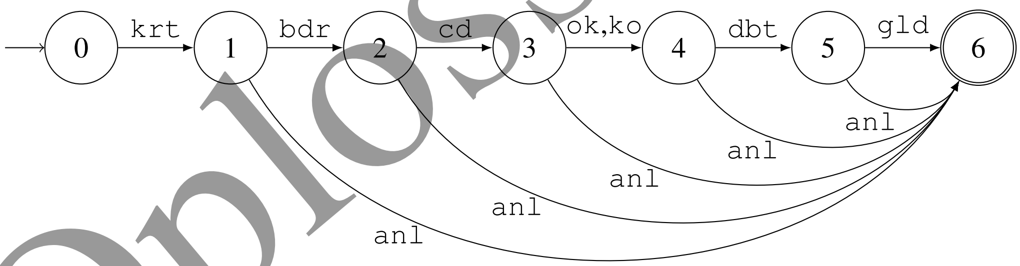
Vraag 6 – De geldautomaat van de Dagobert bank (26 ptn)

In een poging te moderniseren heeft oom Dagobert besloten geldautomaten te maken voor zijn bank. Om de kosten te drukken heeft hij het ontwerp van de machines uitbesteed aan zijn neefjes Kwik, Kwek en Kwak.

Ze komen op de proppen met een automaat die de volgende acties kan herkennen of uitvoeren :

Actie	Beschrijving
`krt`	De gebruiker steekt zijn bankkaart in de automaat om te beginnen.
`bdr`	De gebruikt typt het gevraagde bedrag in.
`cd`	De gebruiker typt de code van de kaart in.
`ok`	De automaat herkent de code als juist.
`ko`	De automaat herkent de code als fout.
`dbt`	Het bedrag wordt van de rekening van de gebruiker gehaald.
`gld`	De automaat geeft het geld en de kaart terug.
`anl`	De gebruiker drukt op de toets « annuleren » en de kaart wordt teruggegeven.

Om precies te kunnen beschrijven *in welke volgorde deze acties na elkaar uitgevoerd kunnen worden*, stellen Kwik, Kwek en Kwak het volgende diagram voor :



Op dit diagram stelt iedere cirkel een *staat* voor waarin de automaat zich kan bevinden (we hebben ze genummerd van 0 tot 6), en elke pijl stelt een *mogelijke actie* voor. Op ieder moment bevindt de automaat zich in slechts één staat. In het begin is de automaat in staat 0, aangeduid door de kleine pijl links. Elke actie doet de automaat van staat veranderen, aangegeven door de bijhorende pijl. Bijvoorbeeld, als de automaat in staat 1 is, zijn de acties `bdr` en `anl` mogelijk : het uitvoeren van de actie `bdr` brengt de automaat in staat 2, het uitvoeren van `anl` brengt de automaat in staat 6. Wanneer een pijl meerdere acties bevat, mag je exact 1 (en niet meer) van deze acties kiezen om van de ene staat naar de andere te gaan. Bijvoorbeeld, om van 3 naar 4 te gaan, moet ofwel de actie `ok` ofwel de actie `ko` ondernomen worden. Tot slot is staat 6 hier de eindstaat (aangegeven door de dubbele rand) : de transactie is gedaan wanneer de automaat in deze staat aankomt.

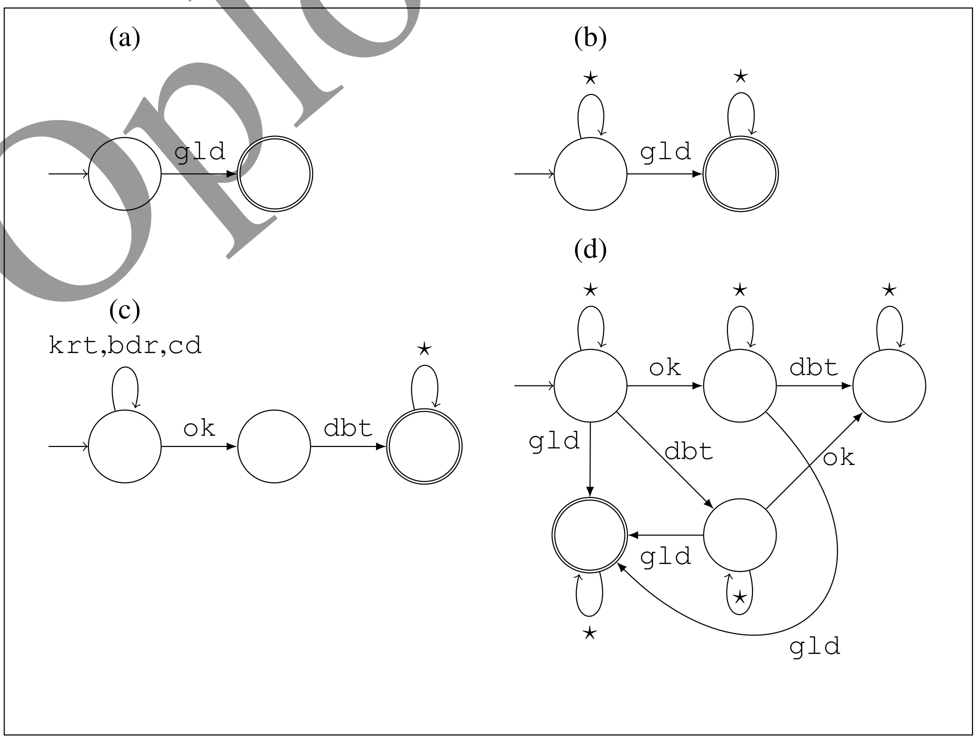
Alle mogelijke scenario's die de automaat aankan, worden in dit schema voorgesteld. Bijvoorbeeld, de gebruiker kan zijn kaart insteken (actie `krt`, de automaat komt in staat 1), daarna een bedrag ingeven (actie `bdr`, de automaat komt in staat 2), dan zich bedenken en op annuleren drukken (actie `anl`, de automaat komt in eindstaat 6). Zoals je kan zien, heeft de automaat van Kwik, Kwek en Kwak enkele gebreken. . . . Welke van de volgende scenario's kunnen uitgevoerd worden (d.w.z. beginnend in de beginstaat en eindigend in de eindstaat) met het diagram hierboven ?

	Mogelijk	Niet mogelijk	Is het mogelijk om . . . ?
Q6(a) [2 ptn]	☐	☒	geld te ontvangen (actie `gld`) zonder een code te hebben ingegeven (actie `cd`).
Q6(b) [2 ptn]	☒	☐	geld van je rekening te zien gaan (actie `dbt`) zonder dat geld ook te ontvangen (actie `gld`).
Q6(c) [2 ptn]	☒	☐	geld te ontvangen (actie `gld`) zonder de correcte code te hebben gegeven (actie `ok`).
Q6(d) [2 ptn]	☐	☒	geld te ontvangen (actie `gld`) zonder een kaart te hebben ingevoerd (actie `krt`).

Oom Dagobert is niet heel gelukkig - dit kost hem tijd, en tijd is geld ! Hij beslist dat de automaat ten allen tijde de volgende voorwaarde moet respecteren :

> **Als** geld wordt gegeven (actie `gld`) **dan** moet de gebruiker een correcte code hebben gegeven (actie `ok`) **en** moet het geld van de rekening zijn gehaald (actie `dbt`).

Om dat duidelijk te maken gebruikt Dagobert eenzelfde soort diagram als dat van Kwik, Kwek en Kwak om *enkele probleemgevallen* te beschrijven. Hij maakte de 4 diagrammen hieronder. Het symbool $\star$ wordt hier gebruikt ter vereenvoudiging, en betekent « alle andere acties ». Bijvoorbeeld op diagram (b) hieronder, betekent het symbool $\star$ op de lus van de beginstaat : « alle acties behalve `gld` », omdat `gld` al verschijnt op de pijl tussen de beginstaat en de eindstaat. Met andere woorden, we kunnen $\star$ daar vervangen door « `krt`,`bdr`,`cd`,`ok`,`ko`,`gld`,`dbt`,`anl` ». Het symbool $\star$ op de lus van de eindstaat betekent daarentegen wel « eender welke actie », omdat er vanuit de eindstaat geen enkele andere pijl vertrekt.



Zoek uit in welke mate deze diagrammen Dagoberts voorwaarde schenden (het maakt hier niet uit of ze door de automaat worden toegelaten of niet). Net zoals in het eerder gegeven diagram, komt een scenario overeen met een pad door het diagram vertrekkend van de beginstaat en eindigend in de eindstaat. Een pad dat niet eindigt in de eindstaat moet je negeren.

	Ja	Nee	Is het waar dat . . . ?
Q6(e) [2 ptn]	☒	☐	alle mogelijke scenario's in diagram (a) de voorwaarde schenden.
Q6(f) [2 ptn]	☒	☐	er een scenario bestaat in diagram (a) dat de voorwaarde schendt.
Q6(g) [2 ptn]	☐	☒	alle mogelijke scenario's in diagram (b) de voorwaarde schenden.
Q6(h) [2 ptn]	☒	☐	er een scenario bestaat in diagram (b) dat de voorwaarde schendt.
Q6(i) [2 ptn]	☐	☒	alle mogelijke scenario's in diagram (c) de voorwaarde schenden.
Q6(j) [2 ptn]	☐	☒	er een scenario bestaat in diagram (c) dat de voorwaarde schendt.
Q6(k) [3 ptn]	☒	☐	alle mogelijke scenario's in diagram (d) de voorwaarde schenden.
Q6(l) [3 ptn]	☒	☐	er een scenario bestaat in diagram (d) dat de voorwaarde schendt.

Vraag 7 – Van X naar Y (20 ptn)

In een lang vervlogen tijd, toen er nog geen computers bestonden, moesten mensen inventief zijn om de saaie dagen van een leven zonder videogames door te komen. Op een wel heel wanhopig moment kwam een groep vrienden op dit idee : een persoon kiest twee strikt positieve gehele getallen, en de anderen moeten een manier vinden om van het ene naar het andere te gaan volgens enkele vooraf vastgestelde regels.

Neem bijvoorbeeld als reglement aan dat we bij een getal 10 mogen optellen en 1 mogen aftrekken. Om op die manier van 7 naar 25 te gaan, kunnen we de volgende stappen doen :

$$7 \xrightarrow{+10} 17 \xrightarrow{+10} 27 \xrightarrow{-1} 26 \xrightarrow{-1} 25$$

Om dat wat interessanter te maken, hebben ze beslist een extra voorwaarde toe te voegen : het pad dat gevonden wordt moet zo weinig mogelijk stappen bevatten. Het voorbeeld hierboven bevat er 4, aangeduid door pijlen.

Gegeven bepaalde regels en twee gehele getallen, moet je in deze vraag het **kleinst mogelijke** aantal stappen zoeken dat nodig is om van het eerste getal naar het tweede te gaan. Als het niet mogelijk is om van het ene naar het andere getal te gaan, antwoord dan -1.

— **Reglement 1 :** je mag bij een getal 5 optellen, 5 aftrekken, 5 vermenigvuldigen, of het door 5 delen, zolang na elke stap het tussenresultaat een strikt positief geheel getal is (delen door 5 kan je dus enkel bij veelvouden van vijf). Bijvoorbeeld, een correct pad zou zijn :

$$10 \xrightarrow{-5} 5 \xrightarrow{\div 5} 1 \xrightarrow{+5} 6$$

Q7(a) [2 ptn]	**Hoeveel stappen zijn nodig om van 2 naar 3 te gaan ? Antwoord -1 als dit onmogelijk is.**
Oplossing: 3	

Q7(b) [2 ptn]	**Hoeveel stappen zijn nodig om van 15 naar 7 te gaan ? Antwoord -1 als dit onmogelijk is.**
Oplossing: 3	

Q7(c) [2 ptn]	**Hoeveel stappen zijn nodig om van 30 naar 60 te gaan ? Antwoord -1 als dit onmogelijk is.**
Oplossing: 4	

— **Reglement 2 :** je mag 3 optellen bij een getal, of twee cijfers van een getal verwisselen zolang het tussenresultaat op die manier niet begint met 0. Bijvoorbeeld, een correct pad zou zijn :

$$24 \xrightarrow{+3} 27 \xrightarrow{\leftrightarrow} 72 \xrightarrow{+3} 75 \xrightarrow{+3} 78$$

Q7(d) [2 ptn]	**Hoeveel stappen zijn nodig om van 13 naar 55 te gaan ? Antwoord -1 als dit onmogelijk is.**
Oplossing: 6	

Q7(e) [2 ptn]	**Hoeveel stappen zijn nodig om van 12 naar 37 te gaan ? Antwoord -1 als dit onmogelijk is.**
Oplossing: -1	

Q7(f) [2 ptn]	**Hoeveel stappen zijn nodig om van 98 naar 14 te gaan ? Antwoord -1 als dit onmogelijk is.**
Oplossing: 8	

Q7(g) [2 ptn]	**Hoeveel stappen zijn nodig om van 99 naar 36 te gaan ? Antwoord -1 als dit onmogelijk is.**
Oplossing: -1	

— **Reglement 3 :** je mag een getal vermenigvuldigen met 2, of er 1 bij optellen ; je mag ook alle cijfers weghalen uit het getal die groter dan of gelijk zijn aan 5, tegelijkertijd, op voorwaarde dat het tussenresultaat minstens één cijfer bevat en dat het eerste cijfer ervan niet 0 is. Bijvoorbeeld, een correct pad zou zijn :

$$24 \xrightarrow{\times 2} 48 \xrightarrow{\cancel{8}} 4 \xrightarrow{\times 2} 8 \xrightarrow{+1} 9$$

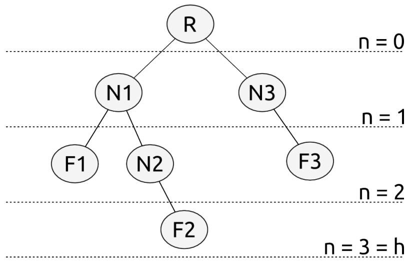
Q7(h) [2 ptn]	**Hoeveel stappen zijn nodig om van 5 naar 4 te gaan ? Antwoord -1 als dit onmogelijk is.**
Oplossing: 5	

Q7(i) [2 ptn]	**Hoeveel stappen zijn nodig om van 20 naar 3 te gaan ? Antwoord -1 als dit onmogelijk is.**
Oplossing: 5	

Q7(j) [2 ptn]	**Hoeveel stappen zijn nodig om van 113 naar 1 te gaan ? Antwoord -1 als dit onmogelijk is.**
Oplossing: 6	

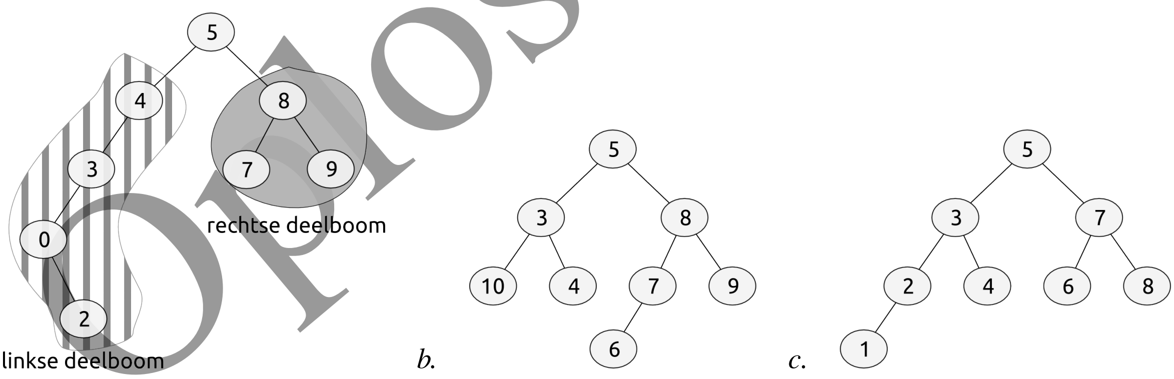
Vraag 8 – Gebalanceerde binaire zoekbomen (25 ptn)

Om op een efficiënte manier gegevens terug te kunnen vinden, worden ze soms opgeslagen in de vorm van een binaire zoekboom. Zo'n boom wordt meestal voorgesteld zoals in de afbeelding hieronder. Een element van een boom (hier getekend als ovaal) noemen we een **knoop**. Bij een binaire boom kan iedere knoop een **ouder** zijn van maximaal twee **kinderen**. Op de afbeelding is N1 de **ouder** van twee **kinderen** : F1 en N2. De knopen onderaan, die geen kinderen hebben, zijn de **bladeren** van de boom. Elke boom heeft ook exact één element zonder ouder : het bovenste element dat we de **wortel** noemen. Een knoop kan een **waarde** bevatten, in deze vraag steeds één uniek getal.



De **hoogte** van een boom is het aantal knopen dat men moet doorlopen om van de wortel naar het verst verwijderde blad te gaan. In het voorbeeld hierboven is de hoogte van de boom **3** omdat je door 3 knopen heen moet om van de wortel naar het verste blad daarvandaan (F2) te gaan. Het **niveau** van een knoop is de afstand ervan tot de wortel, zo ligt bvb. blad F3 op niveau 2.

Een binaire boom is **gebalanceerd** als voor elke knoop van de boom geldt dat de hoogtes van diens **deelbomen** (die van het linkerkind en die van het rechterkind) niet meer dan 1 van elkaar verschillen. In de volgende voorbeelden is de boom links niet gebalanceerd, omdat de twee deelbomen van knoop **5** een hoogte hebben die meer dan 1 verschilt : de hoogte van de linkerdeelboom is 3 en die van de rechterdeelboom is 1 (en $3 - 1 > 1$).



a. *b.* *c.*

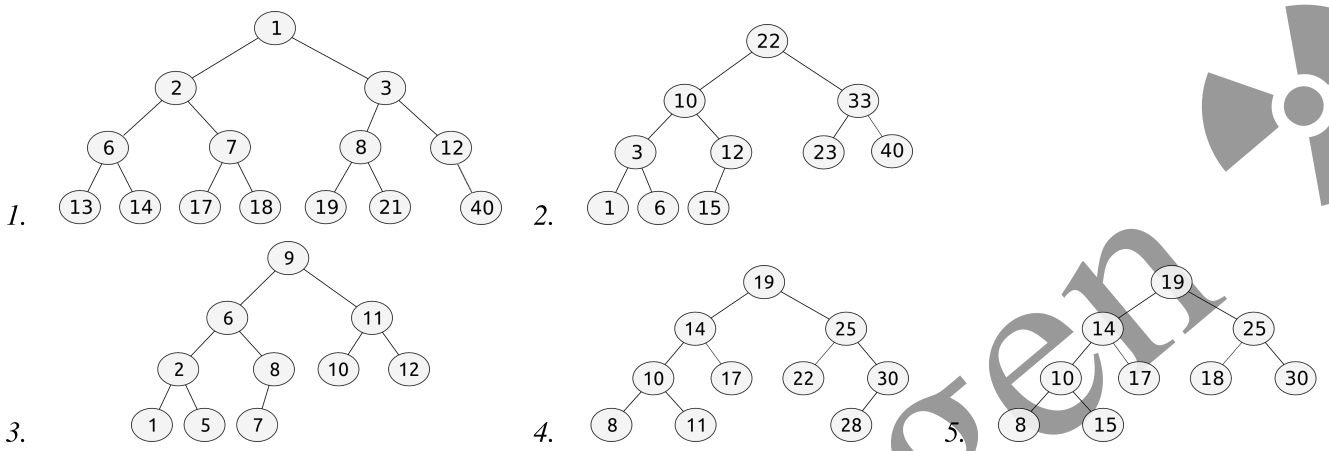
Q8(a) [2 ptn]	**Wat is het niveau van knoop 4 in boom *a.* hierboven ?**
	Oplossing: 1

Q8(b) [2 ptn]	**Wat is de hoogte van boom *b.* hierboven ?**
	Oplossing: 3

Q8(c) [2 ptn]	**Wat is de hoogte van de deelboom van knoop 3 in boom *c.* hierboven ?**
	Oplossing: 2

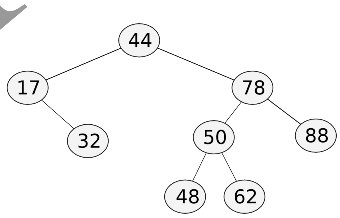
Een bijzonder soort zoekboom is de **AVL**-boom (genoemd naar de uitvinders ervan : Georgy **A**delson-**V**elsky en Evguenii **L**andis). AVL-bomen zijn gebalanceerde zoekbomen waarbij alle knopen n aan de volgende voorwaarde voldoen : alle knopen van de linkerdeelboom van n hebben een waarde die *kleiner is dan* n, en alle knopen van de rechterdeelboom hebben een waarde die *groter is dan* n. Boom b hierboven is gebalanceerd maar niet AVL, boom c is wel AVL.

Zeg van elk van de 5 bomen hieronder of het een AVL-boom is of niet.

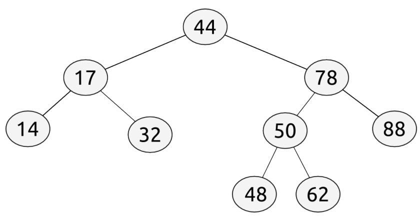


	AVL	Niet-AVL	Nummer van de boom
Q8(d) [1 pt]	☐	☒	Boom *1.*
Q8(e) [1 pt]	☐	☒	Boom *2.*
Q8(f) [1 pt]	☒	☐	Boom *3.*
Q8(g) [1 pt]	☒	☐	Boom *4.*
Q8(h) [1 pt]	☐	☒	Boom *5.*

Bekijk nu de volgende AVL-boom :

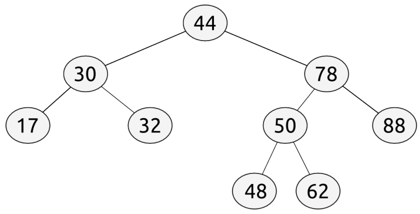


We willen deze boom aanpassen door er waarden aan toe te voegen of uit te verwijderen, op zo'n manier dat de eigenschappen van een AVL-boom behouden blijven, maar wel door zo weinig mogelijk knopen te verplaatsen. We beschouwen een knoop als verplaatst als de waarde van diens ouder is veranderd. Bij elke vraag moet je de AVL-boom tekenen die je bekomt als je de gevraagde bewerking hebt uitgevoerd. **Vertrek voor elke vraag opnieuw van de initiële boom.**

Q8(i) [2 ptn]	**Teken de AVL-boom die je bekomt na het toevoegen van de waarde 14 aan de initiële boom.**
	Oplossing: 

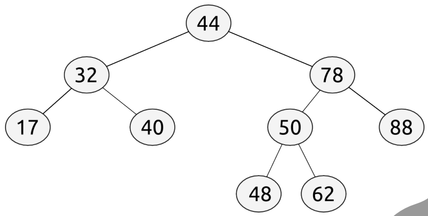
Q8(j) [2 ptn]	**Teken de AVL-boom die je bekomt na het toevoegen van de waarde 30 aan de initiële boom.**

Oplossing:



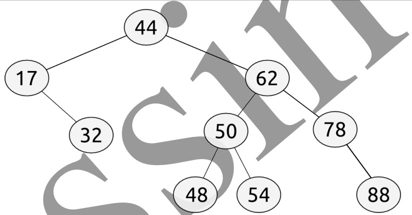
Q8(k) [2 ptn]	**Teken de AVL-boom die je bekomt na het toevoegen van de waarde 40 aan de initiële boom.**

Oplossing:



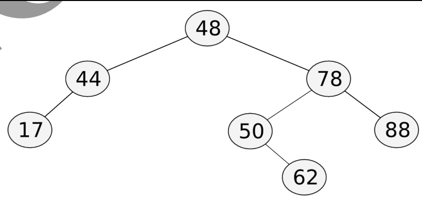
Q8(l) [4 ptn]	**Teken de AVL-boom die je bekomt na het toevoegen van de waarde 54 aan de initiële boom.**

Oplossing:



Q8(m) [4 ptn]	**Teken de AVL-boom die je bekomt na het verwijderen van de waarde 32 uit de initiële boom.**

Oplossing:



Vraag 9 – Dubbele 1 (19 ptn)

Schrijf een programma die elke "1" die voorkomt in een array van n getallen "verdubbelt". Bijvoorbeeld, als je de array $[1, 1, 5, 1, 4]$ krijgt, moet jouw programma dit veranderen in $[1, 1, 1, 1, 5, 1, 1, 4]$. Om de zaken te vereenvoudigen, krijgt jouw programma alvast een array van lengte $2n$, zodat je de gegeven array kan aanpassen zonder er een nieuwe te moeten maken.

Dit is de definitie van de input en output van het algoritme.

```
Input : n, een geheel getal.
        arr, een array van lengte 2n die gehele getallen bevat.
Output: arr is aangepast zodat elk getal 1 onder de eerste n getallen
        tweemaal voorkomt (het aantal 1-en is verdubbeld).
```

We stellen je twee algoritmes voor waarmee je die taak kan uitvoeren, je moet ze allebei aanvullen.

Algoritme 1

```
count ← 0
for (i ← 0 to [...] step 1)        // (a)
{
  if (arr[[...]] = 1)                      // (b)
  {
    for (j ← [...] to i+1 step -1)        // (c)
    {
      arr[[...]] ← arr[[...]] // (d), (e)
    }
    count ← count + 1
  }
}
```

Q9(a) [2 ptn]	Welke expressie zoeken we bij (a) ?
Oplossing: $n - 1$	

Q9(b) [3 ptn]	Welke expressie zoeken we bij (b) ?
Oplossing: $i + count$	

Q9(c) [2 ptn]	Welke expressie zoeken we bij (c) ?
Oplossing: n	

Q9(d) [2 ptn]	Welke expressie zoeken we bij (d) ?
Oplossing: $j + count$	

Q9(e) [2 ptn]	Welke expressie zoeken we bij (e) ?
Oplossing: $j + count - 1$	

Algoritme 2

```
count ← 0
for (i ← 0 to n − 1 step 1)
{
  if (arr[i] = 1)
  {
    count ← count + 1
  }
}
for (j ← [...] to [...] step [...])   // (f), (g), (h)
{
  [...]                          // (i)
  if (arr[j] = 1)
  {
    arr[j+[...]] ← 1 // (j)
    count ← count − 1
  }
}
```

Q9(f) [2 ptn]	Welke expressie zoeken we bij `(f)` ?
Oplossing: $n-1$	

Q9(g) [1 pt]	Welke expressie zoeken we bij `(g)` ?
Oplossing: 0	

Q9(h) [1 pt]	Welke expressie zoeken we bij `(h)` ?
Oplossing: -1	

Q9(i) [5 ptn]	Welke instructie zoeken we bij `(i)` ?
Oplossing: $arr[j + count] \leftarrow arr[j]$	

Q9(j) [4 ptn]	Welke expressie zoeken we bij `(j)` ?
Oplossing: $count - 1$	

Als je weet dat het ongeveer 8 minuten duurt om algoritme 1 uit te voeren op een moderne computer als n gelijk is aan 1 000 000,

Q9(k) [5 ptn]		**Hoeveel tijd heeft diezelfde computer nodig om algoritme 2 uit te voeren ?**
(1)	☒	Ongeveer 10 milliseconden
(2)	☐	Ongeveer 4 minuten
(3)	☐	Ongeveer 8 minuten
(4)	☐	Ongeveer 15 minuten
(5)	☐	Meerdere dagen

Vul hier uw antwoorden in !

	Zeker	Niet zeker	Stelling	/2
Q1(a) /1	☐	☐	Arthur is een Goede	
Q1(b) /1	☐	☐	Arthur is een Valse	

	Zeker	Niet zeker	Stelling	/2
Q1(c) /1	☐	☐	Brice is een Goede	
Q1(d) /1	☐	☐	Candice is een Goede	

	Zeker	Niet zeker	Stelling	/2
Q1(e) /1	☐	☐	Danaë is een Valse	
Q1(f) /1	☐	☐	Er ligt een schat op het eiland	

	Zeker	Niet zeker	Stelling	/3
Q1(g) /1	☐	☐	Francis is een Valse	
Q1(h) /1	☐	☐	Ethan is een Valse	
Q1(i) /1	☐	☐	Een van de twee is een Goede, de ander is een Valse	

	Zeker	Niet zeker	Stelling	/4
Q1(j) /1	☐	☐	Minstens één van de drie is een Goede	
Q1(k) /1	☐	☐	Minstens twee van de drie zijn Goeden	
Q1(l) /1	☐	☐	Een oneven aantal van deze drie zijn Goeden	
Q1(m) /1	☐	☐	Minstens één van deze drie is een Valse	

	Zeker	Niet Zeker	Stelling	/3
Q1(n) /1	☐	☐	Louis is een Goede	
Q1(o) /1	☐	☐	Louis is een Valse	
Q1(p) /1	☐	☐	Sommige Dorpelingen zijn Valsen	

Q2(a)		**Kies alleen maar EEN antwoord !**	/6
(1)	☐	$(s_1 + d_1 > e_2 - d_2)$ of $(s_2 + d_2 > e_1 - d_1)$	
(2)	☐	$s_1 \neq s_2$ en $e_1 \neq e_2$	
(3)	☐	$(s_1 + d_1 - 1 \leq e_2 - d_2)$ of $(s_2 + d_2 - 1 \leq e_1 - d_1)$	
(4)	☐	$(e_2 - d_1 - d_2) \geq s_1$	

Q2(b)		**Kies alleen maar EEN antwoord !**	/6
(1)	☐	$e_2 + 1$	
(2)	☐	$s_2 + d_2$	
(3)	☐	$e_2 - d_2 - d_1 + 1$	
(4)	☐	$\max(s_2 + d_2, s_1)$	
(5)	☐	s_1	
(6)	☐	als $(s_1 + d_1 - 1 \leq e_2 - d_2)$ dan s_1 , anders $\max(s_2 + d_2, s_1)$	

Q3(a)	**Een getal**	/3
Q3(b)	**Een getal**	/2
Q3(c)	**Een getal**	/2
Q3(d)	**Een getal**	/2
Q3(e)	**Een getal**	/2
Q3(f)	**Een getal**	/2
Q3(g)	**Een getal**	/3
Q3(h)	**Een getal**	/2
Q3(i)	**Een getal**	/2
Q3(j)	**Een getal**	/2
Q4(a)	**Een expressie**	/2
Q4(b)	**Een expressie**	/2

be-OI 2015 woensdag 19 november 2014	**Gereserveerd** 100

Q4(c)	**Een expressie**	/1
Q4(d)	**Een expressie**	/1
Q4(e)	**Een instructie**	/2
Q4(f)	**Een expressie**	/1
Q4(g)	**Een expressie**	/1
Q4(h)	**Een expressie**	/2
Q4(i)	**Een instructie**	/3
Q4(j)	**Een voorwaarde**	/3
Q5(a)	**Een natuurlijk getal**	/1
Q5(b)	**Een aantal oproepen**	/2
Q5(c)	**Een aantal oproepen**	/3
Q5(d)	**Een aantal oproepen**	/4
Q5(e)	**Een aantal oproepen**	/4

	Mogelijk	Niet mogelijk	Is het mogelijk om ... ?	/8
Q6(a) /2	☐	☐	geld te ontvangen (actie `gld`) zonder een code te hebben ingegeven (actie `cd`).	
Q6(b) /2	☐	☐	geld van je rekening te zien gaan (actie `dbt`) zonder dat geld ook te ontvangen (actie `gld`).	
Q6(c) /2	☐	☐	geld te ontvangen (actie `gld`) zonder de correcte code te hebben gegeven (actie `ok`).	
Q6(d) /2	☐	☐	geld te ontvangen (actie `gld`) zonder een kaart te hebben ingevoerd (actie `krt`).	

	Ja	Nee	Is het waar dat ... ?	/18
Q6(e) /2	☐	☐	alle mogelijke scenario's in diagram (a) de voorwaarde schenden.	
Q6(f) /2	☐	☐	er een scenario bestaat in diagram (a) dat de voorwaarde schendt.	
Q6(g) /2	☐	☐	alle mogelijke scenario's in diagram (b) de voorwaarde schenden.	
Q6(h) /2	☐	☐	er een scenario bestaat in diagram (b) dat de voorwaarde schendt.	
Q6(i) /2	☐	☐	alle mogelijke scenario's in diagram (c) de voorwaarde schenden.	
Q6(j) /2	☐	☐	er een scenario bestaat in diagram (c) dat de voorwaarde schendt.	
Q6(k) /3	☐	☐	alle mogelijke scenario's in diagram (d) de voorwaarde schenden.	
Q6(l) /3	☐	☐	er een scenario bestaat in diagram (d) dat de voorwaarde schendt.	

Q7(a)	**Het kleinst mogelijke aantal stappen, of** -1	/2
Q7(b)	**Het kleinst mogelijke aantal stappen, of** -1	/2
Q7(c)	**Het kleinst mogelijke aantal stappen, of** -1	/2
Q7(d)	**Het kleinst mogelijke aantal stappen, of** -1	/2
Q7(e)	**Het kleinst mogelijke aantal stappen, of** -1	/2
Q7(f)	**Het kleinst mogelijke aantal stappen, of** -1	/2
Q7(g)	**Het kleinst mogelijke aantal stappen, of** -1	/2

Q7(h) **Het kleinst mogelijke aantal stappen, of** -1 /2

. . . .

Q7(i) **Het kleinst mogelijke aantal stappen, of** -1 /2

. . . .

Q7(j) **Het kleinst mogelijke aantal stappen, of** -1 /2

. . . .

Q8(a) **Het niveau van knoop 4 in boom *a*.** /2

. . . .

Q8(b) **De hoogte van boom *b*.** /2

. . . .

Q8(c) **de hoogte van de deelboom van knoop 3 in boom *c*.** /2

. . . .

	AVL	Niet-AVL	Nummer van de boom	/5
Q8(d) /1	☐	☐	Boom *1*.	
Q8(e) /1	☐	☐	Boom *2*.	
Q8(f) /1	☐	☐	Boom *3*.	
Q8(g) /1	☐	☐	Boom *4*.	
Q8(h) /1	☐	☐	Boom *5*.	

Q8(i) **De boom na het toevoegen van de waarde 14.** /2

Q8(j) **De boom na het toevoegen van de waarde 30.**	/2
Q8(k) **De boom na het toevoegen van de waarde 40.**	/2
Q8(l) **De boom na het toevoegen van de waarde 54.**	/4

Q8(m)	**De boom na het verwijderen van de waarde 32**	/4
Q9(a)	**Een expressie**	/2
Q9(b)	**Een expressie**	/3
Q9(c)	**Een expressie**	/2
Q9(d)	**Een expressie**	/2
Q9(e)	**Een expressie**	/2
Q9(f)	**Een expressie**	/2
Q9(g)	**Een expressie**	/1
Q9(h)	**Een expressie**	/1
Q9(i)	**Een instructie**	/5
Q9(j)	**Een expressie**	/4

Q9(k)		Kies alleen maar EEN antwoord !	/5
(1)	☐	Ongeveer 10 milliseconden	
(2)	☐	Ongeveer 4 minuten	
(3)	☐	Ongeveer 8 minuten	
(4)	☐	Ongeveer 15 minuten	
(5)	☐	Meerdere dagen	