be-OI 2017 **Finale - BELOFTEN** zaterdag 11 maart 2017	**Invullen in HOOFDLETTERS en LEESBAAR aub** VOORNAAM : ..... NAAM : ..... SCHOOL :	O **Gereserveerd**

Belgische Informatica-olympiade (duur : 2u maximum)

Dit is de vragenlijst van de finale van de Belgische Informatica-olympiade 2017. Ze bevat 11 vragen, en je krijgt **maximum 2u** de tijd om ze op te lossen.

Algemene opmerkingen (lees dit aandachtig voor je begint)

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

Veel succes !

Overzicht pseudo-code

Gegevens worden opgeslagen in variabelen. Je kan de waarde van een variabele veranderen met $\leftarrow$. In een variabele kunnen we gehele getallen, reële getallen of arrays opslaan (zie verder), en ook booleaanse (logische) waarden : waar/juist (**`true`**) of onwaar/fout (**`false`**). Op variabelen kan je wiskundige bewerkingen uitvoeren. Naast de klassieke operatoren $+$, $-$, $\times$ en $/$, kan je ook $\%$ gebruiken : als a en b allebei gehele getallen zijn, dan zijn a/b en $a\%b$ respectievelijk het quotiënt en de rest van de gehele deling (staartdeling). Bijvoorbeeld, als $a = 14$ en $b = 3$, dan geldt : $a/b = 4$ en $a\%b = 2$. In het volgende stukje code krijgt de variabele *leeftijd* de waarde 20.

```
geboortejaar ← 1997
leeftijd ← 2017 − 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

Bekijk de volgende code, waarvan het resultaat afhangt van een waarde $N \geq 1$. In deze code is de expressie $j\%2 = 0$ waar, als en slechts als j een even getal is.

```
Input : een waarde N ≥ 1

j ← 1
while (j < N)
{
  j ← j * 2
}

if (j%2 = 0)
{
  Print ''A''
}

if (j ≤ 100)
{
  if (j ≥ 1024)
  {
    Print ''B''
  }
  else
  {
    Print ''C''
  }
}
else
{
  if (j/2 > 511 and j/2 ≤ 512)
  {
    Print ''D''
  }
  else
  {
    Print ''E''
  }
}
```

Q1(a) [2 ptn]	**Wat print het programma uit als $N = 1$?**
Q1(b) [2 ptn]	**Wat print het programma uit als $N = 50$?**
Q1(c) [2 ptn]	**Wat print het programma uit als $N = 1024$?**
Q1(d) [2 ptn]	**Wat print het programma uit als $N = 1026$?**

Vraag 2 – In de rij per twee !

Een lerares wilt het aantal leerlingen in haar klas tellen. Ze weet dat het aantal leerlingen een even getal is en een macht van 2 is. Ze verzint drie manieren om de leerlingen te tellen.

De eerste manier gaat als volgt :
— Ze initialiseert een teller op 0 ;
— Ze wandelt langs *elke leerling*, en, bij elke leerling, *telt ze* 1 *op bij de teller* ;
— Wanneer ze langs alle leerlingen is gepasseerd, bevat de teller het aantal leerlingen.

De tweede manier gaat als volgt :
— Ze initialiseert een teller op 0 ;
— Ze passeert langs alle *paren* van leerlingen, en, bij elk paar, *telt ze* 2 *op* bij de teller ;
— Wanneer ze langs alle leerlingen is gepasseerd, bevat de teller het aantal leerlingen.

Met de derde manier, telt de lerares niet meer zelf, maar zet ze haar leerlingen aan het werk :
— Ze vraagt aan alle leerlingen om recht te staan, en om tegelijkertijd de volgende procedure uit te voeren :
  1. Iedere leerling onthoudt een teller die in het begin gelijk is aan 1, daarna :
  2. Alle rechtstaande leerlingen vormen groepjes van twee ;
  3. In ieder paar leerlingen, telt de grootste leerling bij zijn teller, de waarde op van de teller van de kleinste leerling. Daarna gaat de kleinste zitten.
  4. Herbegin vanaf stap 2 met alle leerlingen die nog rechtstaan.
— De procedure eindigt wanneer er nog maar 1 leerling rechtstaat. De lerares neemt aan dat de teller van deze laatste leerling gelijk is aan het aantal aanwezige leerlingen.

Q2(a) [2 ptn]	**Hoeveel optellingen doet de lerares met de eerste manier, als er 16 leerlingen zijn ?**

Q2(b) [2 ptn]	**Hoeveel optellingen doet de lerares met de tweede manier, als er 16 leerlingen zijn ?**

Q2(c) [3 ptn]	**Hoeveel optellingen heeft de laatst rechtstaande leerling gedaan met de derde manier, als er 16 leerlingen zijn ?**

Vraag 3 – Woordenboek

We willen de betekenis van een woord opzoeken in de Dikke Van Dale (waarin alle woorden zijn gesorteerd in alfabetische volgorde, zoals bij elk woordenboek). We hebben twee algoritmes om de betekenis van een woord te vinden :

Algoritme 1 :

```
Input : een reeks karakters woord, het op te zoeken woord

// open het woordenboek op de eerste pagina
while (niet aan het einde van het woordenboek)
{
    if (woord staat op de pagina) {
        // lees de betekenis
        return
    }
    else {
        // draai pagina om
    }
}
```

Algoritme 2 :

```
Input : een reeks karakters woord, het op te zoeken woord

// open het woordenboek in het midden
if (woord staat op deze pagina) {
    // lees de betekenis
    return
}
else {
    // scheur het woordenboek in twee
    if (woord staat voor de middelste pagina) {
        herbegin het algoritme met de eerste helft
    }
    else if (woord staat na de middelste pagina) {
        herbegin het algoritme met de tweede helft
    }
    else { // Vraag (d)
        return
    }
}
```

We willen het gedrag van deze twee algoritmes beter kennen. Wat ons interesseert zijn de slechtst en best mogelijke gevallen voor beide (wat betreft de snelheid van uitvoering).

Q3(a) [2 ptn]	**Wat is het *best mogelijke* geval voor het eerste algoritme?**
☐	het woord begint met A en staat op de eerste pagina.
☐	het woord staat in het midden van het woordenboek.
☐	het woord begint met Z en staat op de laatste pagina.
☐	het woord staat niet in het woordenboek.

Q3(b) [2 ptn]	**Wat is het *best mogelijke* geval voor het tweede algoritme?**
☐	het woord begint met A en staat op de eerste pagina.
☐	het woord staat in het midden van het woordenboek.
☐	het woord begint met Z en staat op de laatste pagina.
☐	het woord staat niet in het woordenboek.

Q3(c) [2 ptn]	**Wat is het *slechtst mogelijke* geval voor het tweede algoritme?**
☐	het woord begint met A en staat op de eerste pagina.
☐	het woord begint met A en staat op de laatste pagina.
☐	het woord staat in het midden van het woordenboek.
☐	het woord staat niet in het woordenboek.

Q3(d) [2 ptn]	**Wanneer komen we terecht in de laatste `else` van het tweede algoritme?**
☐	Wanneer het woord in de eerste helft van het woordenboek staat.
☐	Wanneer het woord niet in het woordenboek staat.
☐	Wanneer het woord in de tweede helft van het woordenboek staat.
☐	Wanneer het woord begint met A.

Vraag 4 – Bibliotheek

Elk boek heeft een uniek *International Standard Book Number* of ISBN. Dat bestaat uit 10 cijfers, en het laatste cijfer kan ook vervangen zijn door 'X'. Dat laatste cijfer is niet willekeurig : het wordt berekend met de volgende code.

```
Input  : p: een array van 9 cijfers
Output : een cijfer of het karakter 'X'.

a ← 0
j ← 10
for (i ← 0 to 8 step 1)
{
    a ← a + p[i] * j
    j ← j − 1
}
resultaat ← 11 − (a%11)
if (resultaat = 10)
{
    return 'X'
}
else
{
    return resultaat%11
}
```

Q4(a) [2 ptn]	**Wat is de waarde van variabele j na de uitvoering van dit programma ?**

Q4(b) [2 ptn]	**Wat is het kleinste getal dat dit programma kan teruggeven ?**

Q4(c) [2 ptn]	**Wat zijn de kleinste en de grootste waarden die de variabele $resultaat$ kan krijgen (in de lijn vlak na de lus ?)**

Q4(d) [4 ptn]	**Als de variabele a na de lus in dit programma gelijk is aan 35, wat geeft het programma dan terug ?**

Vraag 5 – Bingewatching

John heeft op zijn smartphone een app waarmee hij bijhoudt welke TV-series hij bekijkt. Dit is de informatie die John moet ingeven voor elke serie :

- **Naam** : de naam van de serie (bvb "Game of Thrones").
- **Aantal seizoenen** : het aantal seizoenen die momenteel bestaan (bvb 6 seizoenen).
- **Aantal afleveringen per seizoen** : het aantal afleveringen per seizoen [1] (bvb 10 afleveringen per seizoen).
- **Duur van een aflevering** : de tijdsduur in uren en minuten van een aflevering [2] (bvb : 1u10).
- **Afgelopen** : om aan te geven of een serie volledig is afgelopen of nog steeds wordt geproduceerd. Anders gezegd : is het laatste seizoen al helemaal uitgezonden of niet ?

Dit zijn alle series die John heeft ingegeven in deze app :

Naam	Aantal seizoenen	Afleveringen per seizoen	Duur aflevering	Afgelopen ?
Game of Thrones	6	10	1h09	Nee
Lost	6	20	0h42	Ja
Breaking Bad	5	12	0h50	Ja
The Walking Dead	7	13	1h06	Nee
Pretty Little Liars	7	21	0h44	Nee

Met deze app kan John ook bijhouden tot waar hij is geraakt bij het kijken van al deze series. Om dat te doen moet John bij het kijken ingeven welk seizoen hij bekijkt [3], het nummer van de aflevering die hij bekijkt [4] en het tijdstip waarop hij is gestopt met kijken.

Bijvoorbeeld, voor "Game of Thrones" is John nu bij seizoen 5, aflevering 8 en aan de 35ste minuut in deze aflevering. Zo weet John dat hij de volgende keer deze aflevering vanaf de 35e minuut kan starten om zo de serie verder te bekijken.

Hier is, voor elk van deze series, het moment waarop John is gestopt met kijken :

Naam	Bekeken seizoen	Bekeken aflevering	Timing
Game of Thrones	5	8	0u35
Lost	6	20	0u42
Breaking Bad	4	2	0u35
The Walking Dead	3	3	0u37
Pretty Little Liars	6	17	0u29

Q5(a) [2 ptn]	**Hoeveel minuten (in totaal) moet John nog bekijken om aan het einde van het laatste bestaande seizoen van *Game of Thrones* te raken ?**

Q5(b) [2 ptn]	**Hoeveel minuten (in totaal) moet John nog bekijken om aan het einde van het laatste bestaande seizoen van *Lost* te raken ?**

Q5(c) [2 ptn]	**Naar welke serie moet John nog het langst kijken om aan het einde van het laatste seizoen ervan te komen ?**

1. Je mag aannemen dat elk seizoen van eenzelfde serie hetzelfde aantal afleveringen heeft.
2. Je mag aannemen dat elke aflevering van eenzelfde serie even lang duurt.
3. We beginnen te tellen vanaf seizoen 1 voor elke serie
4. We beginnen te tellen vanaf aflevering 1 voor elke serie

Vraag 6 – Zandkastelen

Ter voorbereiding op je vakantie in Blankenberge bestudeer je de structuur van zand, zodat je de mooiste zandkastelen van het strand kan bouwen. Je goede vriend Abel stelt een computersimulatie voor :

Stel dat je een vierkant oppervlak hebt waarop je emmertjes zand omkeert. Ik verdeel dat oppervlak in 5×5 *vierkante vakjes. Dan kan ik mijn oppervlak voorstellen als een rooster waarvan elk vakje een aantal emmertjes zand bevat. Bijvoorbeeld dit rooster* M *:*

		2	2	
		3		

bevat 3 emmertjes zand in vakje $M[2][2]$ *en 2 emmertjes in de vakjes* $M[1][2]$ *en* $M[1][3]$ *(ter herinnering,* $M[i][j]$ *is het vakje op rij* i*, kolom* j*, en de rijen en kolommen zijn genummerd vanaf* 0*).*

We kunnen emmertjes zand toevoegen aan alle vakjes, maar als het aantal emmers in een vakje 4 (of meer) wordt, dan valt het zandkasteel in dat vak uit elkaar : dat vakje wordt dan leeg terwijl alle ernaast liggende vakjes elk een emmer zand bijkrijgen. Met andere woorden, als $M[i][j]$ *gelijk wordt aan* 4 *(of meer) dan zetten we* $M[i][j]$ *terug op* 0 *en voegen we* 1 *toe aan* $M[i-1][j]$*,* $M[i+1][j]$*,* $M[i][j-1]$ *en* $M[i][j+1]$ *als die vakjes bestaan (als niet, dan waait het zand gewoon weg). Bijvoorbeeld, door 1 toe te voegen aan vakje* $M[2][2]$ *hierboven, zitten we aan 4 emmers in dat vakje, en krijgen we :*

		3	2	
	1		1	
		1		

Uiteraard kunnen de emmers zand die in andere vakjes vallen, het aantal emmers in dat vakje ook tot 4 *(of meer) doen stijgen. Dan herhalen we deze procedure, totdat er geen vakje meer is dat* 4 *of meer bevat. Ik garandeer je dat dit altijd kan in een eindig aantal stappen.*

Maak de volgende oefeningen om het systeem beter te begrijpen :

Q6(a) [4 ptn] **Welk rooster bekomen we als we een emmer zand toevoegen aan vakje** $M[1][4]$ **van het volgende rooster :**

			1	**3**
			2	**2**
	1			

Q6(b) [4 ptn] **Welk rooster bekomen we als we een emmer zand toevoegen aan vakje** $M[2][2]$

van het volgende rooster :

			1	
	2	**3**	**3**	
			1	

Vraag 7 – Plooien

Je hebt een blad papier van x mm dikte. Als je dat dubbel plooit krijg je een dikte van $2x$ mm. Als je dat nog eens dubbel ploot wordt de dikte $4x$ mm. . . enzovoort. Het volgende algoritme heeft twee parameters : een waarde voor x en een gewenste dikte g. Het berekent het minimum aantal plooien dat je moet maken om een dikte te bekomen van minstens g.

```
Input  : x > 0, de dikte van het blad papier in mm.
         g > 0, de minimale gewenste dikte in mm.
Output : het aantal keer dat je het blad minimaal moet plooien om minstens
         g mm dikte te bekomen.

n ← 0

while (x < g)
{
   [...] // Missende instructies
}

return n
```

Q7(a) [5 ptn]	Wat zijn de missende instructies ?
☐	$x \leftarrow n^2$; dan $n \leftarrow n + 1$
☐	$n \leftarrow n + 1$; dan $x \leftarrow n \times 2$
☐	$x \leftarrow x \times 2$; dan $n \leftarrow n + 1$
☐	$x \leftarrow x + n$; dan $n \leftarrow n + 1$

Vraag 8 – De langste strikt stijgende reeks

Het algoritme hieronder berekent, gegeven een array arr van n gehele getallen, de lengte van het langste aaneengesloten deel van de array waarvan de inhoud een strikt stijgende reeks getallen is. Bijvoorbeeld, in de volgende array :

1	4	2	5	7	2	2

zijn er 10 deelreeksen die strikt stijgend zijn. Geklasseerd volgens lengte zijn ze :

— $[]$;
— $[1]$, $[4]$, $[2]$, $[5]$, $[7]$ (Merk op dat de reeks $[2]$ drie keer voorkomt) ;
— $[1, 4]$, $[2, 5]$, $[5, 7]$;
— $[2, 5, 7]$.

Voor dit voorbeeld gaat het algoritme dus de waarde 3 teruggeven (de lengte van $[2, 5, 7]$). Vervolledig het onderstaande algoritme, zodat het correcte resultaat wordt teruggegeven, ongeacht de lengte van de array arr.

Opmerking : De functie `max(a,b)` berekent het maximum van a en b, d.w.z. geeft de grootste van de twee waarden terug.

```
Input  : arr, een array van gehele getallen.
         n, een positief geheel getal (n ≥ 0), de lengte van de array arr.
Output : Geeft de lengte van de langste strikt stijgende deelreeks in arr terug.
         De array arr werd niet gewijzigd.

length ← [...]                                                    // (a)
m ← [...]                                                         // (b)

for (i ← [...] to [...] step 1)                                   // (c,d)
{
    if ([...])                                                    // (e)
    {
        length ← 1
    }
    else
    {
        length ← length + 1
    }
    m ← max (m, length)
}
return m
```

Q8(a) [2 ptn]	**Wat is expressie** `(a)` **?**

Q8(b) [2 ptn]	**Wat is expressie** `(b)` **?**

Q8(c) [2 ptn]	**Wat is expressie** `(c)` **?**

Q8(d) [2 ptn]	**Wat is expressie** `(d)` **?**

Q8(e) [4 ptn]	**Wat is expressie** `(e)` **?**

Vraag 9 – Populatiegroei

Bart onderzoekt de groei van populaties. Hij telt het aantal personen die ergens wonen op verschillende tijdstippen, en wil die gegevens gebruiken om de toekomstige populatie te voorspellen. Bart rekent op computerwetenschapper Lisa om een programma te schrijven dat de grootte van een populatie in de toekomst kan berekenen (op basis van de gemaakte metingen). Bart geeft zijn gegevens aan Lisa. Lisa leidt daaruit af hoe de populatie evolueert in de tijd. Daarna schrijft ze een programma dat deze waarden kan berekenen.

De programma's van Lisa zijn net aangekomen en Bart wil zeker zijn dat ze het juiste resultaat geven. Hij controleert of de programma's op zijn minst de juiste waarden berekenen voor de metingen die hij al gedaan heeft.

Elk programma is een functie met één parameter t. Deze parameter is een geheel getal die het tijdstip voorstelt waarop we de populatie willen kennen. (Voor $t = 0$ berekent het algoritme de beginpopulatie.) De functie geeft dus de grootte van de populatie terug op tijdstip t. Dit is het eerste algoritme :

```
function populatie1(t)
{
    if (t = 0)
    {
        return 0
    }
    else
    {
        return populatie1(t − 1) + 3
    }
}
```

Q9(a) [3 ptn]	**Wat zijn de uitkomsten die berekend worden door algoritme** `populatie1` **voor de waarden** $t = 0$**,** $t = 1$**, ... tot en met** $t = 5$**?**

```
function populatie2(t)
{
    if (t = 0)
    {
        return 1
    }
    else
    {
        return populatie2(t − 1) × 3
    }
}
```

Q9(b) [3 ptn]	**Wat zijn de uitkomsten die berekend worden door algoritme** `populatie2` **voor de waarden** $t = 0$**,** $t = 1$**, ... tot en met** $t = 5$**?**

Vraag 10– Orde in de chaos

Het algoritme "Quick Sort" sorteert een array in stijgende volgorde en werkt als volgt. Eerst kies je een element uit de array (meestal het laatste) dat we *pivot* noemen. Daarna reorganiseren we de array in 3 delen : in een bepaald vakje j (dat we moeten vinden) komt de *pivot* terecht ; in de posities vanaf het begin tot en met $j-1$ zetten we alle waarden die kleiner zijn dan de *pivot* ; en in de posities vanaf $j+1$ tot het einde zetten we alle waarden die groter zijn dan de *pivot*. Als deze opdeling is gemaakt, herhalen we ditzelfde algoritme op elk van de delen van de array links en rechts van j, totdat we het algoritme op slechts 1 element zouden moeten uitvoeren.

Dat ziet eruit als volgt :

```
Input  : arr, een array van gehele getallen
         begin en eind, indices van de array, zodat begin ≤ eind.
Output : De elementen in de array tussen de posities begin en eind (beide inbegrepen)
           zijn gesorteerd in oplopende volgorde.
         Het algoritme geeft geen waarde terug.

function quicksort (arr, begin, eind)
{
    if (eind − begin > 0)
    {
        pivotindex ← partition (arr, begin, eind)
        quicksort (arr, begin, pivotindex − 1)
        quicksort (arr, pivotindex + 1, eind)
    }
}
```

Om de array in 3 delen te reorganiseren en de waarde van j te bepalen, gebruikt dit algoritme de functie `partition` hieronder. Die gebruikt dan weer de functie `swap(arr,i,j)` waarmee je de waarden op indices i en j van de array arr met elkaar verwisselt.

```
Input  : arr, een array van getallen.
         begin en eind, indices van de array zodat begin ≤ eind.
Output : Deze functie reorganiseert de array tussen indices begin en eind in 3 delen
           zoals hierboven beschreven. De rest van arr wordt niet gewijzigd.
         De index j van de pivot wordt teruggegeven.

function partition (arr, begin, eind)
{
    j ← begin
    for (i ← begin to eind step 1)
    {
        if (arr[i] < arr[eind])          // gebruik het laatste element als pivot
        {
            swap (arr, i, [...])                                          // (a)
            [...]                                                         // (b)
        }
    }
    [...]                                                                 // (c)

    return j
}
```

Vervolledig dit algoritme :

Q10(a) [4 ptn]	**Wat is expressie** `(a)` **?**

Q10(b) [4 ptn]	**Wat is instructie** `(b)` **? Je mag de functie** `swap` **gebruiken als je dat nodig vindt.**

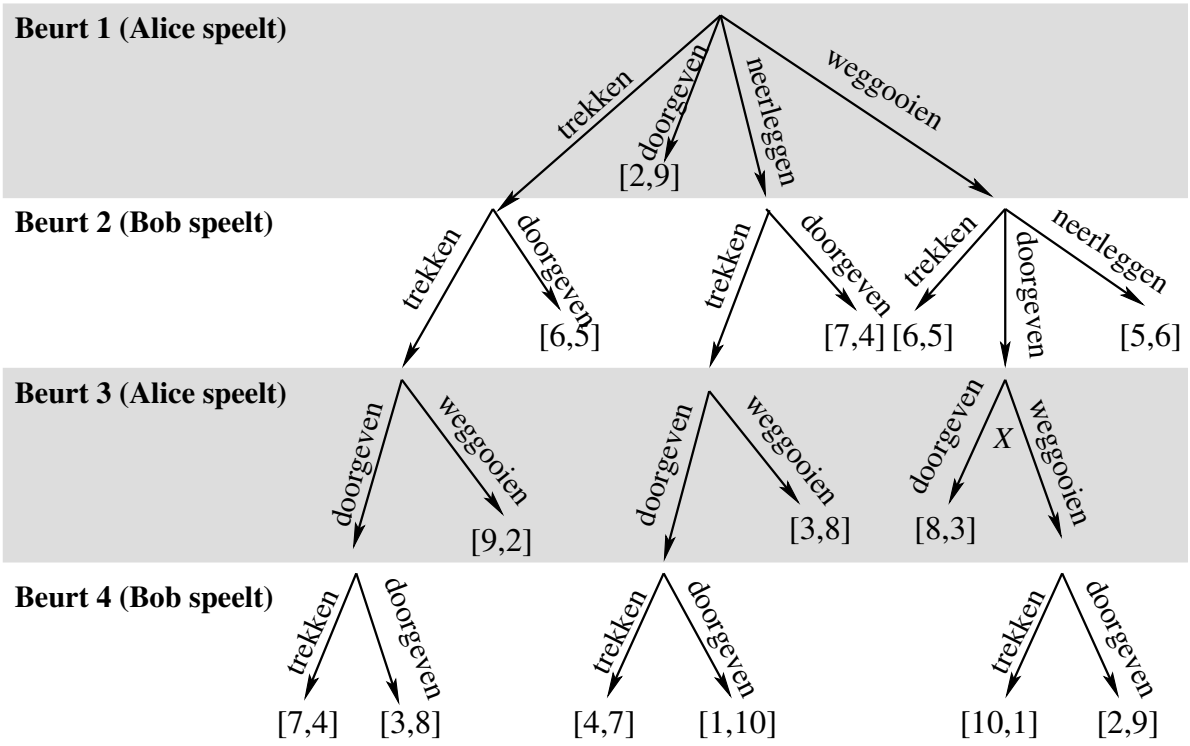
Q10(c) [4 ptn]	**Wat is instructie** `(c)` **? Je mag de functie** `swap` **gebruiken als je dat nodig vindt.**

Vraag 11 – Strategisch Spel

Alice en Bob spelen een complex strategisch spel waarbij ze een elk een sterkere beschaving proberen op te bouwen dan hun tegenstander. Een beschaving wordt opgebouwd via een reeks door de spelers gekozen kaarten.

Op het einde van het spel wordt de score van elke speler bepaald door 11 overwinningspunten te verdelen onder de spelers, aan de hand van verschillende criteria (2 punten voor militaire overmacht, 3 punten voor technologische ontwikkeling, enz.) De speler met de hoogste score wint. Een spannend spel kan bijvoorbeeld nipt gewonnen worden door Alice met 6 punten, tegenover 5 punten voor Bob (deze uitkomst van het spel wordt voorgesteld als $[6, 5]$).

Alice wordt uitgeloot om het spel te beginnen. Ze wil niets aan het toeval overlaten en tekent een diagram met alle mogelijke uitkomsten van het spel (zie hieronder). In haar eerste beurt heeft Alice vier mogelijke acties (een kaart *trekken*, *doorgeven*, *neerleggen* of *weggooien*). Vervolgens is het de beurt aan Bob, enzovoort, tot het einde van het spel, dat hier wordt voorgesteld door de eindscore. Het aantal mogelijke acties is afhankelijk van de spelsituatie. Merk ook op dat een spel sneller kan eindigen, bijvoorbeeld in verlies voor Alice als zij in haar eerste beurt een kaart doorgeeft.



Q11(a) [3 ptn]	**Als Alice weet dat de strategie van Bob erin bestaat om altijd *door te geven*, met welke actie moet Alice dan beginnen om haar score te maximaliseren, en wat zal die score zijn?**

Vul hier je antwoorden in!

Q1(a)	**Eén of meerdere letters**	/2
. . . . .		
Q1(b)	**Eén of meerdere letters**	/2
. . . . .		
Q1(c)	**Eén of meerdere letters**	/2
. . . . .		
Q1(d)	**Eén of meerdere letters**	/2
. . . . .		
Q2(a)	**Een geheel getal**	/2
. . . . .		
Q2(b)	**Een geheel getal**	/2
. . . . .		
Q2(c)	**Een geheel getal**	/3
. . . . .		

Q3(a)		**Er is maar EEN juist antwoord!**	/2
	☐	het woord begint met A en staat op de eerste pagina.	
	☐	het woord staat in het midden van het woordenboek.	
	☐	het woord begint met Z en staat op de laatste pagina.	
	☐	het woord staat niet in het woordenboek.	

Q3(b)		**Er is maar EEN juist antwoord!**	/2
	☐	het woord begint met A en staat op de eerste pagina.	
	☐	het woord staat in het midden van het woordenboek.	
	☐	het woord begint met Z en staat op de laatste pagina.	
	☐	het woord staat niet in het woordenboek.	

Q3(c)		**Er is maar EEN juist antwoord!**	/2
	☐	het woord begint met A en staat op de eerste pagina.	
	☐	het woord begint met A en staat op de laatste pagina.	
	☐	het woord staat in het midden van het woordenboek.	
	☐	het woord staat niet in het woordenboek.	

Q3(d)		**Er is maar EEN juist antwoord!**	/2
	☐	Wanneer het woord in de eerste helft van het woordenboek staat.	
	☐	Wanneer het woord niet in het woordenboek staat.	
	☐	Wanneer het woord in de tweede helft van het woordenboek staat.	
	☐	Wanneer het woord begint met A.	

Q4(a) **Een getal**	/2
. . . . .	

Q4(b) **Een getal**	/2
. . . . .	

Q4(c) **Twee getallen**	/2
. . . . .	

Q4(d) **Een getal of 'X'**	/4
. . . . .	

Q5(a) **Een aantal minuten**	/2
. . . . .	

Q5(b) **Een aantal minuten**	/2
. . . . .	

Q5(c) **De naam van een serie**	/2
. . . . .	

Q6(a) **Teken het resulterende rooster**	/4

Q6(b) **Teken het resulterende rooster**	/4

Q7(a)		**Er is maar EEN juist antwoord!**	/5
	☐	$x \leftarrow n^2$; dan $n \leftarrow n + 1$	
	☐	$n \leftarrow n + 1$; dan $x \leftarrow n \times 2$	
	☐	$x \leftarrow x \times 2$; dan $n \leftarrow n + 1$	
	☐	$x \leftarrow x + n$; dan $n \leftarrow n + 1$	

Q8(a) **Een expressie**	/2
Q8(b) **Een expressie**	/2
Q8(c) **Een expressie**	/2
Q8(d) **Een expressie**	/2
Q8(e) **Een expressie**	/4
Q9(a) **Een reeks van 6 getallen**	/3
Q9(b) **Een reeks van 6 getallen**	/3
Q10(a) **Een expressie**	/4
Q10(b) **Een instructie**	/4
Q10(c) **Een instructie**	/4
Q11(a) **Een actie en een getal**	/3