be-OI 2019 **Finale - BELOFTEN** zaterdag 30 maart 2019	**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 2019. Ze bevat 5 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!

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

Overzicht pseudo-code

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

```
geboortejaar ← 2002
leeftijd ← 2019 − geboortejaar
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Vraag 1 – Pixels-Inleiding

Een scherm bestaat uit pixels. Een Full HD-scherm, bijvoorbeeld, bestaat uit 1080 regels van elk 1920 pixels. Sommige pixels kunnen defect zijn. Als er teveel defecte pixels zijn, of als de verdeling van de defecte pixels over het scherm ongunstig is, moet het scherm verworpen worden.

Een toestel om schermen te controleren produceert een tabel $pix[\,][\,]$ van $r \times c$ niet-negatieve gehele getallen, waarbij r het aantal regels en c het aantal kolommen is van het gecontroleerde scherm.

De waarde van $pix[i][j]$ stelt de toestand voor van de pixel op regel i en kolom j. De waarde is hoger naarmate de pixel beter werkt. Om na te gaan of een pixel defect is, vergelijkt men $pix[i][j]$ met een waarde Q die afhangt van het soort scherm en de kwaliteitsvereisten. Men zegt dat de pixel goed werkt als $pix[i][j] \geq Q$ (met andere woorden, de pixel wordt als defect beschouwd als $pix[i][j] < Q$).

Zie hier een (klein) voorbeeld dat de resultaten voorstelt van een controle van een scherm van 6 regels en 12 kolommen.

	0	1	2	3	4	5	6	7	8	9	10	11
0	141	101	40	207	129	157	247	244	235	229	108	159
1	175	223	222	254	196	162	152	127	235	243	161	183
2	232	123	226	72	245	109	99	87	254	115	142	0
3	172	123	130	178	155	100	198	226	157	110	202	183
4	112	198	212	241	233	191	218	68	76	72	248	162
5	127	137	110	219	196	194	111	213	141	111	249	147

pix[ ][ ]

Als we de waarde 100 nemen voor de kwaliteit Q, moeten we de 8 grijs gekleurde pixels als defect beschouwen.
De pixel op regel 0 en kolom 2 is defect want $pix[0][2] = 40$ is kleiner dan $Q = 100$.
Met deze waarde van Q bevatten 3 regels en 7 kolommen defecte pixels, en zijn er 3 opeenvolgende defecte pixels op regel 4, maar is regel 2 de regel met de meeste defecte pixels want die bevat er 4.

Vervolledig de volgende algoritmes door de `...` in te vullen om de gevraagde resultaten te berekenen. In elk algoritme mag je de tabel $pix[\,][\,]$ gebruiken die de resultaten bevat van het schermcontroletoestel, de variabelen r en c die het aantal regels en kolommen van het scherm bevatten, en de variabele Q die de drempelwaarde voor de kwaliteit bevat.

Algoritme Pixels : Verwerp een scherm als het aantal defecte pixels een gegeven limiet overschrijdt.

```
Input  : pix[][], r, c en Q.
                dmax, het maximale aanvaardbare aantal defecte pixels.
Output : "XX" als het aantal defecte pixels groter is dan dmax,
                "OK" als het aantal defecte pixels kleiner is dan dmax
                   of gelijk is aan dmax.

d ← 0
for (i ← 0 to r-1 step 1)
{
    for (j ← 0 to c-1 step 1)
    {
        if ( ... )                    //(a)
        {
            d ← ...                   //(b)
        }
    }
}


if ( ... )                            //(c)
{
    return "XX"
}
else
{
    return "OK"
}
```

Q1(a) [2 ptn]	**Geef uitdrukking (a) in het algoritme Pixels.**
Oplossing: $pix[i][j] < Q$	

Q1(b) [2 ptn]	**Geef uitdrukking (b) in het algoritme Pixels.**
Oplossing: $d + 1$	

Q1(c) [2 ptn]	**Geef uitdrukking (c) in het algoritme Pixels.**
Oplossing: $d > dmax$ (andere oplossing: $dmax < d$)	

Algoritme Regels : Hoeveel regels bevatten minstens 1 defecte pixel?

```
Input  : pix[][], r, c en Q.
Output : dr, het aantal regels dat minstens één defecte pixel bevat.

dr ← 0
for (i ← 0 to r-1 step 1)
{
    j ← 0
    while (j < c) and (...)      //(d)
    {
        j ← ...                  //(e)
    }

    if ( ... )                   //(f)
    {
        ...                      //(g)
    }
}
return dr
```

Opmerking over de manier waarop de computer een uitdrukking evalueert van de vorm (voorwaarde 1) **and** (voorwaarde 2):
als (voorwaarde 1) **false** is, wordt (voorwaarde 2) niet geëvalueerd; de waarde van de uitdrukking is immers sowieso **false**.

Q1(d) [2 ptn]	**Geef uitdrukking (d) in het algoritme Regels.**
Oplossing: $pix[i][j] \geq Q$	

Q1(e) [2 ptn]	**Geef uitdrukking (e) in het algoritme Regels.**
Oplossing: $j + 1$	

Q1(f) [2 ptn]	**Geef uitdrukking (f) in het algoritme Regels.**
Oplossing: $j < c$ (ook juist: $j \neq c$)	

Q1(g) [2 ptn]	**Geef opdracht (g) in het algoritme Regels.**
Oplossing: $dr \leftarrow dr + 1$ (ook aanvaard: dr++)	

Tel het aantal defecte pixels in een rechthoekig gebied, gegeven de positie van de pixel uiterst links bovenaan (regel $r1$, kolom $c1$) en de pixel uiterst rechts onderaan (regel $r2$, kolom $c2$) in de rechthoek.

Algoritme Rechthoek : Tel het aantal defecte pixels in een rechthoek.

```
Input  : pix[][], r, c en Q.
         r1 en c1
            Het rij- en kolomnummer van de pixel uiterst links bovenaan in de rechthoek.

         r2 en c2
            Het rij- en kolomnummer van de pixel uiterst rechts onderaan in de rechthoek.

         0 ≤ r1 ≤ r2 < r en 0 ≤ c1 ≤ c2 < c
Output : d, het aantal defecte pixels in de gegeven rechthoek.
d ← 0
for (i ← r1 to ... step 1)          //(h)
{
    for (j ← ... to ... step 1)     //(i) et (j)
    {
        if ( ... )                  //(k)
        {
            ...                     //(l)
        }
    }
}
return d
```

Q1(h) [1 pt]	**Geef uitdrukking (h) in algoritme Rechthoek.**
Oplossing: $r2$	

Q1(i) [1 pt]	**Geef uitdrukking (i) in algoritme Rechthoek.**
Oplossing: $c1$	

Q1(j) [1 pt]	**Geef uitdrukking (j) in algoritme Rechthoek.**
Oplossing: $c2$	

Q1(k) [1 pt]	**Geef uitdrukking (k) in algoritme Rechthoek.**
Oplossing: $pix[i][j] < Q$	

Q1(l) [1 pt]	**Geef opdracht (l) in algoritme Rechthoek.**
Oplossing: $d \leftarrow d + 1$	

Q1(m) [1 pt]	**Hoeveel keer zal uitdrukking (k) geëvalueerd worden als $r1 = 500$, $c1 = 480$, $r2 = 599$, $c2 = 511$?**
Oplossing: 3200 (meer bepaald: $100 \times 32 = 3200$)	

Vraag 2 – Pixels-Vervolg

Deze vraag gebruikt dezelfde concepten en variabelen als de vraag "Pixels-Inleiding".
Om de opgave te begrijpen, moet je eerst "Pixels-Inleiding" lezen.
Je antwoorden op deze vraag tellen mee, zelfs als je de vraag "Pixels-Inleiding" niet volledig opgelost hebt.

Soms is het nodig, achtereenvolgens de defecte pixels te tellen in een hele reeks verschillende rechthoeken van verschillende omvang en op verschillende posities op eenzelfde scherm.
In dat geval is het **Algoritme Rechthoek** van de vraag "Pixels-Inleiding" niet efficiënt; een snellere aanpak is mogelijk.
Door gebruik te maken van een nieuwe tabel *pixtot*[ ][ ] kan je razendsnel (zonder enige **`for`**- of **`while`**-lus) het aantal defecte pixels in eender welke rechthoek berekenen.

pixtot[ ][ ] bestaat uit evenveel regels en kolommen als *pix*[ ][ ].
pixtot[*i*][*j*] is het aantal defecte pixels in de rechthoek die strekt van linksboven op het scherm (regel 0, kolom 0) tot de pixel op regel *i* en kolom *j*.

Zie hier enkele voorbeelden gebaseerd op het kleine voorbeeldscherm van de vraag "Pixels-Inleiding".
(De kleuren dienen om je te helpen de verschillende waarden in de tabel te herkennen.)

	0	1	2	3	4	5	6	7	8	9	10	11
0	141	101	40	207	129	157	247	244	235	229	108	159
1	175	223	222	254	196	162	152	127	235	243	161	183
2	232	123	226	72	245	109	99	87	254	115	142	0
3	172	123	130	178	155	100	198	226	157	110	202	183
4	112	198	212	241	233	191	218	68	76	72	248	162
5	127	137	110	219	196	194	111	213	141	111	249	147

pix[ ][ ]

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	1	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1	1	1
2	0	0	1	2	2	2	3	4	4	4	4	5
3	0	0	1	2	2	2	3	4	4	4	4	5
4	0	0	1	2	2	2	3	5	6	7	7	8
5	0	0	1	2	2	2	3	5	6	7	7	8

pixtot[3][6] = 3

Er zijn 3 defecte pixels in de rechthoek die strekt van linksboven op het scherm tot de pixel op regel 3 en kolom 6.

	0	1	2	3	4	5	6	7	8	9	10	11
0	141	101	40	207	129	157	247	244	235	229	108	159
1	175	223	222	254	196	162	152	127	235	243	161	183
2	232	123	226	72	245	109	99	87	254	115	142	0
3	172	123	130	178	155	100	198	226	157	110	202	183
4	112	198	212	241	233	191	218	68	76	72	248	162
5	127	137	110	219	196	194	111	213	141	111	249	147

pix[ ][ ]

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	1	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1	1	1
2	0	0	1	2	2	2	3	4	4	4	4	5
3	0	0	1	2	2	2	3	4	4	4	4	5
4	0	0	1	2	2	2	3	5	6	7	7	8
5	0	0	1	2	2	2	3	5	6	7	7	8

pixtot[2][7] = 4

Er zijn 4 defecte pixels in de rechthoek die strekt van linksboven op het scherm tot de pixel op regel 2 en kolom 7.

Zie hier de *pixtot*[ ][ ]-tabel die overeenkomt met een *ander scherm* van 8 regels en 16 kolommen.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2
2	0	0	0	1	1	1	1	1	1	3	4	5	5	5	5	6
3	0	0	0	1	1	1	1	1	1	3	4	5	5	5	5	6
4	0	0	0	1	1	2	3	3	3	5	6	7	7	7	7	8
5	0	0	0	1	1	2	3	3	3	5	6	7	7	7	7	9
6	0	0	0	1	1	2	3	3	3	6	7	8	8	8	8	10
7	0	0	0	1	1	2	3	3	3	6	7	8	8	8	8	10

pixtot[ ][ ]

Gebruik deze laatste tabel om te antwoorden op de volgende vragen.

Q2(a) [1 pt]	**Hoeveel defecte pixels zijn er op dit scherm?**
Oplossing: 10	

Q2(b) [1 pt]	**Hoeveel defecte pixels zijn er op regel nummer 2?**
Oplossing: 4	

Q2(c) [1 pt]	**Welke regels hebben geen enkele defecte pixel? (Geef de regelnummers, gescheiden door komma's.)**
Oplossing: 3, 7	

Q2(d) [1 pt]	**Hoeveel kolommen hebben geen enkele defecte pixel?**
Oplossing: 9	

Voor de vragen hieronder gebruiken we de volgende **notaties**:

- "pixel $(r1, c1)$" voor de pixel op regel $r1$ en kolom $c1$.
- "rechthoek $(r1, c1) - (r2, c2)$" voor de rechthoek die strekt van regel $r1$ tot regel $r2$ en van kolom $c1$ tot kolom $c2$.

Q2(e) [2 ptn]	**Hoeveel defecte pixels bevinden zich in de rechthoek $(0, 0) - (6, 6)$?**
Oplossing: 3	

Q2(f) [2 ptn]	**Hoeveel defecte pixels bevinden zich in de rechthoek $(3, 3) - (6, 6)$?**
Oplossing: 2	

Q2(g) [2 ptn]	**Hoeveel defecte pixels bevinden zich in de rechthoek $(2, 0) - (2, 12)$?**
Oplossing: 3	

Q2(h) [2 ptn]	**Is pixel** $(4, 9)$ **defect?**
Oplossing: NEE	

Q2(i) [3 ptn]	**Hoeveel defecte pixels bevinden zich in de rechthoek** $(2, 4) - (6, 10)$**?**
Oplossing: 4	

Q2(j) [5 ptn]	**Meer algemeen, gegeven** $1 \leq r1 < r2 < r$ **en** $1 \leq c1 < c2 < c$**, geef een uitdrukking voor het aantal defecte pixels in de rechthoek** $(r1, c1) - (r2, c2)$ **in termen van de elementen van** $pixtot[\,][\,]$**.**
Oplossing: $pixtot[r2][c2] - pixtot[r1-1][c2] - pixtot[r2][c1-1] + pixtot[r1-1][c1-1]$	

Vervolledig het algoritme "pixtot Invullen", dat $pixtot[\,][\,]$ berekent op basis van $pix[\,][\,]$.
Dit algoritme krijgt als invoer de tabel $pix[\,][\,]$ geproduceerd door het schermcontroletoestel (zie vraag "Pixels-Inleiding").
De tabel $pixtot[\,][\,]$ bevat bij aanvang enkel nullen: voor elke i en j geldt $pixtot[i][j] = 0$.
Het algoritme moet $pixtot[\,][\,]$ invullen zodanig dat voor elke i en j, $pixtot[i][j]$ gelijk is aan het aantal defecte pixels in de rechthoek $(0,0) - (i,j)$.

Algoritme pixtot Invullen: Gebruik de tabel $pix[\,][\,]$ om de tabel $pixtot[\,][\,]$ in te vullen.

```
Input  : pix[][], r, c, Q en pixtot[][] geïnitialiseerd met nullen.
Output : pixtot[][] ingevuld zoals hierboven beschreven.

//Linkerbovenhoek.
if (pix[0][0] < Q )
{
    pixtot[0][0]=1
}

//Rest van de eerste regel.
for (j ← 1 to c-1 step 1)
{
    pixtot[0][j] ← pixtot[0][j-1]
    if (pix[0][j] < Q )
    {
        pixtot[0][j] ← ...              //k
    }

//Andere regels.
for (i ← 1 to r-1 step 1)
{
    d = 0 //Telt de defecte pixels op de regel.
    for (j ← 0 to c-1 step 1)
    {
        if (pix[i][j] < Q )
        {
            ...                         //l
        }
        pixtot[i][j] ← ...              //m
    }
}
return pixtot
```

Q2(k) [2 ptn]	**Geef uitdrukking (k) in algoritme pixtot Invullen.**
Oplossing: $pixtot[0][j] + 1$ (ook juist: $pixtot[0][j-1] + 1$)	

Q2(l) [2 ptn]	**Geef uitdrukking (l) in algoritme pixtot Invullen.**
Oplossing: $d \leftarrow d + 1$	

Q2(m) [4 ptn]	**Geef uitdrukking (m) in algoritme pixtot Invullen.**
Oplossing: $pixtot[i-1][j] + d$	

Vraag 3 – Arthur, Merlijn, het monster van het meer en de anderen…

Een draak bedreigt het rijk van Koning Arthur; hij heeft dringend zijn zwaard Excalibur nodig. Zijn trouwe raadgever, Merlijn de tovenaar, zou hem precies kunnen zeggen hoe hij te werk moet gaan om Excalibur te bemachtigen; Merlijn is echter in het buitenland voor een tovenaarsbijeenkomst. Gelukkig vindt Arthur tussen Merlijns aantekeningen wel de volgende informatie:

(a) Bij de kabouters kan je voor drie gouden muntstukken een diamant kopen.

(b) De heks bereidt je een liefdesdrankje voor de prijs van één koperen muntstuk.

(c) Het monster dat in het meer woont, wil een liefdesdrankje en een boeket rozen om een sirene te verleiden. Wie hem dat brengt, krijgt in ruil Excalibur.

(d) Als je bij de kabouterbank twee gouden muntstukken betaalt, krijg je één zilveren muntstuk en twee koperen muntstukken.

(e) De dame van het Meer geeft je Excalibur als je haar een diamant bezorgt.

(f) Je kan op de markt een boeket rozen kopen voor één zilveren muntstuk.

Arthur heeft **vier gouden muntstukken** in zijn bezit.

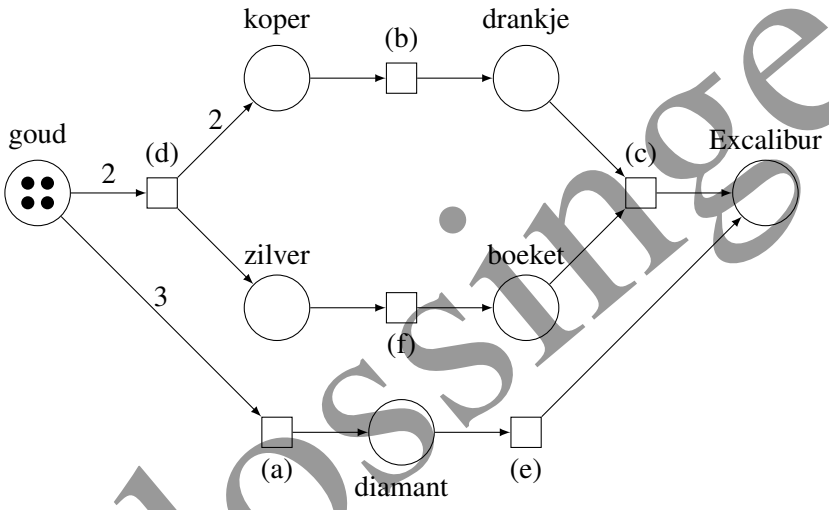
Arthur overweegt hoe hij te werk zou kunnen gaan om Excalibur te bemachtigen, vertrekkend van de vier gouden muntstukken die hij heeft. Hij overweegt verschillende procedés. Bijvoorbeeld: het procedé (a), (b), (c) betekent dat Arthur eerst drie gouden muntstukken gebruikt om bij de kabouters een diamant te kopen. Na die stap heeft hij één gouden muntstuk en één diamant. Daarna probeert hij een liefdesdrankje te kopen bij de heks, maar dat lukt niet want hij heeft op dat moment geen koperen muntstuk… Arthur geeft echter niet op…

Duid voor de volgende procedés aan of ze "Onmogelijk" of "Mogelijk" zijn, door de overeenkomstige vakjes aan te vinken op het antwoordblad. Als een procedé mogelijk is én Excalibur oplevert, vink dan **ook** het vakje "Excalibur" aan.

	Onmogelijk	Mogelijk	Excalibur	Procedés
Q3(a) [1 pt]	☐	☒	☐	(d), (f), (b)
Q3(b) [1 pt]	☒	☐	☐	(d), (b), (c), (f)
Q3(c) [1 pt]	☒	☐	☐	(a), (d), (b), (f), (c)
Q3(d) [1 pt]	☐	☒	☒	(a), (e)
Q3(e) [1 pt]	☒	☐	☐	(d), (a), (e)
Q3(f) [1 pt]	☐	☒	☒	(d), (d), (b), (f), (b), (b), (f), (c)

Q3(g) [2 ptn]	**Geef alle procedés die Excalibur opleveren, in een minimaal aantal stappen, en waarbij Arthur twee gouden muntstukken overhoudt.**
Oplossing: (d), (b), (f), (c) en (d), (f), (b), (c)	

Arthur zegt tegen zichzelf: als de raadgever van de koning er niet is, moet de koning zijn plan trekken.
Hij beslist een tekening te maken in het zand om een beter zicht te krijgen op het probleem.
Voor elk van de *middelen* (muntstukken, diamanten, drankjes, ...) tekent hij een cirkel, en voor elk van de *handelingen* tekent hij een rechthoek. Hij legt steentjes in de cirkels om te tellen over hoeveel middelen hij beschikt. In het begin van een procedé heeft hij vier gouden muntstukken; dat stelt hij voor door vier steentjes te leggen in de cirkel "gouden muntstukken". Hij tekent een pijl van een middel M naar een handeling H als H M nodig heeft, en hij duidt op de pijl de vereiste hoeveelheid aan. Er gaat bijvoorbeeld een pijl met bijschrift 2 van "gouden muntstukken" naar (d).
Op dezelfde manier tekent Arthur een pijl van een handeling H naar een middel M als H M oplevert, en opnieuw duidt hij op de pijl de hoeveelheid aan.
Voor de eenvoud schrijft Arthur niets bij een pijl als de geconsumeerde of geproduceerde hoeveelheid gelijk is aan 1.
Ziehier zijn tekening:



Als Arthur nu bijvoorbeeld de handeling (d) uitvoert, blijven er nog maar twee steentjes over in middel "goud", maar verschijnen er twee in middel "koper" en één in middel "zilver".
Op dezelfde manier moet er, om de handeling (c) uit te kunnen voeren, *minstens* één steentje liggen in middel "drankje" **en** *minstens* één steentje in middel "boeket".
Als we die handeling dan uitvoeren, verwijderen we één steentje van elk van deze twee middelen, en voegen we er eentje toe aan "Excalibur", het middel waar het voor Arthur om gaat. Om een handeling uit te voeren moeten dus de voorwaarden gegeven door *alle* inkomende pijlen van de handeling voldaan zijn; de handeling heeft tot gevolg dat de middelen voorgesteld door *alle* inkomende pijlen geconsumeerd worden, en dat de middelen voorgesteld door *alle* uitgaande pijlen geproduceerd worden.

Aangenomen dat Arthur telkens begint met 4 steentjes in "goud" en geen enkel steentje in de andere middelen, hoeveel steentjes liggen er dan in elk middel nadat Arthur de volgende procedés uitgevoerd heeft?

Q3(h) [6 ptn]	**Na (d), (b), (f)?**
Oplossing: goud: 2 koper: 1 zilver: 0 drankje: 1 boeket: 1 diamant: 0 Excalibur: 0	

Q3(i) [7 ptn]	**Na (d), (b), (b), (d), (b), (f)?**
Oplossing: goud: 0 koper: 1 zilver: 1 drankje: 3 boeket: 1 diamant: 0 Excalibur: 0	

Arthur begint nu aan een nieuwe zoektocht: hij wil de Heilige Graal bemachtigen. Ziehier de relevante aantekeningen van Merlijn:

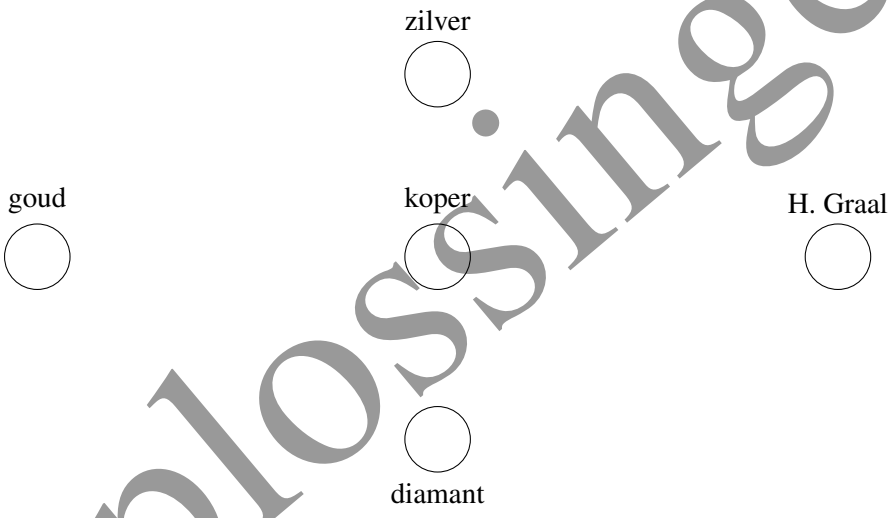
(a) Als je de kabouters één gouden muntstuk geeft, krijg je er twee zilveren muntstukken én één koperen muntstuk voor in de plaats.

(b) Bij de juwelier kan je een diamant kopen voor twee gouden muntstukken.

(c) Merlijns oude vriend Gandalf is dermate bijzichtig en seniel dat hij je voor elk zilveren muntstuk dat je hem geeft, een gouden muntstuk in de plaats geeft.

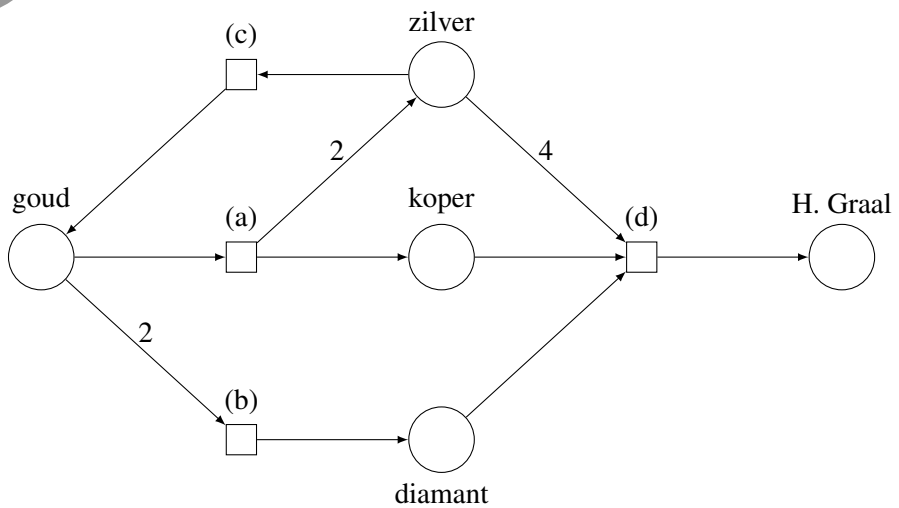
(d) De Amazones hebben een nieuwe webshop geopend waarop je de Heilige Graal kan kopen als je vier zilveren muntstukken, één koperen muntstuk én één diamant kunt neertellen!

Zoals voor de vorige zoektocht, wil Arthur een tekening maken die het probleem in kaart brengt. Hij begint met de cirkels (middelen) te tekenen, maar heeft jouw hulp nodig om de handelingen en de pijlen toe te voegen.



Q3(j) [8 ptn] **Voeg de handelingen (a), (b), (c) en (d) en de pijlen (met de hoeveelheden) toe volgens de aantekeningen van Merlijn (maak eerst een kladversie hierboven en dan een nette versie op het antwoordblad).**

Oplossing:



Vraag 4 – Getallen op het bord

Alice speelt met niet-negatieve gehele getallen op een krijtbord. Aanvankelijk staan er een aantal dergelijke getallen op het bord. Ze kiest er twee uit. Die wist ze, waarna ze hun som of hun absoluut verschil[1] er weer op schrijft.
Nu staat er één getal minder op het bord. Alice doet hetzelfde nu opnieuw en opnieuw, totdat er nog maar één getal op het bord staat.
Alice wil *de kleinst mogelijke waarde voor het laatste overblijvende getal* achterhalen; die noemt ze *het antwoord*.

Een voorbeeld met $1, 4, 6$: Alice zou kunnen beginnen met 1 en 6 op te tellen, zodat 4 en 7 overblijven op het bord. Die vervangt ze dan door hun absoluut verschil $|4 - 7| = 3$. Maar ze kan beter: als ze begint met 1 en 4 op te tellen, blijven op het bord 5 en 6 over. Als ze dan hun absoluut verschil berekent, krijgt ze $|5 - 6| = 1$. Beter kan niet; *het antwoord* dat Alice zoekt is dus 1.

Q4(a) [1 pt]	**Wat is het antwoord voor de lijst 4, 2 ?**
Oplossing: 2	

Q4(b) [1 pt]	**Wat is het antwoord voor de lijst 2, 6, 3 ?**
Oplossing: 1	

Q4(c) [2 ptn]	**Wat is het antwoord voor de lijst 3, 5, 8, 10 ?**
Oplossing: 0	

Q4(d) [2 ptn]	**Wat is het antwoord voor de lijst 14, 51, 29, 52, 15, 30 ?**
Oplossing: 1	

Alice wil het zichzelf nu moeilijker maken.
Ze stelt zich vragen zoals "Bestaat er een lijst van 4 gehele getallen tussen 1 en 10 waarvoor het antwoord 2 is?"
De lijst 1, 1, 3, 7 is er zo één, want $|7 - (3 + (1 + 1))| = 2$ en het is niet mogelijk een kleinere eindwaarde te krijgen.
Opgelet: herinner je dat *het antwoord* de *kleinste mogelijke eindwaarde* is voor de gegeven lijst. Bijgevolg is de lijst 1, 2, 3, 4 niet een lijst die we zoeken, want ook al geldt $|(4 + 2) - (1 + 3)| = 2$, dat is niet *het antwoord* want het is mogelijk een kleinere eindwaarde te krijgen, bv. $|(4 + 1) - (3 + 2)| = 0$.

Bij de vragen hieronder moet je een lijst zoeken waarvoor *het antwoord* gelijk is aan de gegeven waarde. Als je er één vindt, geef dan haar elementen, gescheiden door komma's. Als je overtuigd bent dat een dergelijke lijst niet bestaat, schrijf je "NEE". Een lijst mag hetzelfde getal meerdere keren bevatten.

Q4(e) [1 pt]	**Bestaat er een lijst van 2 gehele getallen tussen 1 en 8 waarvoor het antwoord 3 is?**
Oplossing: 1,4 of 2,5 of 3,6 of 4,7 of 5,8	

Q4(f) [2 ptn]	**Bestaat er een lijst van 3 gehele getallen tussen 1 en 6 waarvoor het antwoord 2 is?**
Oplossing: 1,1,4 of 1,2,5 of 1,3,6 of 2,2,2 of 2,2,6 of 2,3,3 of 2,4,4 of 2,5,5 of 2,6,6 of 3,3,4 of 3,4,5 of 3,5,6 of 4,4,6	

[1] Het absoluut verschil tussen twee getallen x en y is de absolute waarde van hun verschil, genoteerd als $|x - y|$. Je krijgt het als je het kleinste getal aftrekt van het grootste. Bijvoorbeeld: het absoluut verschil tussen 3 en 5 is $|3 - 5| = |5 - 3| = 2$, en het absoluut verschil tussen 7 en 7 is $|7 - 7| = 0$.

Q4(g) [2 ptn]	**Bestaat er een lijst van 4 gehele getallen tussen 1 en 5 waarvoor het antwoord 5 is?**
Oplossing: NEE	

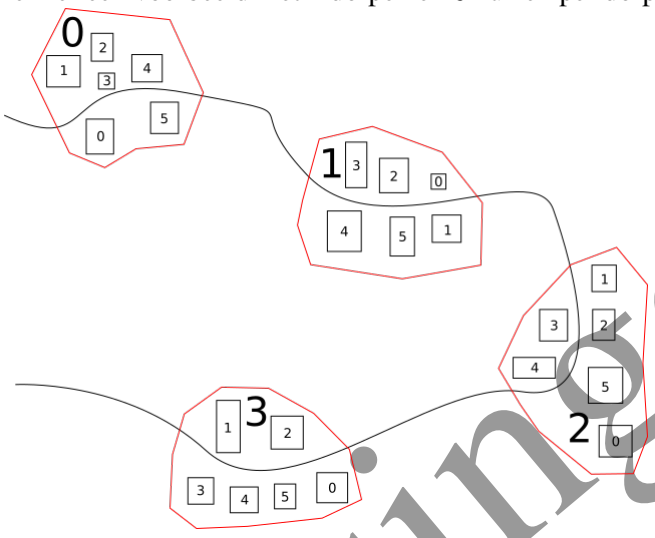
Q4(h) [2 ptn]	**Bestaat er een lijst van 5 gehele getallen tussen 1 en 3 waarvoor het antwoord 3 is?**
Oplossing: 3,3,3,3,3	

Q4(i) [2 ptn]	**Bestaat er een lijst van 6 gehele getallen tussen 1 en 2 waarvoor het antwoord 1 is?**
Oplossing: 1,1,1,1,1,2 of 1,1,1,2,2,2 of 1,2,2,2,2,2	

Vraag 5 – Twee nummeringen

Een bergachtig gebied telt n dorpen. Eén weg verbindt al deze dorpen. De dorpen zijn genummerd van 0 tot $n-1$ en tellen elk precies t huizen. In elk dorp onderscheidt men de huizen aan de hand van een *lokaal nummer* tussen 0 en $t-1$.

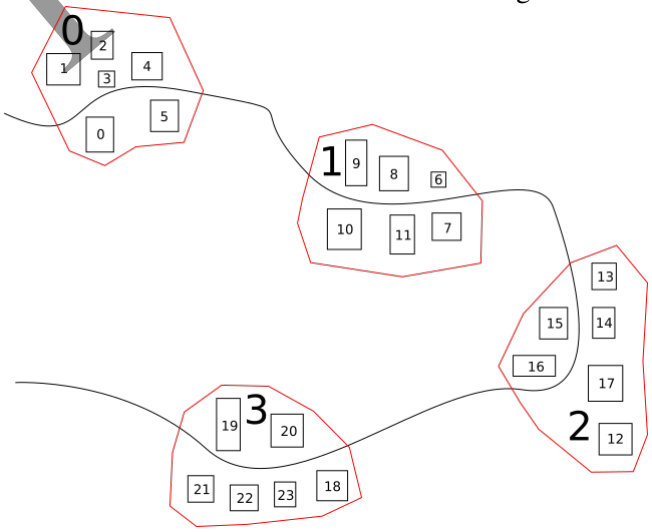
Zie hier een voorbeeld met 4 dorpen en 6 huizen per dorp.



De dorpen zijn getekend in het rood; hun nummer is aangegeven met een groot cijfer.
Elk huis wordt voorgesteld door een kleine rechthoek met daarin het lokaal nummer van het huis.

Elk huis heeft ook een **regionaal nummer** toegekend door de centrale overheid van het gebied. De regionale nummers zijn toegekend in stijgende volgorde vertrekkend van 0, waarbij de dorpen doorlopen worden in de volgorde van hun nummer, en waarbij binnen elk dorp de volgorde van het lokale nummer van de huizen gevolgd wordt. De nummering begint dus met de huizen van dorp 0, nummert vervolgens die van dorp 1, dan die van dorp 2, enzovoort.

De tekening hieronder verduidelijkt de regionale nummering.
Ze geeft aan hoe de centrale overheid de huizen van het voorgaande voorbeeld nummert.



Het **regionale nummer** van elk huis staat vermeld in het overeenkomstige rechthoekje.

Q5(a) [3 ptn]	**Als er** 7 **dorpen zijn met elk** 10 **huizen, wat is dan het regionale nummer van het huis met lokaal nummer** 2 **in dorp** 4 **?**
Oplossing: 42	

Q5(b) [3 ptn]	**Als elk dorp** 13 **huizen telt, wat is dan het regionale nummer van het huis met lokaal nummer** 9 **in dorp** 7 **?**
Oplossing: 100	

Q5(c) [3 ptn]	**Als de dorpen elk** 25 **huizen hebben, wat is dan het nummer van het dorp en het lokale nummer van het huis met regionaal nummer** 83 **?**
Oplossing: Dorp nummer 3 , lokaal nummer 8	

Q5(d) [3 ptn]	**Als het huis met lokaal nummer** 8 **in dorp** 5 **het regionaal nummer** 63 **heeft, wat is dan het aantal huizen in elk dorp?**
Oplossing: 11	

Q5(e) [3 ptn]	**Als er** n **dorpen zijn met elk** t **huizen, geef dan een uitdrukking voor het regionaal nummer van het huis met lokaal nummer** a **in dorp** b**.**
Oplossing: $b \cdot t + a$	

Mijnheer Janssens moet alle huizen van het gebied bezoeken in een heel bijzondere volgorde. Hij moet beginnen met de huizen met lokaal nummer 0 van de verschillende dorpen, in de volgorde van het nummer van het dorp, dan verder gaan met de huizen met lokaal nummer 1 van de verschillende dorpen, opnieuw in de volgorde van het nummer van het dorp, enzovoort totdat hij het huis met het grootste lokale nummer in het dorp met het grootste dorpnummer bezocht heeft.
Een ambtenaar van het gebied moet een lijst voorbereiden met de regionale nummers van deze huizen, in de volgorde waarin ze bezocht moeten worden. Bijvoorbeeld, als er 4 dorpen zijn met elk 6 huizen (zoals op de tekeningen), dan is de lijst 0, 6, 12, 18, 1, 7, 13, 19, 2, 8, 14, 20, 3, 9, 15, 21, 4, 10, 16, 22, 5, 11, 17, 23.
Het algoritme **Bezoeken** stelt zo'n lijst op, gegeven het aantal dorpen en het aantal huizen per dorp. Het vult een tabel $V[\,]$ (die evenveel elementen bevat als er huizen zijn in het gebied) zodanig dat voor elke index i geldt dat $V[i]$ het regionaal nummer is dat Mijnheer Janssens bij het i^{de} bezoek moet bezoeken (waarbij we zoals gewoonlijk de bezoeken nummeren vanaf 0).

Algoritme Bezoeken :

```
Input  : n, het aantal dorpen.
         t, het aantal huizen per dorp.
         V[ ], de tabel, geïnitialiseerd met 0.
Output : V[ ], gevuld met de regionale nummers, in de volgorde van de bezoeken.

k ← 0
for (i ← 0 to t−1 step 1)
{
    for (j ← 0 to n−1 step 1)
    {
        V[...] ← ...        //(f) (g)
        k ← k+1
```

```
    }
}
return V
```

Q5(f) [2 ptn]	**Geef uitdrukking (f) in algoritme Bezoeken.**
Oplossing: k	

Q5(g) [3 ptn]	**Geef uitdrukking (g) in algoritme Bezoeken.**
Oplossing: $j \times t + i$	