OI 2010 **Halve finale** 24 maart 2010	**Vul deze gegevens in, in HOOFDLETTERS en LEESBAAR aub** VOORNAAM : ..... NAAM : ..... SCHOOL :	**Gereserveerd**

Belgische Olympiades in de Informatica (duur : maximum 3u)

Dit is de vragenlijst van de halve finale van de Belgische Olympiades in de Informatica voor de categorie secundair onderwijs. Ze bevat 6 vragen. De eerste twee vragen zijn meerkeuzevragen, de vier overige vragen zijn open vragen. Naast elke vraag vind je een indicatie van de tijd die het mogelijk kost om de vraag op te lossen. Dit is slechts een schatting.

Algemene Opmerkingen (lees dit aandachtig voordat je begint met het beantwoorden van de vragen)

1. Vul je naam, voornaam en school **enkel in op het eerste blad**. Op alle andere bladzijden mag je alleen schrijven in de **kaders voorzien voor je antwoord**.

2. Je mag alleen iets bij je hebben om te schrijven. Rekenmachines, GSM, ... zijn **verboden**.

3. Schrijf je antwoorden met een **(bal)pen**. Laat geen antwoorden staan in potlood. Als je kladbladen wenst, vraag ze dan aan een toezichthouder.

4. Bij de meerkeuzevragen mag je slechts **één enkel antwoord** kiezen. Een correct antwoord levert je 1 punt op, geen antwoord 0 punten, en een fout antwoord wordt bestraft met $-0{,}5$ punt.

5. Op de open vragen mag je antwoorden in pseudo-code, met behulp van een diagram of in een van de toegelaten programmeertalen: Java, C, C++, C#, Pascal, Python, Ruby, PHP en Visual Basic. In het laatste geval, mag je **geen enkele externe bibliotheek gebruiken**, enkel standaardconstructies uit de taal zijn toegestaan. Syntaxfouten worden niet bestraft. In geval van twijfel, doe beroep op een toezichthouder.

6. Je mag hulpfuncties definiëren en recursie gebruiken, behalve als in de vraag expliciet wordt vermeld dat het verboden is.

7. De notatie **`for`** $(i \leftarrow a$ **`to`** b **`step`** $k)$ staat voor een lus die zich herhaalt zolang $i \leq b$, waarbij i vertrekt vanaf waarde a en verandert met stapgrootte k.

8. Je hebt **exact 3 uur** de tijd om alle vragen te beantwoorden.

Succes !

Vraag 1 – Plooien (10 min)

Gegeven een blad papier van x mm dikte. Als ik dit blad dubbel vouw, krijg ik een dikte van $2x$ mm. Als ik dat opnieuw doe, is de dikte $4x$ mm … enzovoort. Het onderstaande algoritme aanvaardt twee parameters: de waarde van x en een gewenste dikte g. Het berekent het minimum aantal keer dat je het blad moet dubbelplooien om tenminste g mm dikte te bekomen.

```
Input  : x > 0, dikte van het papier in mm%épaisseur du papier en mm
         g > 0, minimaal gewenste dikte in mm%épaisseur minimale souhaitée en mm
Output : het benodigde aantal keer dubbelvouwen, om tenminste g mm dikte te bekomen %nombre d

n ← 0
while (x < g)
{
    [...]
}
return n
```

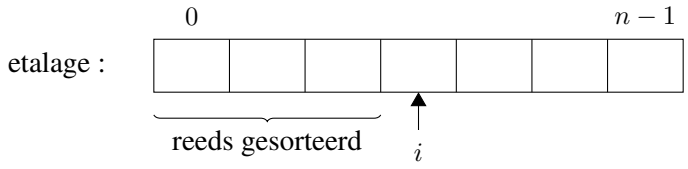
Welke instructies ontbreken om dit algoritme te laten doen wat het moet doen?

☐	$x \leftarrow n^2$ $n \leftarrow n + 1$
☐	$x \leftarrow 2 \cdot x$ $n \leftarrow n + 1$
☐	$n \leftarrow n + 1$ $x \leftarrow 2 \cdot n$
☐	$x \leftarrow x + n$ $n \leftarrow n + 1$

Vraag 2 – Een beetje opruimen (15 min)

Je neemt werk aan als jobstudent in een lokale supermarkt, en je wordt verantwoordelijk voor het departement dat schoenen verkoopt. Je baas vraagt je om de schoenen te ordenen volgens hun schoenmaat, in oplopende volgorde. Je hebt n paar schoenen ter beschikking, en elk daarvan bevindt zich al in de etalage op een zekere plaats. We nummeren de n plaatsen in de etalage van 0 tot en met $n-1$. In het onderstaande algoritme is $etalage[pos]$ de maat van het paar schoenen dat zich bevindt op positie pos in de etalage. Het algoritme sorteert de schoenen in oplopende volgorde.

Het algoritme gaat de etalage overlopen van links naar rechts. Op elke positie i neem je het paar schoenen dat zich daar bevindt. De eerste lus(**while**) laat toe om de plaats te vinden (aan je linkerkant) waar het paar schoenen dat je zonet hebt genomen moet terechtkomen (pos). De tweede lus (**for**) gaat alle schoenen indien nodig een plaats opschuiven, om het paar schoenen in je hand daar te kunnen plaatsen.




```
Input  : etalage, de schoenmaten van de schoenparen
         n > 0, het aantal paar schoenen
Output : etalage werd gesorteerd in oplopende volgorde

for (i ← 1 to n − 1 step 1)
{
    pos ← 0
    while (etalage[pos] < etalage[i])
    {
        pos ← pos + 1
    }
    in_de_hand ← etalage[i]

    for ([...])
    {
        etalage[j + 1] = etalage[j]
    }
    etalage[pos] = in_de_hand
}
```

Welke waarden van j moet men doorlopen om de tweede fase correct te doen verlopen?

☐	$j \leftarrow i$ **to** pos **step** -1
☐	$j \leftarrow i+1$ **to** pos **step** -1
☐	$j \leftarrow i-1$ **to** pos **step** -1
☐	$j \leftarrow i$ **to** 1 **step** -1

Vraag 3 – Kralensnoer (20 min)

Je zus speelt met kralen, zwarte en witte, en ze wilt halskettingen maken. In een vlaag van esthetische inspiratie, wilt ze de kleuren afwisselen, maar toch een zekere periodiciteit behouden. Ze plaatst dus eerst x witte kralen, gevolgd door y zwarte kralen, en ze herhaalt dit n keer.

Stel bijvoorbeeld dat ze $x = 12$ witte kralen en $y = 6$ kralen bezit. Dan zijn er meerdere mogelijkheden:

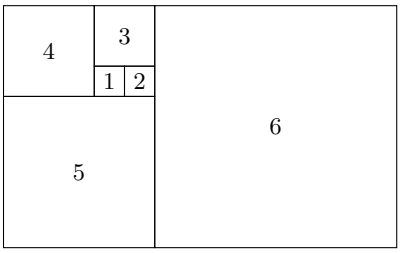
- één herhaling (12 noires, 6 blanches) ○○○○○○○○○○○○●●●●●●
- twee herhalingen (6 noires, 3 blanches) ○○○○○○●●●○○○○○○●●●
- drie herhalingen (4 noires, 2 blanches) ○○○○●●○○○○●●○○○○●●
- zes herhalingen (2 noires, 1 blanche) ○○●○○●○○●○○●○○●○○●

Gegeven twee strikt positieve gehele getallen x et y die het aantal witte respectievelijk zwarte parels voorstellen. Schrijf een algoritme dat het maximum aantal herhalingen berekent dat je zus met deze kralen kan bekomen.

Q3

Vraag 4 – Rechthoeken van vierkanten (20 min)

Je kleine broer heeft vierkante houten puzzelstukken gekregen, waarvan je de lengte van de zijden niet kent. Hij gaat een speciaal Grieks motief construeren met zijn puzzelstukken. Eerst vindt hij de allerkleinste stukken, en plaatst twee ervan naast elkaar. Zo vormt hij een rechthoek met breedte 1 en lengte 2 (vierkanten 1 en 2). Dan vindt hij een stuk waarvan de zijde even lang is als de lengte van de eerder verkregen rechthoek (vierkant 3), en hij plaatst dat vierkant naast de rechthoek om zo een nieuwe rechthoek te vormen. Hij herhaalt dit proces meerdere keren. De lengte van de verkregen rechthoeken zijn respectievelijk $3, 5, 8, 13, \ldots$



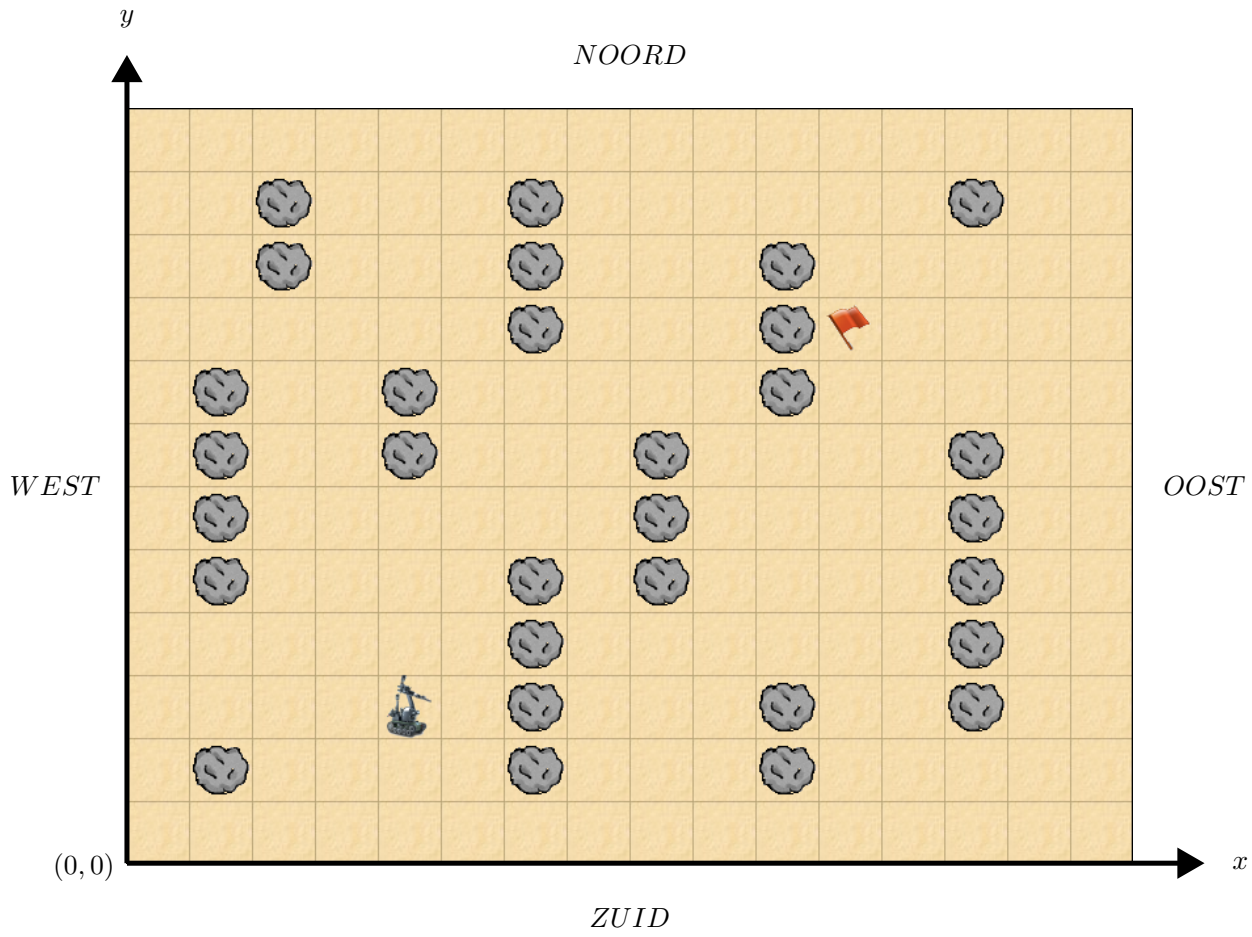
We kunnen een wiskundige functie L definiëren die de lengte van de de rechthoek berekent nadat n vierkante puzzelstukken op deze manier werden gelegd. Schrijf een algoritme dat de waarden van de lengtes van de rechthoeken uitprint, voor de gehele getallen $i = 1, 2, 3, \cdots$ tot een gegeven geheel getal n, waarbij $n \geq 1$.

$$L(n) = \begin{cases} 0 & \text{, als } n = 0 \\ 1 & \text{, als } n = 1 \\ L(n-1) + L(n-2) & \text{, als } n > 1 \end{cases}$$

Q4 **! Je mag hier geen hulpfuncties gebruiken, noch recursie**

Vraag 5 – Waarlangs ? (50 min)

De ESA (European Space Agency) huurt je in om de toekomstige maanexploratierobot te programmeren. Je moet een algoritme schrijven dat de robot toelaat zich te verplaatsen totdat het een zeker doel bereikt (de vlag). Het terrein is onderverdeeld in vierkante vakken en de robot is in de startpositie oostwaarts georiënteerd. Het terrein is bezaaid met onbeweegbare obstakels, dit zijn steeds verticale rotsformaties. Het is steeds mogelijk om rondom de rotsen heen te gaan. We veronderstellen ook dat de robot zich altijd ten westen van het doel bevindt. Het terrein hieronder is slechts een voorbeeld. Je robot moet kunnen functioneren in alle omstandigheden die aan de bovenvermelde voorwaarden voldoen. De grootte van het terrein kan variëren, het aantal obstakels en de positie ervan ook.



Je robot kan slechts een beperkt aantal instructies begrijpen, en je mag enkel deze instructies gebruiken in je algoritme. Sommige van deze instructies doen de robot bewegen, andere instructies laten toe om de status van de robot op te vragen of om informatie te bekomen over de omgeving.

`move`	Doet de robot bewegen in zijn huidige richting, als hij tenminste niet tegen de rand van het terrein of tegen een rots aanloopt. In die laatste gevallen, doet hij niets.
`turnLeft`	De robot draait 90 graden naar links.
`turnRight`	De robot draait 90 graden naar rechts
`canMove`	Test of de robot zich voorwaarts kan bewegen. (d.w.z. niet voor een rots of de rand staat).
`getX, getY`	Vraag de coördinaten x of y op van de robot.
`getDirection`	Vraag de huidige bewegingsrichting van de robot op (`NOORD`, `ZUID`, `WEST` of `OOST`).
`targetX, targetY`	Vraag de coördinaten x of y op van het doelwit.
`isOnTarget`	Test of de robot zich op het doelwit bevindt.

Hieronder staat een voorbeeld van een mogelijk algoritme. We zien hier dat de robot zoveel mogelijk tracht te bewegen, zolang hij het doel niet heeft bereikt. Eens dat niet meer mogelijk is, draait hij 90 graden naar links.

```
while (not isOnTarget)
{
    while (canMove)
    {
        move
    }

    turnLeft
}
```

Ontwikkel een algoritme dat de robot toelaat om het doel te bereiken in zo weinig mogelijk verplaatsingen (zo weinig mogelijk `move` bewegingen). Er zijn meerdere oplossingen, stel diegene voor die je het beste lijkt. Je mag meerdere oplossingen voorstellen - geef ons bijvoorbeeld eerst de gekste die enorm veel tijd vraagt, en nadien een intelligentere.

Q5 **! Je mag enkel de gegeven functies gebruiken**

Q5 (vervolg)

Vraag 6 – Super Programmer Star (50 min)

Een commerciële tv-zender wilt een nieuw format lanceren. De deelnemers moeten overleven op een verlaten eiland en in team een computerspel programmeren. Ze moeten overeenkomen op het vlak van het programmeren zelf, maar ook over het design, de scenario's en de personages. Elke week worden meerdere teams geëlimineerd door televoting.

Elk team heeft een nummer, een natuurlijk getal $n > 0$. Elke kijker kan stemmen voor meerdere teams, en de lijst $stemmen$ bevat alle geregistreede stemmen. Elk team dat geen aantal stemmen heeft gekregen dat groter of gelijk is aan het gemiddelde van de ontvangen stemmen voor elk team, ligt eruit.

Schrijf een algoritme dat een aantal teams N en een lijst $stemmen$ aanvaardt als parameters. Het algoritme geeft een nieuwe lijst terug, van dezelfde lengte als $stemmen$, waarvan de eerste elementen de nummers bevatten van de teams die afvallen en de andere elementen -1. Het aantal stemmen ontvangen door team i duiden we aan met $stemmen[i]$.

Bijvoorbeeld, stel dat er 10 teams deelnemen, waarvan de nummers $1, 2, \cdots, 10$ zijn. Het productiehuis heeft 15 stemmen ontvangen, die staan in de lijst $votes = [3, 7, 3, 3, 9, 9, 5, 1, 2, 5, 7, 1, 9, 5, 2]$. We kunnen het aantal ontvangen stemmen voor elk team berekenen:

Team	1	2	3	4	5	6	7	8	9	10
AantalStemmen	2	2	3	0	3	0	2	0	3	0

We berekenen het gemiddeld aantal stemmen door het totaal aantal ontvangen stemmen (15) te delen door het aantal teams dat tenminste 1 stem ontving (6). We negeren dus de teams die geen enkele stem hebben gehaald! Dit geeft:

$$m = \frac{15}{6} = 2{,}5$$

Alle teams die niet op zijn minst $m = 2{,}5$ stemmen hebben ontvangen, zijn uitgeschakeld. De lijst die je algoritme dus moet produceren is :

$result = [1, 2, 4, 6, 7, 8, 10, -1, -1, -1, -1, -1, -1, -1, -1]$

Q6

Q6 (vervolg)