OI 2010 **Finale** 12 Mei 2010	**Gegevens invullen in HOOFDLETTERS en LEESBAAR, aub** VOORNAAM : ..... NAAM : ..... SCHOOL :	**Gereserveerd**

Belgische Olympiades in de Informatica (duur : maximum 1u15')

Dit is de vragenlijst voor het **gedeelte op papier** van de finale van de Belgische Olympiades in de Informatica voor de categorie **secundair onderwijs**. Ze bevat 8 vragen die opgelost moeten worden in **maximaal 1u15'**. Naast elke vraag staat een indicatie van de tijd die het kan kosten om de vraag op te lossen. Dit is slechts een schatting.

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

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

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

3. Je antwoorden moeten geschreven zijn in zwarte of blauwe **(bal)pen**. Laat geen antwoorden staan in potlood. Als je kladbladen nodig hebt, vraag ze dan aan een toezichthouder.

4. Voor de meerkeuzevragen, mag je slechts **één enkel antwoord** geven. Kruis het vakje van je keuze aan. Als je je vergist, kleur het foutieve vakje dan helemaal zwart om je antwoord te annuleren. Een correct antwoord levert 1 punt op, geen antwoord is geen punten, en een foutief antwoord wordt bestraft met $-0{,}5$ punten.

5. Op de open vragen **moet** je antwoorden in **pseudo-code**. Voor syntaxfouten worden er geen punten afgetrokken. Tenzij het anders staat aangegeven, is het verboden om voorgedefinieerde functies te gebruiken, met uitzondering van `max (a, b)`,`min (a, b)` en `pow (a, b)` waarbij die laatste a^b berekent.

6. Arrays van lengte n worden geïndexeerd van 0 tot $n-1$. De notatie **`for`** $(i \leftarrow a$ **`to`** b **`step`** $k)$ beschrijft een lus die zich herhaalt zolang $i \leq b$, waarbij i vertrekt van de waarde a en aan het eind van elke iteratie verhoogd wordt met k.

7. Je mag **op geen enkel moment communiceren** met eender wie, tenzij met de toezichthouders of organisatoren. Elke vraag voor verduidelijking of technische problemen mag enkel aan de organisatoren worden gesteld. Voor vragen niet gerelateerd aan de wedstrijd kan je bij de toezichthouders terecht.

8. Het is strikt **verboden te eten of drinken** tijdens de test. De deelnemers mogen **in geen geval hun plaats verlaten** terwijl de test bezig is, ook niet om naar het toilet te gaan of te roken.

9. Je hebt exact **1 uur en een kwartier** om alle vragen te beantwoorden.

Succes !

Vragenlijst finale papier secundair

Vraag 0 – Een opwarmertje (10 min)

(a) Gegeven de functie `notdivisible (x, n)` die **`true`** teruggeeft als `x` niet deelbaar is door `n`, en anders **`false`** teruggeeft. Welke uitdrukking laat je toe om na te gaan dat `x` niet deelbaar is door 5, niet door 3, én niet door 7?

☐	`not (notdivisible (x, 5) or notdivisible (x, 3) or notdivisible (x, 7))`
☐	`notdivisible (x, 5) and notdivisible (x, 3) and notdivisible (x, 7)`
☐	`not notdivisible (x, 5) or not notdivisible (x, 3) or notdivisible (x, 7)`
☐	`notdivisible (x, 5) or notdivisible (x, 3) or notdivisible (x, 7)`

(b) Welke van de volgende uitdrukkingen is equivalent aan : **`not`** $(a > 4)$ **`and`** $3a = b$

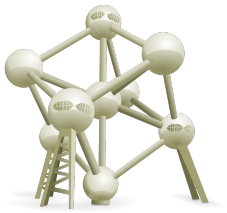
☐	**`not`** $(a > 4$ **`or`** $3a \neq b)$
☐	**`not`** $(a > 4$ **`or`** $3a \neq b)$
☐	**`not`** $(a \leq 4$ **`or`** $3a \neq b)$
☐	**`not`** $(a \leq 4$ **`and`** $3a = b)$

(c) Wat is de waarde van n na het uitvoeren van onderstaand algoritme?

```
n ← 0
a ← 3
while (a ≤ 4)
{
    n ← n + a
}
```

☐	7
☐	11
☐	0
☐	geen van de mogelijkheden hierboven

Vraag 1 – Colis souvenirs (10 min)



Op de Internationale Olympiade in de Informatica moet je België vertegenwoordigen en promoten. Je hebt daarvoor a miniatuurtjes van het atomium en b staaltjes bier. Je wilt pakketjes maken, en elk pakketje bevat een aantal atomiums en een aantal biertjes. Om een maximum aantal personen te bereiken, heb je een algoritme geschreven dat het maximaal aantal pakketjes berekent dat je kan maken, onder de voorwaarden dat elk pakket hetzelfde aantal X atomiums bevat en hetzelfde aantal Y biertjes. Je wilt ook niet terugkeren met de overschot: alle objecten moeten in een pakket terechtkomen.

Hier zijn enkele voorbeelden. Als we 13 atomiums en 7 biertjes hebben, kunnen we maar 1 pakket maken dat 13 atomiums en 7 biertjes bevat. Als we daarentegen 12 atomiums en 2 biertjes hebben, kunnen we 2 pakketten maken met elk 6 atomiums en 1 biertje.

```
Input  : a en b, twee natuurlijke getallen, niet nul, met a ≥ b
Output : het maximaal aantal pakketten dat we kunnen maken

n ← a
m ← b
r ← a mod b

while (r ≠ 0)
{
    n ← m
    m ← r
    [...]
}

return m
```

Welke instructie ontbreekt in dit algoritme om het gewenste resultaat te bereiken?

Q1 **(één instructie)**

. . . . . .

Vraag 2 – Hoeveel mogelijke combinaties ? (5 min)

Als ik n elementen heb en ik wil er p uitkiezen, hoeveel verschillende mogelijkheden heb ik dan om dat te doen? We zijn bijvoorbeeld geïnteresseerd in het aantal mogelijke kaartcombinaties dat we kunnen gedeeld krijgen bij poker. Dit getal heet een binomiaal en wordt ook genoteerd als $\binom{n}{p}$ of C_n^p. We kunnen het resultaat ervan berekenen met behulp van de driehoek van Pascal. De eerste kolom van deze driehoek en de schuine zijde bevatten het getal 1. Verder is de waarde van elk ander vakje gelijk aan de som van de 2 waarden die er boven en linksboven staan.

Dit zijn de eerste 5 lijnen van de driehoek van Pascal:

1				
1	1			
1	2	1		
1	3	3	1	
1	4	6	4	1

Het volgende algoritme berekent de eerste n rijen van de driehoek van Pascal. Het resultaat wordt berekend in een vierkante matrix, en elk element van die matrix dat geen deel uitmaakt van de driehoek krijgt waarde 0.

```
Input  : n, een positief geheel getal
Output : een matrix van n rijen en n kolommen die de
         eerste n rijen van de driehoek van Pascal bevat

c ← matrix van gehele getallen van n rijen en n kolommen, geinitialiseerd met nullen.
c[0][0] ← 1

for (i ← 1 to n − 1 step +1)
{
    c[i][0] ← 1
    for (j ← 1 to i − 1 step +1)
    {
        [...]
    }
    c[i][i] ← 1
}

return c
```

Welke instructie moeten we toevoegen opdat het algoritme het gewenste resultaat berekent?

Q2 **(één instructie)**

. . . . . .

Vraag 3 – Verborgen wiskunde ... (10 min)

Bij het opruimen van de zolder bots je op een oud wiskundeboek. Terwijl je erdoor bladert valt je oog op een bladzijde waarop een algoritme staat dat een mysterieuze functie berekent (`odd (n)` geeft **`true`** terug als n oneven is, anders **`false`**).

```
Input  : k, z, twee positieve gehele getallen
Output : ?

y ← 1

while (k ≠ 0)
{
    if (odd (k))
    {
        k ← k − 1
        y ← y · z
    }
    k ← k div 2                          % div berekent het quotient van de gehele deling
    z ← z · z
}

return y
```

Dit algoritme lijkt je wel interessant te zijn. Helaas, de pagina's met uitleg ontbreken. Wat is de wiskundige functie die hier berekend wordt?

Q3 **(één wiskundige uitdrukking)**

. . . . . .

Vraag 4 – De gehele deling (5 min)

Gegeven twee gehele getallen x en y, met $y \neq 0$. De gehele deling van x door y geeft een quotiënt q en een rest r:

$$x = q \cdot y + r \qquad \text{mett } \mid r \mid < \mid y \mid$$

Hier zijn enkele voorbeelden:

x	y	q	r
12	7	1	5
−12	7	−1	−5
12	−7	−1	5
−12	−7	1	−5

Het volgende algoritme berekent het quotiënt en de rest van de gehele deling. De notatie $\mid x \mid$ staat voor de absolute waarde van x.

```
Input  : x en y, twee gehele getallen
Output : (q,r), het quotient resp. de rest van de gehele deling van x door y

q ← 0
r ← | x |

while (r ≥ | y |)
{
    q ← q + 1
    r ← r− | y |
}

if (x · y < 0)
{
    q ← −q
}

if ([...])
{
    r ← −r
}

return (q, r)
```

Dit algoritme is onvolledig. Er ontbreekt een conditie in het laatste **`if`**-statement. Welke?

Q4 **(één conditie)**

. . . . . .

Vraag 5 – Waar is mijn DVD ? (5 min)

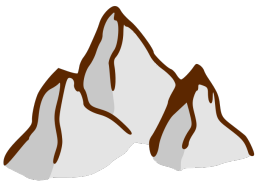
Het is altijd hetzelfde liedje. Wanneer je rustig een film wilt bekijken, vind je met geen mogelijkheid de juiste DVD terug. Om je te helpen, schrijft je broer een algoritme dat toelaat alle DVDs in de kast te overlopen, in een specifieke volgorde en met 2 personen tegelijk. De eerste persoon kijkt of de gezochte DVD niet toevallig de eerste DVD is in de verzameling, terwijl de tweede persoon kijkt of het niet de laatste is. De ene gaat dan verder met controleren of de tweede DVD niet de gezochte is, de ander controleert de voorlaatste, enz.

```
Input  : dvds, een array van n DVDs, indices van 0 tot n − 1
         x, de gezochte DVD
Output : true als DVD x in de array dvds gevonden wordt, anders false

found ← false
i ← 0
while (not found and i ≤ n div 2)        % div berekent het quotient van de gehele deling
{
    if ([...])
    {
        found ← true
    }
    i ← i + 1
}

return found
```

Welke conditie moeten we in het **if**-statement plaatsen om het gewenste resultaat te bekomen?

☐	$dvds[i] = x$ **or** $dvds[n-1-i] = x$
☐	$dvds[i] \neq x$ **and** $dvds[n-i] \neq x$
☐	$dvds[i] \neq x$ **and** $dvds[n-1-i] \neq x$
☐	$dvds[i] = x$ **or** $dvds[n-i] = x$

Vraag 6 – Het langste plateau (10 min)



Volgende maand ga je rondtrekken in de bergen met vrienden. Je hebt het precieze plan van het parcours ontvangen, samen met een hoogteprofiel van de route (d.w.z. een grafiek die aangeeft hoe hoog elk segment van de route gelegen is). Een vraag blijft je achtervolgen, je wilt namelijk weten wat de lengte is van het langste vlakke stuk van het parcours.

Bijvoorbeeld: gegeven de volgende array $[5, 2, 2, 2, 4, 4, 5]$ die een hoogteprofiel voorstelt. Het langste vlakke stuk is dan de sub-array van tweeën, en de lengte ervan is 3.

Gelukkig is er een algoritme om op je vraag te antwoorden:

```
Input  : altitude, een array van n positieve gehele getallen, geïndexeerd van 0 tot n − 1
Output : lengte van het langste vlakke stuk van het parcours.

e ← 0
t ← −1
g ← 0
for (i ← 0 to n − 1 step +1)
{
    if (altitude[i] ≠ t)
    {
        if (e > g)
        {
            g ← e
        }
        e ← 1
        t = altitude[i]
    }
    else
    {
        e ← e + 1
    }
}

return [...]
```

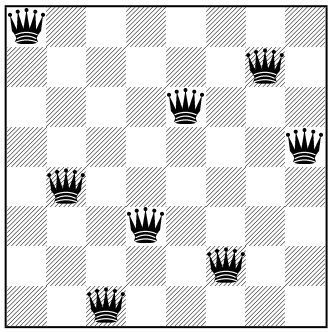
Welke waarde moet het algoritme op het einde teruggeven om het gewenste resultaat te bekomen?

Q6 **(een uitdrukking)**

. . . . . .

Vraag 7 – De 8 koninginnen (15 min)

Je kreeg voor je verjaardag een schaakbord maar je bent helaas geen beste schaker. Je vindt dan maar zelf een spel uit. Je wilt 8 koninginnen op het schaakbord plaatsen, op zo'n manier dat geen koningin een andere koningin op het bord kan aanvallen. Ter herinnering: een koningin kan een andere koningin aanvallen als die op dezelfde rij of kolom of diagonaal staat. Dit is een mogelijke oplossing:



Het is gemakkelijk te controleren dat er op elke kolom slechts 1 koningin staat. We kunnen een oplossing dus voorstellen door middel van een array met lengte 8, waarin elk element het nummer van de rij bevat waar de koningin geplaatst moet worden. Voor de oplossing hierboven, is deze array $[0, 4, 7, 5, 2, 6, 1, 3]$

Het probleem wordt echter moeilijk om op te lossen met de hand als de grootte van het schaakbord toeneemt. Maar je bent niet voor niets informaticus: je hebt een algoritme ontwikkeld die het voor jou oplost! Jouw oplossing maakt gebruik van een ander algoritme, (`unsafe`), dat hieronder beschreven staat. Dit algoritme laat toe om te testen of een koningin op een bepaalde positie aangevallen kan worden of niet.

```
Input  : b, een array van n gehele getallen tussen 0 en n − 1
         y, een geheel getal tussen 0 et n − 1
Output : true, als de koningin op plaats b[y] aangevallen kan worden door een
         koningin geplaatst in een van de kolommen k met k < y, anders false.

function unsafe (b, y)
{
    x ← b[y]
    for (i ← 1 to y step +1)
    {
        t ← b[y − i]
        if ([...])
        {
            return true;
        }
    }
    return false;
}
```

Dit algoritme is onvolledig, er ontbreekt een conditie in het **`if`**-statement. Welke?

Q7a **(een conditie)**

. . . . . .

Vraag 7 (vervolg)

Hier is dan je algoritme om jouw probleem op te lossen, voor een schaakbord met zijde n.

```
Input  : n, positief geheel getal, zijde van het schaakbord,
Output : b, array van n gehele getallen, waar b[i] het rijnummer is waar de (i+1)e
         koningin geplaatst wordt, opdat geen enkele koningin een andere kan aanvallen.

b ← array van n gehele getallen, geïndexeerd van 0 tot n − 1, geïnitialiseerd met 0
y ← 0

while (y < n)
{
    while (b[y] ≠ n and unsafe (b, y))
    {
        b[y] ← b[y] + 1
    }

    if (b[y] < n)
    {
        if (y < n − 1)
        {
            y ← y + 1
            b[y] ← 0
        }
        else
        {
            return b
        }
    }
    else
    {
        [...]
        b[y] ← b[y] + 1
    }
}
```

Hier ontbreekt ook één instructie. Vind ze!

Q7b **(één instructie)**

. . . . . .

Vraag 8 – Exponenten (20 min)

Gegeven een algoritme dat drie positieve gehele getallen x, n en m neemt. Dit algoritme berekent $(x^n) \mod m$, d.w.z. de rest na gehele deling van x^n door m. Dit is het algoritme:

```
Input  : x, n, m, drie positieve gehele getallen
Output : de waarde van (x^n) mod m

result ← 1
while (n ≠ 0)
{
    result ← result * x
    n ← n − 1
}
return result mod m
```

Dit algoritme is niet efficiënt. De uitvoeringstijd is recht evenredig met n. Bovendien, als we gehele getallen gebruiken die met 32 bits worden voorgesteld, dan worden berekeningen met heel grote waarden van n ongeldig (door 'overflow'). Het is mogelijk om een efficiënter algoritme te schrijven, waarvan de uitvoeringstijd recht evenredig is met $\log_2 n$, en dat toelaat om bijvoorbeeld $17^{1000000} \mod 13$ te berekenen door enkel gehele getallen van 32 bits te gebruiken. We vragen je om dat algoritme te vinden, door de onderstaande code te vervolledigen. Je mag de functies div en mod gebruiken, die het quotiënt resp. de rest van de gehele deling berekenen.

```
result ← 1
while (n ≠ 0)
{
    if ([...])      % (a)
    {
        [...]       % (b)
    }
    [...]           % (c)
}
return result
```

Q8a **(een conditie)**

. . . . . .

Q8b **(een instructie)**

. . . . . .

Q8c **(twee instructies)**

. . . . . .

. . . . . .