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 **hoger 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 hoger onderwijs

Vraag 0 – Opwarmertje (10 min)

1. Gegeven de functie `incorrect (n, s)`, die **`true`** teruggeeft als student `s` vraag `n` fout beantwoordt, en anders **`false`** teruggeeft. Welk van de volgende uitdrukkingen test of er geen enkel correct antwoord werd gegeven voor de eerste 3 vragen?

☐	`not incorrect (1, s) or not incorrect (2, s) or not incorrect (3, s)`
☐	`incorrect (1, s) and incorrect (2, s) and incorrect (3, s)`
☐	`incorrect (1, s) or incorrect (2, s) or incorrect (3, s)`
☐	`not (incorrect (1, s) or incorrect (2, s) or incorrect (3, s))`

2. Welk van de volgende uitdrukkingen is equivalent met : `max` $(a,\ 10) = b$ **`and not`** $(a > 2c)$?

☐	**not** (max $(a, 10) = b$ **and** $2c \geq a$)
☐	**not** (max $(a, 10) \neq b$ **or** $a > 2c$)
☐	**not** (max $(a, 10) \neq b$ **or** $2c \geq a$)
☐	**not** (max $(a, 10) \neq b$ **and** $a > 2c$)

3. Wat is de waarde van n na uitvoering van dit algoritme?

```
n ← 0
a ← 5
while (a ≤ 2)
{
    n ← a
    a ← a − 1
}
```

☐	0
☐	1
☐	2
☐	5

Vraag 1 – En als we nu eens in binair tellen ? (5 min)

De binaire voorstelling van een positief geheel getal is een reeks van nullen en enen. Gegeven zo'n reeks van de vorm $b_{n-1}b_{n-2}\cdots b_2b_1b_0$, waarbij $b_i \in \{0,1\}$ voor alle i en $0 \leq i < n$. Zo'n reeks stelt het volgende getal voor:

$$n = \sum_{i=0}^{n-1} b_i \cdot 2^i \quad = \quad b_0 \cdot 2^0 + b_1 \cdot 2^1 + b_2 \cdot 2^2 + \cdots + b_{n-1} \cdot 2^{n-1}$$

Het volgende algoritme berekent de binaire voorstelling van een gegeven positief geheel getal n. De functie `concat (A, B)` laat toe twee reeksen `A` en `B` van karakters aan elkaar te hechten, ze produceert dus één reeks van karakters als resultaat.

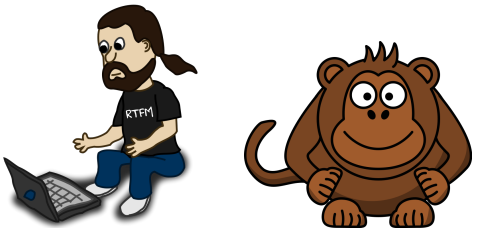
```
Input  : n, positief geheel getal
Output : reeks van karakters die n in binaire vorm voorstelt

convert ← ''                        % convert wordt geïnitialiseerd als een lege reeks

while (n > 0)
{
    if ([...])
    {
        convert ← concat ('0', convert)
    }
    else
    {
        convert ← concat ('1', convert)
    }
    n ← n div 2                                  % div berekent de gehele deling
}

return convert
```

Welke conditie moeten we in het **`if`**-statement zetten opdat het algoritme het gewenste resultaat berekent? **Aandacht**, je mag enkel de volgende operatoren gebruiken: $+$, $-$, $*$ et div (gehele deling).

Q1 **(een conditie)**

. . . . . .

Question 2 – Zijn mensen en apen broers ? (10 min)



Een mogelijke vorm van DNA-analyse omvat het vergelijken van twee DNA-sequenties en het berekenen van de langste gemeenschappelijke subsequentie. Gegeven dus twee sequenties $x = x_1, x_2, \cdots, x_m$ en $y = y_1, y_2, \cdots, y_n$. De langste gemeenschappelijke subsequentie definiëren we als een reeks elementen $z_1, z_2, \cdots, z_k$ die in die volgorde aanwezig zijn in beide sequenties, maar niet noodzakelijk allemaal direct na elkaar. Neem bijvoorbeeld de twee DNA-sequenties `x = GATTACA` en `y = CGTA`. De langste gemeenschappelijke subsequentie is `GTA` en de lengte ervan is 3.

Het onderstaande algoritme laat toe de langste gemeenschappelijke subsequentie te berekenen van twee reeksen x en y van karakters, met lengtes respectievelijk m en n. Het algoritme berekent een matrix c met $m+1$ rijen en $n+1$ kolommen, zodat het element $c[i][j]$ de lengte bevat van de langste gemeenschappelijke subsequenties tussen de karakterreeksen $x[1..i]$ en $y[1..j]$. De notatie $x[1..i]$ stelt de reeks karakters x voor waarbij we enkel de eerste i karakters beschouwen. In het bovenstaande voorbeeld is $x[1..4]$ de karakterreeks `GATA` en $x[1..0]$ is de lege reeks.

```
Input  : x, y, twee arrays van karakters met respectievelijke lengtes m en n
Output : Een matrix result met m+1 rijen en n+1 kolommen, zodat result[i][j]
         de lengte bevat van de langste subsequentie die gemeenschappelijk is aan
         x[1..i] en y[1..j]

result ← matrix van gehele getallen, m+1 rijen en n+1 kolommen, geïnitialiseerd met 0
           waarbij de rijen geïndexeerd worden van 0 tot m en de kolommen van 0 tot n

for (i ← 1 to m step +1)
{
    for (j ← 1 to n step +1)
    {
        if (x[i-1] = y[j-1])
        {
            [...]                          % (a)
        }
        else
        {
            c[i][j] = max ([...]);         % (b)
        }
    }
}
```

Welke instructies ontbreken in dit algoritme om het gewenste resultaat te bereiken?

Q2a **(één instructie)**

. . . . . .

Q2b **(twee expressies)**

. . . . . .

Vraag 3 – Wisselgeld (15 min)

Je hebt een vakantiejob als kassier in de supermarkt. Een vaak terugkerend probleem is dat je wisselgeld in muntstukken moet teruggeven. Om de dag door te komen met je beperkte voorraad muntstukken in de kassa, moet je telkens het minimum aantal muntstukken teruggeven. Gegeven een zekere som wisselgeld M en een reeks muntwaarden $Coins$, zal het volgende algoritme berekenen hoeveel muntstukken je minimaal zal moeten teruggeven.

```
Input  : M, positief geheel getal, som van het terug te geven wisselgeld
         Coins, verzameling strikt positieve gehele getallen, beschikbare muntwaarden
Output : minimum aantal muntstukken met waarden uit Coins dat M kan vormen

arr ← array gehele getallen, grootte M+1, indices van 0 tot M, geïnitialiseerd met 0
tab[0] ← 0

for (m ← 1 to M step +1)
{
    arr[m] ← +∞
    foreach (c ∈ Coins)                    % voor elke muntwaarde in de verzameling Coins
    {
        if (m ≥ c)
        {
            if ([...])
            {
                arr[m] ← arr[m − c]+1
            }
        }
    }
}

return arr[M]
```

Welke conditie is nodig in het **`if`**-statement opdat het algoritme het gewenste resultaat berekent?

☐	$arr[m]-1 \geq arr[m-1]$
☐	$arr[m-1]+c < arr[m]$
☐	$arr[m] > arr[m-1]$
☐	$arr[m-c]+1 < arr[m]$

Vraag 4 – Schrijf een getal achterstevoren (10 min)

1271384
4831721

Het is mogelijk te vertrekken van een getal en een tweede getal te bekomen dat gelijk is aan het eerste, maar dan achterstevoren gelezen, door enkel eenvoudige wiskundige operaties te gebruiken. Het volgende algoritme maakt die berekening en werkt zolang het origineel getal niet eindigt op een nul.

```
Input  : n, strikt positief geheel getal, niet deelbaar door
Output : getal dat achterstevoren gelezen gelijk is aan n

result ← 0
while (n ≠ 0)
{
    [...]
    n ← n div 10                                  % div berekent de gehele deling
}

return result
```

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

☐	$result \leftarrow 10 \cdot result + n \mod 10$
☐	$result \leftarrow (result + n \mod 10) \cdot 10$
☐	$result \leftarrow n \text{ div } 10 + 10 \cdot result$
☐	$result \leftarrow result \text{ div } 10 + n \mod 10$

(De operator $\mod$ berekent de rest na de gehele deling.)

Vraag 5 – Op z'n Russisch (10 min)

Je oude Russische buurman keert terug naar zijn geboorteland en liet je een algoritme na waarvan hij je op het hart drukte dat je het op een dag zou kunnen gebruiken, want het berekent op een gemakkelijke manier een heel nuttige wiskundige functie. Helaas zei hij er niet bij welke functie dat is. Hier is het algoritme:

```
Input  : a en b, twee strikt positieve gehele getallen
Output : ?

r ← b
while (a ≠ 1)
{
    a ← a div 2
    b ← 2 · b

    if (a mod 2 ≠ 0)
    {
        r ← r + b
    }
}

return r
```

Welke wiskundige functie wordt berekend door dit algoritme?

Q5 **(een wiskundige uitdrukking)**

. . . . . .

Vraag 6 – Sorteren ! (15 min)

Het volgende algoritme sorteert de elementen van een array van gehele getallen oplopend. Het algoritme is gebaseerd op een tijdelijke array die informatie opslaat over de te sorteren array. De invulling van deze tijdelijke array ontbreekt in het volgende algoritme, en je moet het vervolledigen.

```
Input  : arr, array gehele getallen met lengte n, geïndexeerd van 0 tot n − 1
         min, het kleinste element van arr
         max, het grootste element van arr
Output : de elementen van arr zijn oplopend gesorteerd

temp ← array van gehele getallen met grootte max − min + 1,
          geïndexeerd van 0 tot max − min, geïnitialiseerd met 0
for (i ← 0 to n − 1 step +1)
{
    [...]
}

j ← 0
for (k ← 0 to max − min step +1)
{
    while (temp[k] ≠ 0)
    {
        data[j] ← min + k
        j ← j + 1
        temp[k] ← temp[k] −1
    }
}
```

Welke ontbrekende instructie is nodig om het gewenste resultaat te bereiken?

Q6 **(één instructie)**

. . . . . .

Vraag 7 – Sub-array met maximale som (20 min)

Gegeven een algoritme dat als input een niet lege array van gehele getallen arr neemt, en de maximale som berekent die men zou kunnen bekomen als men een niet-leeg gedeelte van array arr neemt en de som van de elementen maakt. Dit gedeelte van een array noemen we vanaf nu een sub-array. Neem bijvoorbeeld de array $[1, -2, 4]$. Er zijn 6 mogelijke sub-arrays: $[1]$, $[-2]$, $[4]$, $[1, -2]$, $[-2, 4]$ en $[1, -2, 4]$. Die met de maximale som van elementen is de derde ($[4]$), de som van diens elementen is 4.

```
Input  : arr, array gehele getallen van grootte n, geïndexeerd van 0 tot n − 1, met n > 0
Output : som van de elementen van de niet lege sub-array van arr met maximale som.

max ← arr[0]
for (i ← 0 to n − 1 step +1)
{
    sum ← 0
    for (j ← i to n − 1 step +1)
    {
        sum ← sum + arr[j]
        if (sum > max)
        {
            max ← sum
        }
    }
}
return max
```

Dit algoritme is niet efficiënt. Voor een array van grootte n is de uitvoeringstijd recht evenredig met n^2. De array arr wordt veel te veel overlopen. Het is mogelijk om een efficiënter algoritme te schrijven dat een uitvoeringstijd heeft die recht evenredig is met n in plaats van n^2. We vragen om het volgende algoritme te vervolledigen, waarin de array slechts 1 keer overlopen wordt.

```
i ← 1
s ← arr[0]
max ← arr[0]
while (i ≠ n)
{
    [...]
}
return max
```

Q7 **(drie instructies)**

. . . . . .

. . . . . .

. . . . . .