be-OI 2014 **Finale** 8 februari 2014		

Belgische Informatica-olympiade (duur : 3u30 maximum)

Dit is de vragenlijst van het **gedeelte aan de computer** van de finale van de Belgische Informatica-olympiade. Ze bevat vier vragen. De eerste vraag staat niet op punten maar dient om te oefenen gedurende het eerste **kwartier**. De drie volgende vragen worden pas uitgedeeld nadat dit kwartier is afgelopen, en voor die vragen krijg je dan **maximum 3u30 de tijd**.

Algemene opmerkingen (lees dit aandachtig voor je begint)

1. Leg je studentenkaart of identiteitskaart op de tafel.

2. Ga zitten op de **plaats** die je wordt **toegewezen** door de organisatoren.

3. Je mag alleen iets om te schrijven bij je hebben. Rekenmachines, GSM, ... zijn **verboden**. Laat je persoonlijke bagage achter op de plaats die wordt aangeduid door de toezichters.

4. Je mag **op geen enkel moment met iemand communiceren**, behalve met de organisatoren of toezichters. Alle inhoudelijke of technische vragen mag je enkel stellen aan de organisatoren, of vragen via het indieningsplatform (zie volgende bladzijde). Andere vragen (bvb om kladpapier) mogen gesteld worden aan de toezichters.

5. Je hebt **geen internettoegang** tijdens de proef. Elke poging om andere deelnemers of buitenstaanders te contacteren zal worden bestraft.

6. Je **mag** alle functionaliteit gebruiken van de standaardbibliotheken van de taal die je kiest (Java, C, C++, Pascal, Python of PHP), behalve multi-threading en alles dat communicatie met de buitenwereld impliceert. Enkel de standaard input en output zijn daarvoor toegelaten. In de praktijk betekent dat dat je geen toegang tot het netwerk mag maken, en ook niet mag lezen/schrijven van/naar extra bestanden op de harde schijf.

7. Je mag **kladbladen** vragen aan de toezichters.

8. Je mag in principe je **plaats niet verlaten** tijdens de proef. Moet je dringend naar het toilet, meldt dit dan aan een toezichter. Hij of zij kan dit toelaten of weigeren, afhankelijk van de mogelijkheid om je te laten vergezellen door een van de toezichters.

Wat moet je indienen en hoe werkt de evaluatie ?

Je programma leest parameters en gegevens via de standaard input en moet het resultaat naar de standaard output schrijven. Je moet je programma afgeven via het online indieningssysteem. Enkel de resultaten geproduceerd door de ingediende code worden gebruikt om jouw score te berekenen. De code zelf (de kwaliteit ervan) wordt niet geëvalueerd.

Elke taak is onderverdeeld in subtaken die elk bestaan uit een verzameling tests. Zo kan je met een correct maar niet-optimaal algoritme toch nog een deel van de punten halen. Je krijgt echter slechts punten voor een subtaak als alle tests die er deel van uitmaken succesvol werden afgerond. Is dat niet het geval, dan krijg je geen punten voor die subtaak. Als een subtaak faalt, worden de daaropvolgende subtaken wel nog uitgevoerd. De testen zijn zodanig opgesteld dat hetzelfde algoritme altijd hetzelfde aantal punten oplevert, ongeacht de programmeertaal die je gebruikt.

Een deelnemer die de wedstrijd volledig beëindigt (de maximumscore behaalt) voor het eindsignaal, krijgt **1 extra punt per 10 minuten** die nog resten tot het eindsignaal. In het geval van gelijke scores op het eindtotaal van de finale, bepalen de scores zonder deze bonussen de plaats. Als er dan opnieuw gelijkheid is, wordt het tijdstip waarop je de code indiende waarmee je jouw eindscore hebt behaald, gebruikt om het eindklassement op te stellen.

Praktische instructies

Deze instructies leggen uit hoe je aan je programma kan werken en hoe je het nadien moet indienen op de officiële server. We raden je aan dit te lezen en tegelijk toe te passen op *Taak 0*.

Downloaden en aanpassen van de skeletprogramma's

- Op de wedstrijdsite vindt je voor elke taak bij *Opgave* een link naar een zip-bestand. Dit bevat de skeletprogramma's voor die taak. Klik en download het bestand. Alle downloads komen automatisch terecht in de map *Downloads* in je persoonlijke map. Klik op het map-icoon in de linker toolbar om erheen te gaan. Je kan een gedownload bestand gemakkelijk verplaatsen, bvb naar het bureaublad, door het met de muis verslepen.
- Rechtsklik op het zip-bestand en klik op *Hier Uitpakken*. In de map (bvb. `Q0`) vindt je voor elke programmeertaal een skeletprogramma dat je kan aanvullen (bvb. `task.c`), een voorbeeld van inputgegevens (`input.txt`) en de overeenkomstige resultaten die verwacht worden (`output-expected.txt`).
- Kies de taal waarin je wilt werken, en dubbelklik op het overeenkomstige codebestand. Het opent in de teksteditor *gedit* . Er zijn andere editors beschikbaar als je wil : die kan je starten door de naam ervan in te typen in de zoekbalk na klikken op de knop *Dash home* linksboven. We raden aan om in *gedit* te blijven werken ; enkel *gedit* werd speciaal geconfigureerd voor deze proef.
- Een skeletprogramma bevat alvast code voor het inlezen van inputbestanden en het schrijven van output. Je moet het aanvullen met het algoritme dat het probleem oplost.

Compileren, uitvoeren en testen van je programma

- In *gedit* kan je **Ctrl+R** drukken om het programma waaraan je werkt te testen. Deze sneltoets bewaart het bestand, compileert het, voert het uit en test het met het bestand `input.txt` als inputgegevens. De resultaten verschijnen in de console in het onderste venster van *gedit*.
- Als je programma met succes werd gecompileerd, zal het uitgevoerd worden met als inputgegevens het bestand `input.txt` dat aanwezig is in de map. Als deze uitvoering zonder fouten verloopt, dan zal de output van je programma zich in het bestand `output.txt` bevinden. Dat wordt op zijn beurt vergeleken met het verwachte resultaat dat in het bestand `output-expected.txt` staat. Als ergens onderweg een fout optreedt (bij compilatie of uitvoering), wordt informatie daarover weergegeven onderaan in de console van *gedit*.
- Je mag het bestand `input.txt` aanpassen als je andere tests wil uitvoeren, maar let erop dat je in dat geval ook de overeenkomstige verwachte output in het bestand `output-expected.txt` moet aanpassen.

Je programma indienen

- Om in te dienen ga je op de website naar de sectie *Indieningen* voor de taak waar je aan werkt. Klik op *Bladeren*, selecteer het bestand waaraan je werkte, en klik *Indienen*.
- De resultaten van je indiening verschijnen op dezelfde pagina in de kolom *Publieke score*. De eindscore voor een taak zal de hoogste score zijn die je ooit voor die taak hebt behaald.
- Je kan een gedetailleerd verslag, met onder andere uitvoeringstijd voor alle tests, zien door op *Details* te klikken.
- Alleen het resultaat van jouw programma op de automatische tests bepaalt het eindresultaat. Met de kwaliteit van je code wordt helemaal geen rekening gehouden.

Werking van het indieningssysteem

- Zodra je nieuwe code indient, wordt die in een wachtrij gezet. Ze zal zo snel mogelijk worden uitgevoerd maar dat hangt ook af van hoeveel andere deelnemers er recent nieuwe code hebben ingediend. Het is dus mogelijk dat naar het einde van de proef toe, de wachttijd voor je resultaten oploopt tot enkele minuten.
- Je kan geen code indienen als je minder dan **20 seconden** voordien al code had ingediend. Let hiervoor op bij de laatste minuten van de wedstrijd !
- Als je na het eindsignaal nog code in de wachtrij hebt zitten, zal die uiteraard nog meetellen. Je zal echter niet meer weten hoeveel punten ze je oplevert.
- **Alleen de best behaalde score op elke taak die je indient zal meetellen om je eindscore op de computerproef te berekenen.**
- Voordat je indient : zorg er zeker voor dat je alle tekstberichten die je eventueel in je programma laat uitprinten om te *debuggen*, ook terug verwijdert. Anders verschijnen ze op de standaard output, en wordt het resultaat van je programma incorrect.

Opmerkingen

- Documentatie voor elke taal bevindt zich op de wedstrijdserver onder *Documentatie* .
- Je kan een vraag stellen aan de organisatoren van de wedstrijd via dezelfde website onder *Communicatie*. Je wordt automatisch op de hoogte gebracht als er een antwoord is gekomen.
- Als je je code nog niet hebt ingediend kan het altijd handig zijn om regelmatig ergens een kopie te bewaren, voor het geval dat je per ongeluk iets verkeerd doet. Deze *"backups"* kunnen veel gevloek voorkomen.
- De code in *gedit* zal nooit worden bekeken. Alleen de code die werd ingediend via de server telt mee voor je eindscore.
- **known bug :** Na het testen van jouw programma met **Ctrl+R** blijft de muiscursor er na afloop uitzien alsof de test nog steeds bezig is. Dit is een bug in gedit zelf. De test is wel degelijk afgelopen en je kan gewoon rustig verder werken. De muiscursor herstelt zich na een tijd vanzelf.
- Als er bovenaan je scherm meldingen zouden verschijnen over bvb *muisintegratie* mag je deze gewoon wegklikken. Ze zijn onderdeel van de virtuele omgeving.
- Op de wedstrijdserver staat ook een sectie *Testen* onderaan. Ze laat je toe om je programma samen met een zelfgemaakt inputbestand in te dienen om het dan op de evaluatieserver zelf te laten uitvoeren. Je mag deze functionaliteit gebruiken als je dat wil.

Taak 0 – (0 ptn)

Dit is een kleine oefening om je vertrouwd te maken met de werkomgeving en de server voor het indienen van de code. Je moet gewoon de som maken van twee positieve gehele getallen.

Taak

Schrijf een programma dat, gegeven 2 positieve gehele getallen, hun som berekent.

Limieten en beperkingen

- $1 \leq a, b \leq X_{MAX}$, de getallen die opgeteld moeten worden ;

	X_{MAX}
Sub-taak A	100
Sub-taak B	2 000 000

Maximale uitvoeringstijd : **1 seconde**. Geheugenlimiet : **256 MB**.

Input

De input die je programma krijgt heeft het volgende formaat :

- De eerste en enige lijn bevat twee positieve gehele getallen a en b, gescheiden door één spatie.
- De input eindigt met een nieuwe lijn.

Output

Je programma schrijft de som van a en b naar de output, gevolgd door een nieuwe lijn.

Voorbeeld

Gegeven de volgende input die aan je programma wordt gegeven :

```
10 81
```

De output van je programma moet dan zijn :

```
91
```

Taak 1 – Hittegolf (Heat) – (30 ptn)(A : 11 ptn, B : 19 ptn)

Informatici hebben de reputatie om liever binnen te blijven bij hun computer in plaats van buiten te komen om te genieten van het zonnetje in de warme zomermaanden. Een nieuwe wetenschappelijke studie heeft de reden daarvoor gevonden : informatici kunnen gewoon slecht tegen de hitte. Ze verliezen hun denkvermogen als ze te lang worden blootgesteld aan een temperatuur die boven hun persoonlijke limiet ligt.

Het comité van de Internationale Olympiade Informatica (IOI) is verontrust door deze ontdekking, en wil maatregelen nemen om geschiktere gastlanden te kunnen selecteren voor deze wedstrijd. Je moet een programma schrijven dat, voor een bepaalde bestemming, voor elke deelnemer de *hittegolf-index* berekent. Die is gedefinieerd als het maximaal aantal opeenvolgende dagen tijdens dewelke de temperatuur groter dan of gelijk is aan de persoonlijke temperatuurlimiet van de deelnemer. Je krijgt voor elke kandidaat-gaststad een lijst met de dagelijkse maximumtemperaturen gemeten tijdens de laatste zomers. Je kent ook de persoonlijke temperatuurlimieten die de deelnemers kunnen verdragen.

Taak

Schrijf een algoritme dat, gegeven de temperatuurlimiet van een deelnemer en een reeks temperatuurmetingen, de lengte berekent van de langste opeenvolgende reeks metingen waarbij de temperaturen groter dan of gelijk zijn aan die limiet.

Limieten en beperkingen

- $0 < N < N_{MAX}$, het aantal dagen waarop de temperatuur werd gemeten.
- $0 < H < T_{MAX}$, de persoonlijke limiet van de deelnemer. Die verdraagt dus geen temperaturen groter dan of gelijk aan dit getal.
- $0 < T_i < T_{MAX}$, de maximumtemperatuur op dag i.

	N_{MAX}	T_{MAX}
Sub-taak A	50	10 000
Sub-taak B	100 000	10 000

Maximale uitvoeringstijd : **2 seconden**. Geheugenlimiet : **128 MB**.

Input

De input die je programma krijgt heeft het volgende formaat :

- De eerste lijn bevat N en H, gescheiden door één spatie.
- De volgende N lijnen bevatten alle T_i.
- De input eindigt met een nieuwe lijn.

Output

Jouw programma moet één enkel getal teruggeven : de lengte van de langste 'hittegolf' volgens de gegeven temperatuurlimiet. De output eindigt met een nieuwe lijn.

Voorbeeld 1

Voor de volgende input :

```
13 38
31
39
42
33
30
33
40
38
39
41
37
34
27
```

moet jouw programma deze output geven :

```
4
```

Voorbeeld 2

Voor de volgende input :

```
5 40
28
39
38
27
34
```

moet jouw programma deze output geven :

```
0
```

Taak 2 – Vuurwerk (Firework) – (60 ptn) (A : 23 ptn, B : 19 ptn, C : 18 ptn)

In Taiwan, zoals ook elders in dat deel van Azië, steekt men vaak vuurwerk af om iets te vieren. Bij de start van de Internationale Olympiade Informatica willen de organisatoren een grote vuurwerkshow organiseren en ze hebben je nodig om het spectaculairste resultaat te boeken.

Je hebt een hoeveelheid pijlen ter beschikking. Deze pijlen worden aan jou gegeven in een bepaalde, vaste volgorde. Je kan de vuurwerkshow beginnen bij eender welke pijl je maar wilt, maar je kan de volgorde waarin ze worden afgestoken niet veranderen : eens een bepaalde pijl werd gelanceerd, kunnen alle vorige pijlen niet meer worden afgestoken. Elke vuurpijl heeft ook de volgende eigenschappen :

— x : de positie van het vuurwerkboeket aan de horizon
— $[a, b]$: het harmonie-interval van de pijl (hieronder verder uitgelegd)
— p : de schoonheids-index van de pijl

Om het beste vuurwerk te bekomen, wil je zeker zijn dat elke vuurpijl die je afsteekt een harmonie-interval heeft dat de positie van het voorgaande vuurwerkboeket bevat. Bijvoorbeeld, als je een pijl wil afsteken die harmonie-interval $[a_j, b_j]$ heeft, kan dat enkel als de vorige pijl een positie x_{j-1} heeft die ligt in het interval tussen a_j en b_j (inclusief). Je kan natuurlijk eender welke pijl afsteken als eerste pijl, omdat die niet voorafgegaan wordt door een andere. De totale schoonheid van het spektakel is de som van de schoonheids-indices van alle afgeschoten pijlen.

Taak

Bepaal de maximale schoonheid die je kan bereiken met een gegeven volgorde van beschikbare pijlen.

Limieten en beperkingen

- $1 \leq N \leq N_{MAX}$, het aantal pijlen.
- $0 \leq a_i \leq b_i \leq 500\,000$, het harmonie-interval van pijl i.
- $0 \leq x_i \leq 500\,000$, de boeket-positie van pijl i .
- $0 < p_i < P_{MAX}$, de schoonheids-index van pijl i.

	N_{MAX}	P_{MAX}
Sub-taak A	10	100 000
Sub-taak B	1 500	100 000
Sub-taak C	25 000	1 000 000

Voor subtaak C geldt bovendien dat de lengte van de harmonie-intervallen steeds kleiner is dan 20, en de oplossing steeds kleiner is dan 2 miljard.

Maximale uitvoeringstijd : **2,5 seconden**. Geheugenlimiet : **512 MB**.

Input

De input die je programma krijgt heeft het volgende formaat :

- De eerste lijn bevat N.
- De volgende N lijnen bevatten x_i a_i b_i p_i (telkens gescheiden door een spatie) die de eigenschappen van pijl i beschrijven.
- De input eindigt met een nieuwe lijn.

Output

De maximale schoonheids-score die behaald kan worden. De output eindigt met een nieuwe lijn.

Voorbeeld 1

Voor de volgende input :

```
5
9 0 6 4
5 7 8 1
7 0 6 2
2 1 5 3
9 3 8 2
```

moet jouw programma deze output geven :

```
5
```

Uitleg : als pijl 1 wordt afgeschoten heeft die boeket-positie 9. Geen enkele van de daaropvolgende pijlen bevat deze positie in hun harmonie-interval. Het beste vuurwerk dat je startend van die pijl kan maken heeft dus een schoonheidsscore van 4. Pijl 2 kan gevolgd worden door pijlen 3, 4 of 5 omdat de intervallen $[0, 6]$, $[1, 5]$ en $[3, 8]$ allemaal de boeket-positie van pijl 2 bevatten (die 5 is). Daarbovenop kan pijl 5 ook worden afgeschoten na pijl 3. De pijlen 2,3,5 afsteken geeft een totale schoonheidsscore van $1 + 2 + 2 = 5$, wat het maximum haalbare is.

Voorbeeld 2

Voor de volgende input :

```
6
75 10 99 100
15 12 72 100
30 24 72 100
45 11 14 100
60 31 44 100
72 62 72 100
```

moet jouw programma deze output geven :

```
100
```

Uitleg : Het maakt niet uit welke pijl eerst wordt afgestoken, want er is bij de daaropvolgende pijlen geen enkele die de boeket-positie ervan in zijn harmonie-interval heeft.

Taak 3 – Het rijstveld van Mr. Chen (Paddy) – (60 ptn) (A : 20 ptn, B : 20 ptn, C : 20 ptn)

Mr. Chen is een grootgrondbezitter in de provincie Taoyuan, ten zuid-oosten van Taipei. Hij wil een nieuw rijstveld aanleggen, zo groot mogelijk, in het midden van zijn terrein. Het rijstveld moet absoluut rechthoekig zijn, en er moet een hek rond geplaatst worden om het netjes te scheiden van de rest van zijn gronden.

Het enige hek dat Mr. Chen heeft is een lange geschakelde afrastering uit één stuk, die bestaat uit een aaneenschakeling van kleinere stukken hek die allemaal een verschillende lengte kunnen hebben. Mr. Chen wil de oppervlakte van zijn rijstveld optimaliseren en heeft daar jouw vernuft voor nodig.

Taak

Gegeven de lengtes van de secties waaruit de afrastering bestaat, moet je de maximale oppervlakte berekenen die een rechthoekig rijstveld kan hebben zodanig dat alle secties van het hek gebruikt worden in ongewijzigde volgorde. Als het niet mogelijk is een rechthoekig veld te vormen, moet jouw programma 0 teruggeven.

Hint : De rechthoek die bij eenzelfde omtrek voor een maximale oppervlakte zorgt, is altijd diegene die zo goed mogelijk een vierkant benadert.

Limieten en beperkingen

- $1 \leq N \leq N_{MAX}$, het aantal secties waaruit het hek bestaat ;
- $1 \leq L_i \leq 1\,000$, de lengte van sectie i van het hek ;
- $1 \leq T \leq T_{MAX}$, de totale lengte van het hek ;

	N_{MAX}	T_{MAX}
Sub-taak A	30	30 000
Sub-taak B	1 000	100 000
Sub-taak C	100 000	100 000 000

Merk op dat voor sub-taak C, de totale oppervlakte groter kan worden dan $6,25 \times 10^{14}$, zodat het nodig kan zijn om voor deze waarde een variabele te gebruiken die groot genoeg kan worden (afhankelijk van je programmeertaal).

Maximale uitvoeringstijd : **4 seconden**. Geheugenlimiet : **512 MB**.

Input

De input die je programma krijgt heeft het volgende formaat :

- De eerste lijn bevat een positief geheel getal N ;
- De volgende N lijnen bevatten elk een positief geheel getal L_i ;
- De input eindigt met een nieuwe lijn.

Output

Jouw programma geeft als output de maximale oppervlakte die een rechthoekig rijstveld kan hebben, als het omgeven wordt met de gegeven geschakelde afrastering (die ongewijzigd moet blijven). Als het niet mogelijk is om een rechthoek te vormen met dit hek, moet het programma 0 teruggeven. De output eindigt met een nieuwe lijn.

Voorbeeld 1

Voor de volgende input :

```
7
2
4
1
2
3
1
1
```

moet jouw programma deze output geven :

```
12
```

Uitleg : het grootste veld dat we kunnen maken met deze afrastering is er een van 3 op 4. De eerste zijde wordt gevormd door een sectie van lengte 4 (L_2), de tweede zijde bevat de daaropvolgende secties van lengte 1 (L_3) en 2 (L_4), de derde zijde heeft dus secties met lengtes 3 (L_5) en 1 (L_6), en de laatste zijde de resterende secties met lengtes 1 (L_7) en 2 (L_1). Het is ook mogelijk om een veld van 6 op 1 te maken, maar de oppervlakte daarvan is kleiner.

Voorbeeld 2

Voor de volgende input :

```
5
2
3
2
3
2
```

moet jouw programma deze output geven :

```
0
```

Uitleg : het is niet mogelijk om een rechthoek te vormen met alle secties van de gegeven afrastering.