be-OI 2013 **Finale** 9 maart 2013	**Gegevens invullen in HOOFDLETTERS en LEESBAAR, aub** VOORNAAM+NAAM (HOOFDLETTERS!) : . . . . . . SCHOOL : . . . . . . LOGIN : WACHTWOORD : ZAAL : CANDIX / DAO	**Gereserveerd**

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 je vertrouwd te maken met het systeem 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**.

Af te geven

1. 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 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.
2. Het tijdstip van indienen heeft geen invloed op de score. Enkel in het geval van gelijke scores op het eindtotaal van de finale, wordt het tijdstip van indienen van de code waarop deze scores zijn berekend, gebruikt om de gedeelde plaatsen te rangschikken en zo het eindklassement vast te legggen.
3. Je moet ook je vragenlijst teruggeven, met de hoofding bovenaan op het eerste blad correct ingevuld.

Algemene opmerkingen (lees dit aandachtig voor je begint)

1. Schrijf je naam, voornaam en school op de eerste pagina. Omcirkel de naam van de zaal waarin je je bevindt. 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 toezichthouders.
4. Je mag **op geen enkel moment met iemand communiceren**, behalve met de organisatoren of toezichthouders. Alle inhoudelijke of technische vragen mag je enkel stellen aan de organisatoren. Andere vragen (bvb om kladpapier) mogen gesteld worden aan de toezichthouders.
5. Je **mag** alle functionaliteit gebruiken van de standaardbibliotheken van de taal die je kiest (Java, C, C++, Pascal, Python of PHP), behalve 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 bestanden op de harde schijf.
6. Je mag **kladbladen** vragen aan de toezichthouders.
7. Je mag in principe je **plaats niet verlaten** tijdens de proef. Moet je dringend naar het toilet, meldt dit dan aan een toezichthouder. Hij of zij kan dit toelaten of weigeren, afhankelijk van de mogelijkheid om je te laten vergezellen door een van de toezichters.

Dit werk is vrijgegeven onder de licentie :
'Creative Commons Naamsvermelding 2.0 België'

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 de inhoud tegelijk toe te passen op *Taak 0*.

Inloggen op je computer en de server

- In het begin zal je computer een login-venster tonen. Selecteer jouw taal (Nederlands) en de gebruiker *BeOI user* en klik op *log-in*.
- Ga naar de server door te dubbelklikken op het icoon *BeOI server* op het bureaublad. De welkomstpagina van de server verschijnt. Kies jouw taal (Nederlands). Op de volgende pagina geef je de login en het paswoord weer dat je gekregen hebt en klik je op *Valideren*. Je ziet nu jouw persoonlijke hoofdpagina.
- Jouw hoofdpagina bevat de lijst met taken die overeenkomen met deze vragenlijst. Klik bijvoorbeeld op taak 0. De pagina van die taak verschijnt.
- Taakpagina's zijn georganiseerd in twee kolommen. Links staat de skeletprogramma's voor elke programmeertaal, en de resultaten (in het begin leeg) van je vorige indieningen. Rechts staat een tekstvak dat je zal gebruiken om jouw programma's in te dienen op de server.

Downloaden en aanpassen van de skeletprogramma's

- Eerst moet je de skeletprogramma's downloaden van de server. Ga naar jouw hoofdpagina. Bovenaan de pagina, klik je op de link *Downloaden van de skeletten*. Een gecomprimeerd bestand `skeleton.tgz` wordt nu gedownload. Open het met het *Archiefbeheer* (dit is het standaardprogramma) en pak de inhoud uit naar je bureaublad (*Archiveren* > *Extract* > *Desktop*).
- Nu op het bureaublad, vind je de mappen die je net hebt uitgepakt. Er is één map per vraag (bvb. `Q0`), elke vraag bevat één map per taal (bvb. `C`). Elk van die mappen bevat het skeletprogramma dat je moet aanvullen (bvb. `main.c`), een voorbeeld van inputgegevens (`input.txt`) en de overeenkomstige resultaten die verwacht worden (`output-expected.txt`).
- Dubbelklik op het skeletprogramma dat je wil aanpassen. Het opent in de teksteditor *gedit*. Je kan nu je programma aanpassen in *gedit*. Andere software is beschikbaar (zie het *Application menu* linksboven) maar enkel *gedit* werd speciaal geconfigureerd voor deze proef.
- Als je een enkel skeletprogramma terug zou willen opzoeken, dan zijn deze ook beschikbaar op de pagina van elke taak. In de linkerkolom, sectie *Oorspronkelijke codes*, selecteer je de programmeertaal waarin je wil werken. Het te vervolledigen skeletprogramma voor die taak verschijnt. Je kan het naar je teksteditor kopiëren met copy-paste.

Compileren, uitvoeren en testen van je programma

- Vanuit *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 een consolevenster op het onderste schermdeel 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 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 moet je je programma copy-pasten in het daarvoor voorziene veld op de server. In *gedit* selecteer je je volledige programma en kopieer je het (Ctrl-A, Ctrl-C). Op de server ga je naar de pagina van de taak waaraan je werkt. Plak jouw programma (Ctrl-V) in het tekstveld in de rechterkolom, selecteer de taal die je gebruikt hebt in het menu daar net boven (let erop dat je dat niet vergeet !) en klik op *Indienen*.
- De server compileert en evalueert jouw programma. De evaluatie bestaat uit een reeks tests met verschillende inputgegevens. Voor elke test is een tijdslimiet vastgesteld, die afhangt van de programmeertaal. De evaluatie wordt onderbroken zodra er een test faalt, ofwel door een foute uitkomst of door een gebrek aan tijd.
- De resultaten van je ingediend programma verschijnen bovenaan de rechterkolom. Let op : je moet de pagina verversen om de resultaten bij te werken. De status kan zijn :

  In afwachting van beoordeling : De evaluatie is nog bezig.

  Compilatiefout : Je programma kon niet gecompileerd worden.

  Mislukt : Je programma werd gecompileerd maar een test heeft gefaald door een verkeerd resultaat.

  Tijd is verstreken : Je programma werd gecompileerd maar een test heeft gefaald wegens het overschrijden van de tijdslimiet.

  Geslaagd : Je programma werd gecompileerd en slaagde in alle tests.

  De resultaten van elke uitgevoerde test worden eronder weergegeven. Als je op de link *Code* klikt kan je de code die je hebt ingediend opnieuw bekijken. De historiek van je indieningen voor deze taak bevindt zich onderaan de linkerkolom.
- Alleen het resultaat van jouw programma op de automatische tests bepaalt het eindresultaat. Met de kwaliteit van je code wordt dus helemaal geen rekening gehouden.

Werking van het indienings- en evaluatiesysteem

- 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.
- Als je nieuwe code indient terwijl je op dezelfde vraag nog code in de wachtrij had staan, zal de oude code geannuleerd worden en wordt de nieuwe code opnieuw toegevoegd aan het eind van de wachtrij.
- Als je na het einde van de proeftijd 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 in de map `Docs` op het bureaublad.
- Als je hulp nodig hebt bij enkele dingen die op deze pagina staan, vraag het dan aan een toezichthouder.
- 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.

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

Je programma hoeft enkel problemen te kunnen oplossen die zich binnen deze limieten bevinden. Alle testen die worden uitgevoerd zullen deze limieten respecteren.

- $1 \leq a, b \leq 100$, de getallen die opgeteld moeten worden ;

Wat er ook gebeurt, je programma zal stopgezet worden na **1 seconde** uitvoeringstijd, ongeacht de programmeertaal. Je programma mag niet meer dan **10 MB** geheugen gebruiken.

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 – Bezoek aan Australië – (30 ptn)

In het kader van de Internationale Informatica-Olympiade 2013 in Australië willen de verantwoordelijken van de delegatie een aantal toeristische trekpleisters bezoeken in de regio Brisbane, in de dagen voor de competitie. De organisatoren laten iedereen zijn/haar voorkeuren uitdrukken aan de hand van een stemsysteem. Iedere deelnemer gebruikt zijn punten behaald in de nationale finale om een gewicht toe te kennen, naargelang zijn/haar voorkeuren, aan de bezoeken die hij/zij wil afleggen. Bijvoorbeeld :

Deelnemer A :
- Bezoek 1 : Nationaal Park Springbrook – 5 ptn
- Bezoek 3 : Byronbaai – 20 ptn
- Bezoek 5 : Frasereiland – 10 ptn
- Bezoek 9 : Duiken aan het Groot Barrièrerif – 37 ptn

Deelnemer B :
- Bezoek 3 : Byronbaai – 20 ptn
- Bezoek 4 : Coot-tha berg – 12 ptn
- Bezoek 5 : Frasereiland – 10 ptn
- Bezoek 9 : Duiken aan het Groot Barrièrerif – 20 ptn

Deze lijsten worden daarna twee aan twee gecombineerd om de prioriteiten van de bezoeken van de delegatie te bepalen, en dit op de volgende manier. Indien een bezoek maar op één van beide lijsten voorkomt, is de totaalscore van dit bezoek deze van de lijst waarop het bezoek voorkomt. Indien een bezoek op beide lijsten staat, is de totaalscore van dit bezoek het maximum van de twee scores. Bijvoorbeeld, de combinatie van de twee bovenstaande lijsten is de volgende :

- Bezoek 1 : Nationaal Park Springbrook – 5 ptn
- Bezoek 3 : Byronbaai – 20 ptn
- Bezoek 4 : Coot-tha berg – 12 ptn
- Bezoek 5 : Frasereiland – 10 ptn
- Bezoek 9 : Duiken aan het Great Barrier rif – 37 ptn

Wetende dat deze lijsten geordend zijn volgens het nummer van het bezoek, wordt aan jou gevraagd om een algoritme te schrijven dat de resulterende lijst berekent die gevormd wordt door de twee gegeven lijsten te combineren.

Taak

Schrijf een algoritme dat twee lijsten combineert, die geordend zijn op nummer van het bezoek, volgens de hierboven beschreven regels. Elke te bezoeken site wordt voorgesteld door een nummer en zal een score hebben (een positief geheel getal).

Limieten en beperkingen

Je programma hoeft enkel problemen te kunnen oplossen die zich binnen deze limieten bevinden. Alle testen die worden uitgevoerd zullen deze limieten respecteren.

- $0 < n, m < 100\,000$, de lengtes van de twee lijsten.
- $0 < v < 1\,000\,000$, het nummer van de te bezoeken site
- $0 < w < 1\,000\,000$, het gewicht toegekend aan een bezoek

Wat er ook gebeurt, je programma zal stopgezet worden na **2 sec.** uitvoeringstijd, ongeacht de programmeertaal. Je programma mag niet meer dan **200 MB** geheugen gebruiken.

Input

De input die je programma krijgt heeft het volgende formaat :
- De eerste lijn bevat n
- De n volgende lijnen bevatten elk een nummer van een bezoek en een gewicht, gescheiden door een spatie
- Dan volgt een lijn die m bevat
- De m volgende lijnen bevatten elk een nummer van een bezoek en een gewicht, gescheiden door een spatie
- De input eindigt met een nieuwe lijn.

Output

De output van jouw programma bestaat uit een reeks lijnen die elk twee getallen bevatten, gescheiden door een spatie : het nummer van het bezoek en het gewicht ervan in de gecombineerde lijst. De lijst **moet geordend zijn** op nummer van het bezoek. Het bestand eindigt met een nieuwe lijn.

Voorbeeld

Voor de volgende input

```
4
1 5
3 20
5 10
9 37
4
3 20
4 12
5 10
9 20
```

Moet jouw programma het volgende teruggeven :

```
1 5
3 20
4 12
5 10
9 37
```

Scores

– Tests 1–5 (15 ptn) : $0 < n, m < 1\,000$
– Tests 6–10 (15 ptn) : $0 < n, m < 100\,000$

Taak 2 – Groot Barrièrerif – (60 ptn)

Een duiktocht aan het Groot Barrièrerif is uitverkozen als activiteit. De Belgische delegatie gaat er naartoe om de mooiste koraalkolonies ter wereld te zien, samen met meer dan 1500 soorten vissen en schaaldieren. Onze delegatie hoopt mooie foto's te maken gedurende deze duiktocht.

De duiktocht volgt een lineair tracé langs koraalsites waarin verschillende uitzonderlijke vissen zich verschuilen. Zich verplaatsen van de ene site naar de volgende neemt 5 minuten zwemtijd in beslag. Eens aangekomen bij een nieuwe site, hebben onze beginnende duikers enkel de tijd om één enkele foto te nemen van elke vis, alvorens deze zich verstoppen in het koraal om pas 10 minuten later terug te verschijnen. Vervolgens kunnen ze onmiddellijk beslissen om hun weg verder te zetten, om terug achteruit te gaan, of om ter plaatse te blijven.

Gezien de beperkte capaciteit van hun duikflessen kunnen de duikers zich niet onbeperkt in het water ophouden en moeten ze terug naar het oppervlak zodra hun fles leeg is. Gelukkig kent de ervaren gids het exact aantal vissen op elke site, en zijn onze duikers experts in algoritmiek. Help hen om het aantal foto's gemaakt tijdens hun duiktocht te maximaliseren !

Taak

Gegeven de tijd van de duiktocht en het aantal vissen op elke site, wordt je gevraagd om het maximum aantal foto's te geven die genomen kunnen worden tijdens de duiktocht, wetende dat de duiktocht begint aan de eerste site. Een veiligheidsmarge die de tijd om af te dalen en te stijgen in acht neemt, is reeds afgetrokken van de duiktijd. De duiktijd kan dus volledig gespendeerd worden aan het nemen van foto's.

Limieten en beperkingen

Je programma hoeft enkel problemen te kunnen oplossen die zich binnen deze limieten bevinden. Alle testen die worden uitgevoerd zullen deze limieten respecteren.

- $0 < n < 5\,000\,000$, het aantal sites.
- $0 < t < 5\,000\,000$, de duur van de duiktocht in minuten.
- $0 < a_i < 1\,000$, het aantal vissen op site i.

Wat er ook gebeurt, je programma zal stopgezet worden na **2 sec.** uitvoeringstijd, ongeacht de programmeertaal. Je programma mag niet meer dan **200 MB** geheugen gebruiken.

Input

De input die je programma krijgt heeft het volgende formaat :

- De eerste regel bevat n en t, gescheiden door een spatie.
- Dan volgen n regels, elk ervan bevat het aantal vissen a_i dat zich op de i-de site bevindt.
- De input eindigt met een nieuwe regel.

Output

Het maximum aantal foto's van vissen dat genomen kan worden. Het bestand eindigt met een nieuwe regel.

Voorbeeld

Voor de volgende input :

```
5 20
3
4
3
5
1
```

Moet jouw programma het volgende teruggeven :

```
18
```

Uitleg : onze duikers nemen 3 foto's op de 1ste positie op minuut 0, 4 op de 2de positie op minuut 5, 3 op de 3de positie op minuut 10, 5 op de 4de positie op minuut 15, en keren dan terug (achterwaarts) om 3 foto's te nemen op de 3de positie op minuut 20. Dit resultaat (18) is het best haalbare.

Scores

- Tests 1–3 (18 ptn) : $0 < n < 200$ en $0 < t < 20$
- Tests 4–6 (21 ptn) : $0 < n < 10\,000$ en $0 < t < 10\,000$
- Tests 7–9 (21 ptn) : $0 < n < 5\,000\,000$ en $0 < t < 5\,000\,000$

Taak 3 – Je valt in herhaling

In het broncodebestand voor deze oefening vind je een functie f. Probeer niet te begrijpen wat deze functie precies doet, je hoeft enkel te weten dat de functie één geheel getal n neemt als parameter en ook een geheel getal produceert als waarde. Deze waarde f ligt steeds in het interval $[-2^{31}, 2^{31}-1]$ (=$[-2\,147\,483\,648, 2\,147\,483\,647]$), het standaardbereik voor een geheel getal van 32 bits.

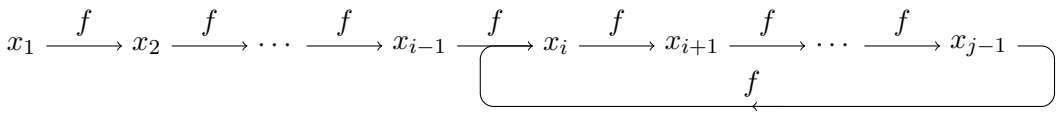
Pas je de functie f toe op een getal x_1, en dan steeds weer opnieuw, dan bekom je de rij $x_1, x_2, x_3, x_4, \ldots$, met $x_2 = f(x_1)$, $x_3 = f(x_2) = f(f(x_1))$, $x_4 = f(x_3) = f(f(f(x_1)))$, ... Met andere woorden, $x_{i+1} = f(x_i)$ voor alle $i \geq 0$.

Bijvoorbeeld, nemen we $x_1 = 1$, dan krijgen we de volgende rij :

$$x_1, \quad x_2, \quad x_3, \quad \ldots \quad = \quad 1, \quad 6, \quad 3, \quad 46, \quad 23, \quad 2646, \quad 1323, \quad 8751642, \quad \ldots$$

(Probeer dit voorbeeld uit met een klein programmaatje om te testen of je de uitleg hierboven wel goed begrepen hebt.)

Vanaf een bepaald punt zal de rij getallen zich beginnen herhalen (maar dit kan wel heel lang duren). Er bestaat dus een index i en een index $j > i$ zodat $x_i = x_j$. We illustreren dit in onderstaand diagram :



Het aantal getallen in de resulterende lus noemt men de *periode* van deze rij. Met onze notatie is die periode gelijk aan $j - i$, als i en j ($> i$) tenminste de kleinste indices zijn waarvoor $x_j = x_i$.

We geven enkele voorbeelden. In onderstaande rij is de periode gelijk aan 3 :

$$10, \quad \underline{100, \quad 33, \quad -2}, \quad 100, \quad 33, \quad -2, \quad 100, \quad 33, \quad -2, \quad 100, \quad 33, \quad -2, \quad 100, \quad 33, \quad -2, \quad \ldots$$

in deze rij is de periode gelijk aan 5

$$2, \quad 3, \quad \underline{7, \quad 5, \quad -8, \quad 12, \quad -3}, \quad 7, \quad 5, \quad -8, \quad 12, \quad -3, \quad 7, \quad 5, \quad -8, \quad 12, \quad -3, \quad \ldots$$

en hier is de periode gelijk aan 2

$$\underline{-14, \quad 14}, \quad -14, \quad 14, \quad -14, \quad 14, \quad -14, \quad 14, \quad -14, \quad 14, \quad -14, \quad 14, \quad -14, \quad 14, \quad \ldots.$$

De drie voorbeeldrijen hierboven hebben niets te maken met de functie f die we je gegeven hebben. Voor onze functie f zal de periode namelijk veel langer zijn, zodanig lang dat je ze niet met de hand kunt uitrekenen.

Taak

Schrijf een computerprogramma dat de periode van f bepaalt voor een gegeven startgetal x_1.

Limieten en beperkingen

Voor deze taak krijgt je programma maar een beperkt geheugen toegewezen en een beperkte uitvoeringstijd. Je kan dus geen arrays gebruiken met miljoenen elementen en je zal ook een efficiënte oplossing moeten bedenken die niet minutenlang moet rekenen om tot een resultaat te komen.

Wat er ook gebeurt, je programma zal stopgezet worden na **5 sec.** uitvoeringstijd (C, Java), **1 min.** uitvoeringstijd (Pascal), of **10 min.** uitvoeringstijd (Python, PHP). Je programma mag niet meer dan **32 MB** geheugen gebruiken.

Input

Een bestand met één lijn die het startgetal x_1 bevat. De input eindigt met een nieuwe lijn.

Output

Een bestand met één lijn die de periode bevat van de functie f voor het startgetal x_1, gevolgd door een nieuwe lijn.

Voorbeeld

Voor de volgende input :

```
254846291
```

Moet jouw programma de volgende output genereren :

```
26
```

Scores

- Test 1 (8 ptn) : $i = 1$ (de lus begint al bij het eerste getal), $j < 1000$;
- Test 2 (8 ptn) : $i = 1$ (de lus begint al bij het eerste getal) ;
- Test 3 (12 ptn) : $i < 1000$ (het startpunt van de lus bevindt zich bij de eerste 1000 getallen) ;
- Test 4 (32 ptn) : geen limieten.