be-OI 2012 **Finale** 14 maart 2012	**Gegevens invullen in HOOFDLETTERS en LEESBAAR, aub** VOORNAAM+NAAM : ..... SCHOOL : ..... LOGIN : `INSERT-LOGIN` WACHTWOORD : `INSERT-PASSWORD` ZAAL : Babbage / Boole	**Gereserveerd** SID

Belgische Olympiade in de Informatica (duur : 1u20 maximum)

Dit is de vragenlijst van het **gedeelte aan de computer** van de finale van de Belgische Olympiade in de Informatica. Er zijn 3 vragen. De eerste vraag staat niet op punten maar dient om je vertrouwd te maken met het systeem gedurende het eerste **kwartier**. De twee volgende vragen worden pas uitgedeeld nadat dit kwartier is afgelopen, en voor die vragen krijg je dan **maximum 1u20 de tijd**. Enkel de code die je via de website indient zal meetellen bij de beoordeling van dit onderdeel.

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 indienen via het indieningssysteem. Alle details daarover staan op de volgende pagina.
2. 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, school en **naam van de zaal** op de eerste pagina. 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 technische en inhoudelijke vragen mag je enkel stellen aan de organisatoren. Logistieke vragen mogen gesteld worden aan de toezichthouders.
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 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 een netwerk mag maken, en ook niet mag lezen uit of schrijven naar bestanden op de harde schijf.
7. Je mag **kladbladen** vragen aan de toezichthouders.
8. Het is strikt **verboden te eten of te drinken** in de computerlokalen. Je mag je **plaats niet verlaten tijdens de proef**, bijvoorbeeld om naar het toilet te gaan of te roken.
9. Je krijgt **exact 1 uur en 20 minuten** om deze proef op te lossen.

Succes !

Vragenlijst finale computer

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*.

Open het skeletprogramma dat je moet vervolledigen

- **Open je persoonlijke map** door te dubbelklikken op het icoon `be-oi12_final` op het bureaublad.
- Deze map `be-oi12_final` bevat één map per probleem (`task_0` tot `task_2`) en een map `docs`.
- De map `docs` bevat één map per programmeertaal, je kan er de documentatie van die taal in terugvinden.
- In de mappen van elk probleem, vind je opnieuw één map per programmeertaal. **Open de map die overeenkomt met de taal** waarin je gaat programmeren (je kan later altijd nog veranderen).
- In die map vind je een **bestand met broncode** genaamd `code.*` (de extensie hangt af van de programmeertaal), **een inputbestand** voor je programma (genaamd `input-example.txt`) en **een outputbestand** dat ermee overeenkomt (genaamd `output-expected.txt`). Het inputbestand bevat een voorbeeld van geldige input, waarvoor je programma als output de inhoud van het outputbestand moet teruggeven. Deze twee bestanden worden gebruikt om je programma te testen.
- Dubbelklik op het bestand met broncode om het te openen. Het programma *gedit* zal geopend worden om het bestand te lezen.
- Om je op weg te helpen, is een deel van de code al aanwezig. Je bent vrij om die aan te passen zoals je wenst.

Het compileren, uitvoeren en testen van je programma

- In elke map zit een uitvoerbaar bestand genaamd `oi-build.sh`. Je kan erop dubbelklikken en de optie `run in terminal` kiezen. De compilatie zal starten en je krijgt een console-venster te zien (ook *command line* genoemd) waar je het verloop ervan kan volgen.
- Als je programma met succes werd gecompileerd, zal het worden uitgevoerd met als input het bestand `input-example.txt` dat in de map aanwezig is. Je mag dat bestand uiteraard ook aanpassen. Het resultaat van je programma wordt weggeschreven naar het bestand `output-example.txt`.
- Als de uitvoering zonder problemen is verlopen, wordt het resultaat `output-example.txt` van jouw programma vergeleken met de verwachte output in het bestand `output-expected.txt`. Als je `input-example.txt` hebt aangepast, vergeet dan niet dat ook met `output-expected.txt` te doen.
- Als je programma deze voorbeeldtest succesvol afrondt, zal een bestand met de naam `tosubmit.zip` worden aangemaakt in de huidige map waar je werkt. Dit zip-bestand is nuttig voor het indienen van het programma op de website waar verdere automatische tests je score zullen bepalen.
- In alle gevallen, als een fout zich voordoet (bij het compileren of uitvoeren), zal de informatie daarover verschijnen in het *command line* venster.
- Om de *command line* te verlaten kan je gewoon op `enter` drukken.

Je programma indienen

- In de map `be-oi12_final` vindt je een link naar de website waar je moet indienen, genaamd `automate`
- Log in op deze website met de login en het paswoord dat zich op de eerste pagina van deze vragenlijst bevindt (dezelfde gegevens als je gebruikt hebt om op je computer in te loggen).
- Selecteer aan de linkerkant de optie **New Submission**.
- Je kan in een lijst de vraag kiezen waarvoor je code wil indienen.

- Klik op de knop **browse** om het zip-bestand te selecteren (aangemaakt tijdens de vorige stap) dat je wil indienen.
- Klik op de knop **OK** om je oplossing in te dienen.
- Om het resultaat van je ingediende code te bekijken, kan je de optie **show exercise** kiezen aan de linkerkant.
- Voor iedere taak krijg je een samenvatting van zijn huidige staat, een beschrijving, en een status die zegt of de taak geslaagd is in de tests (de kolom *passed*). Je kan klikken op de submissie (kolom *as part of submission*) om de details te bekijken.
- Op deze detailpagina wordt het resultaat van de tests bovenaan weergegeven : *ok* (geslaagd), *fail* (gefaald), *in evaluation* (nog in uitvoering). Andere interessante informatie die je er tegenkomt zijn met name :
  - **Data valid** : Het ingediende bestand heeft het correcte formaat.
  - **Failed to prepare evaluation** : Het bestand kon niet worden gecompileerd.
  - **Failed to perform evaluation** : De tests konden niet correct worden beëindigd.
  - **Evaluation done** : De tests werden allemaal beëindigd.
- Je kan de resultaten in detail ook bekijken door te klikken op de link *show* onder de kolom *details*. In het geval van succes krijg je enkel de beschrijving van de taak te zien. Als één of meerdere testen faalden, krijg je het aantal geslaagde tests te zien.
- Enkel het resultaat dat je programma produceert bij het uitvoeren van de automatische tests zal je score bepalen. Met de kwaliteit van je code wordt geen rekening gehouden !

De 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 anderen er recent nieuwe code hebben ingediend. Het is dus mogelijk dat aan het eind van de sessie 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 aan het eind van de wachtrij opnieuw toegevoegd.
- 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 `be-oi12_final/docs`.
- 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 in de editor. 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.

Je hebt slechts 1 uur en 20 minuten voor deze proef. Verkies een tragere maar juiste oplossing boven een ambitieuze poging die uiteindelijk niet functioneert !

Taak 0 – Sum

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. 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 – Synchronisatie

Op het Internet is het heel gebruikelijk dat verschillende servers onderling moeten gesynchroniseerd worden. Een typisch voorbeeld is de synchronisatie tussen een hoofdserver en een aantal *mirror* servers. Één van de gebruikelijke manieren is dat een mirror server een lijst krijgt van de objecten die op de hoofdserver staan : de mirror berekent dan welke van die objecten hij lokaal nog mist, voordat hij een kopieeroperatie start om die objecten alsnog binnen te halen.

Wij vragen je een algoritme te implementeren dat toelaat om te ontdekken welke elementen uit één lijst nog ontbreken in een andere lijst. Elk element heeft een uniek geheel getal als identificatienummer.

Taak

Schrijf een programma dat als input twee geordende arrays A en B van gehele positieve getallen krijgt. A heeft lengte N, B heeft lengte M. Het programma geeft als output de lijst van waarden die in A voorkomen maar niet in B. A bevat zeker alle waarden in B. De twee arrays bevatten geen dubbels. Om de maximale score te verdienen, moet je programma voor alle testen slagen met limieten en beperkingen zoals hieronder aangegeven.

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.

- $1 \leq N, M \leq 100\,000$, de grootte van de twee arrays van gehele getallen ;
- $0 \leq A_i, B_i < 1\,000\,000$, de getallen in de twee gegeven arrays.

Wat er ook gebeurt, je programma zal stopgezet worden na **1 minuut** uitvoeringstijd. Je programma mag niet meer dan **128 MB** geheugen gebruiken .

Input

De input voor je programma wordt gegeven in het volgende formaat :

- De eerste lijn bevat the twee gehele getallen N en M : dat zijn de lengtes van de arrays A en B respectievelijk.
- De volgende M lijnen bevatten elk één verschillend geheel getal : die getallen zijn de elementen van de array B, in stijgende volgorde.
- De laatste N lijnen bevatten elk één verschillend geheel getal : die getallen zijn de elementen van de array A, in stijgende volgorde.
- De input wordt afgesloten met een nieuwe lijn.

Output

Je programma schrijft als ouput alle getallen die in A zitten maar niet in B, in stijgende volgorde.

Voorbeeld

Met de volgende input voor je programma

```
6 2
4
7
1
4
6
7
10
12
```

komt de volgende output overeen :

```
1
6
10
12
```

Score

De totale score voor deze taak is verdeeld over twee soorten input, die elk meetellen voor 50% van het totaal aantal punten voor deze taak.

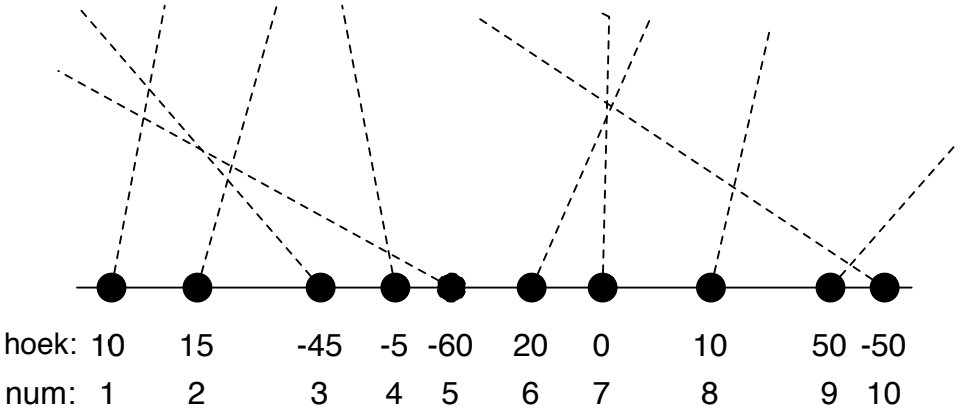
1. $0 \leq N, M \leq 200$ **(5 tests)**
2. $0 \leq N, M \leq 100\,000$ **(5 tests)**

Taak 2 – Ghostbusters

Zoals in de bekende film « Ghostbusters » maakt een leger van spoken, geesten en boosaardige demonen zich klaar om de stad New York aan te vallen. De hulp van een ploeg spokenjagers (ze zijn nu met 10) wordt ingeroepen.

Zoals iedereen weet gebruiken spokenjagers een speciaal wapen, het *proton pack* waarmee men een straal protonen afschiet die de moedwillige geest vangen. Om efficiënt te werk te gaan hebben de spokenjagers zich opgesteld op een lijn en zo proberen ze de geesten te weerstaan. Jammer genoeg heeft een demon zich meester gemaakt van de wil van de spokenjagers, en ze kunnen zich niet meer verplaatsen, en ook de richting waarin ze zullen schieten ligt helemaal vast. De spokenjagers kunnen enkel nog beslissen of ze schieten of niet.

De situatie (bekeken vanuit de hoogte) is voorgesteld in de figuur hieronder. Elk bolletje op de volle lijn is een spokenjager (hier genummerd van *1* tot *10*) : de bijhorende stippellijn geeft aan in welke richting die spokenjager schiet (indien die zou beslissen te schieten). De hoek tussen de volle lijn en de stippellijn is aangegeven onder de spokenjager. Die hoek is « 0 » als de spokenjager recht voor zich uit schiet, negatief als die meer naar links schiet, positief als die meer naar rechts schiet.



Nu hebben *proton packs* het nadeel dat onder geen enkele voorwaarde de stralen van twee packs elkaar mogen kruisen : dat zou onbeschrijflijke gevolgen hebben. Vermits de spokenjagers de schietrichting niet meer kunnen aanpassen, is de enige manier om die catastrofe te vermijden dat sommige spokenjagers niet schieten, zodat er gegarandeerd kan worden dat stralen elkaar zeker niet kruisen. Uiteraard moeten wel zoveel mogelijk spokenjagers schieten, om zoveel mogelijk geesten te neutraliseren . . .

Taak

Om toekomstige aanvallen te kunnen opvangen, wordt je gevraagd om een algoritme te schrijven dat voor een willekeurig aantal spokenjagers (en hun schiethoeken) een zo groot mogelijke deelverzameling van spokenjagers die mogen schieten teruggeeft.

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.

- Je programma krijgt als input een aantal arrays van schiethoeken,uitgedrukt in graden. Het zijn gehele getallen van -90 tot 90 (grenzen inbegrepen).
- Elke array heeft hoogstens 10 000 elementen.
- Er zijn hoogstens 10 000 arrays.

Input De input voor je programma wordt gegeven in het volgende formaat

- Op de eerste lijn staan twee getallen : eerst een getal ℓ dat aangeeft hoeveel keer je het spokenjagersprobleem moet oplossen (m.a.w. het aantal arrays met schiethoeken dat nog zal volgen), gevolgd door één spatie, gevolgd door een getal k dat het aantal spokenjagers is.
- De volgende ℓ lijnen bevatten elk k getallen gescheiden door spaties. De eerste waarde op die lijn is de schiethoek van spokenjager 1, het tweede getal de schiethoek van spokenjager 2, enzovoort.
- De input wordt afgesloten met een nieuwe lijn.

Output De output van je programma heeft het volgende formaat :

- ℓ lijnen met daarop telkens hoogstens k waarden, die de volgnummers van de spokenjagers die moeten schieten aangeven. De volgnummers staan in stijgende volgorde en worden gescheiden door één spatie. De spokenjagers zijn genummerd van 1 tot k.

Voorbeeld

De volgende input bevat 2 arrays van schiethoeken voor telkens 6 spokenjagers :

```
2 6
65 43 6 31 57 -30
-49 44 7 -72 51 54
```

Je programma produceert als output :

```
3 4 5
1 3 5 6
```