Programmeringsolympiadens final 2012

TÄVLINGSREGLER

- Tävlingen äger rum den 2 eller 6 mars. Tävlingstiden är sex timmar effektiv tid.
- Tävlingen består av sju uppgifter som samtliga ska lösas genom datorprogram. Uppgifterna ska lösas i valfritt programmeringsspråk. Du får byta språk mellan olika uppgifter.
- Tävlingsbidragen lämnas i form av programmets källkod (i en eller flera filer). Inkludera en kommentar i källkoden med den fullständiga kommandoraden för att kompilera programmet, eller om du använder ett interpreterande språk (t.ex. Python eller PHP) vilken version programmet är skrivet för. OBS! Om programmet är utvecklat i Windows-miljö ska även EXE-filen bifogas.
- I varje källkodsfil ska finnas en kommentar innehållande namn och skola.
- Lösningarna poängsätts med max 5 poäng per uppgift. Fem tester, med varierande krav hos ditt program, kommer att göras vid rättningen (undantag kan finnas). Möjlighet till delpoäng finns om programmet klarar endast en del av dessa tester. Ingen närmare bedömning av programkoden görs.
- Ingen test av indata behöver göras. Alla testdata följer de specifikationer som givits i uppgiften. Om det trots detta, vid rättningen, uppstår exekveringsfel vid körning av programmet bedöms programmet som felaktigt för det testexemplet.
- Samtliga uppgifter leder fram till program vars exekveringstid bör understiga 5 sekunder på en modern dator. Skulle exekveringstiden för ditt program överskrida denna tid bedöms programmet med 0 poäng för detta testexempel.
- Du har tillgång till de indatafiler som används i uppgiftens exempel.
- Deltagandet är individuellt vilket bland annat innebär att inget utbyte av idéer eller filer får ske under tävlingstiden. Internet eller liknande får inte användas för några ändamål.
- Hjälpmedel: Valfritt skriftligt material samt de manualfiler som är installerade på datorn. Räknedosa är tillåten.
- I flera uppgifter ska indata läsas från en vanlig textfil (se nästa sida). *Det är tillåtet att ändå läsa data från stdin, men du måste lägga en kommentar om det i källkodsfilen. Då kör vi programmet med pipning.*
- Du behöver också veta hur du gör för att hantera och skriva ut stora heltal (64-bitars tal, d.v.s. med belopp upp till 4E18) i det språk du använder.
- Tävlingsbidragen ska läggas i roten på utdelat USB-minne eller i en av läraren angiven hårddiskkatalog. Filerna ska döpas till uppg1...uppg7 med passande filtillägg. Var noga med att lämna in den korrekta versionen av ditt program.
- Tips: Det kan vara värt att göra egna indata för att testa ditt program. Även om programmet klarar testexemplen behöver det inte vara korrekt.

LATHUND FÖR INLÄSNING FRÅN FIL

Exempel på hur man i fem språk kan läsa in följande indata från filen `fil.txt`:

```
4 6
3.22 Text
```

Observera att fel kan förekomma.

C

```
#include <stdio.h>
...
int a1, a2;
char word[100];
double d;
FILE *fil=fopen("fil.txt", "rt");
fscanf(fil, "%d %d", &a1, &a2);
fscanf(fil, "%lf %s", &d, word);
```

C++

```
#include <iostream>
using namespace std;
...
int a1, a2;
char word[100];
double d;
ifstream fil("fil.txt");
fil >> a1 >> a2;
fil >> d >> word;
```

Java (J2SE)

```
import java.util.Scanner;
import java.io.File;
...
Scanner sc=null;
sc = new Scanner(new File("fil.txt"));
int a1=sc.nextInt(), a2=sc.nextInt();
double d=sc.nextDouble();
String word=sc.next();
```

Pascal

```
VAR
infile : Text;
a1, a2 : Integer;
d : Double;
word : string[100];
...
Assign(infile, 'fil.txt');
Reset(infile);
Readln(infile, a1, a2);
Readln(infile, d, word);
Close(infile);
```

Basic

```
Dim s,word As String
Dim a1, a2 As Integer
Dim d As Double
Dim sar() As String
...
Open "fil.txt" For Input As #1
Line Input #1, s
sar = Split(s," ")
a1 = val(sar(0))
a2 = val(sar(1))
Line Input #1, s
sar = Split(s," ")
d = val(sar(0))
word = sar(1)
```

Lycka till!

UPPGIFT 1 – KAPREKARS TAL

Utgå från ett tal mellan 1 och 9999 där inte alla siffror är lika. Du ska alltid tänka dig att talet har fyra siffror så fyll på med nollor om det behövs (11 tänks alltså 0011 och har därmed inte alla siffrorna lika) Sortera de fyra siffrorna i sjunkande ordning samt i stigande ordning och subtrahera dessa båda tal så får du ett nytt tal. Om du upprepar denna procedur kommer du så småningom alltid fram till talet 6174, som är det enda tal som ger sig själv. Detta tal kallas för Kaprekars tal efter den matematiker som upptäckte denna egenskap.

Skriv ett program som läser in ett tal och skriver ut den sekvens av tal som erhålls, fram till och med 6174.

Exempel 1

```
Tal ? 2687
```

Svar

```
6084
8172
7443
3996
6264
4176
6174
```

Förklaring:
8762 - 2678 = 6084
8640 - 0468 = 8172 o.s.v.

Exempel 2

```
Tal ? 7
```

Svar

```
6993
6264
4176
6174
```

UPPGIFT 2 – SKRIVMASKINEN

Författaren Paul Sheldon ska skriva en roman, men hans skrivmaskin fungerar dåligt och du ska hjälpa honom komma runt problemet så gott det går.

Problemet är att efterhand som han skriver slutar bokstäverna att fungera och skriver ut blanksteg istället. Han kan skriva varje bokstav n gånger innan den slutar fungera, utom just bokstaven "n" som är trasig redan från början. För varje onödigt blanksteg måste Paul själv för hand skriva bokstaven.

Paul vill skriva hela romanen, men vill samtidigt skriva så få bokstäver som möjligt för hand. För att åstadkomma detta inför han flexibla ord i romanen. Det är ett utav två möjliga ord, t.ex. "hej/tja", eller "orund/kantig".

Givet romanen, med alla flexibla ord, bestäm vilket av de två orden som ska väljas för varje flexibelt ord så att antalet handskrivna bokstäver minimeras.

Indata

Första raden i filen `skrivmaskinen.dat` innehåller två heltal: n, antalet som kan skrivas av varje bokstav innan den slutar fungera ($0 \leq n \leq 10000$), och m, antalet ord i romanen ($1 \leq m \leq 10000$). Andra raden innehåller romanen (endast små bokstäver a..z, mellanslag samt "/" som avgränsar flexibla ord). Det kommer finnas maximalt 20 stycken flexibla ord (d.v.s. 20 stycken /). Varje ord är mellan 1 och 30 bokstäver långt (men det kan vara längre mellan mellanslagen eftersom ett flexibelt ordpar avgränsas med / och inte med mellanslag).

Utdata

Programmet ska skriva ut det minimala antalet bokstäver som måste skrivas manuellt.

Exempel 1

```
3 7
paul sheldon e/var protagonisten i romanen/boken lida
```

Svar

```
6
```

Kommentar: Skrivmaskinen skriver: `paul sheldo  e protago iste  i b k  lida`

Exempel 2

```
5 15
i romanen som/vilken skrevs/gjordes av stephen king fick sheldon/paul
faktiskt/verkligen en skrivmaskin med trasigt/defekt n
```

Svar

```
14
```

UPPGIFT 3 – CHOKLADKULOR

Kalle älskar choklad, speciellt en sorts chokladkulor som finns i tre färger: mörk, ljus och vit. När han äter sådana är han noga med att aldrig ta två kulor av samma färg efter varandra. En dag började han fundera på hur många olika sätt han skulle kunna äta dem, givet att han har ett visst antal av varje sort, att han inte äter två av samma sort i följd och att han (självklart) vill äta upp alla kulorna.

Skriv ett program som hjälper Kalle att räkna ut detta. Han har (tyvärr) aldrig mer än 20 av varje sort, så svaret överstiger aldrig 10^{17}.

Körningsexempel 1

```
Antal mörka ? 1
Antal ljusa ? 1
Antal vita  ? 2

Det finns 6 sätt.
```

Kommentar: Ordningen kan vara MVLV, LVMV, VMLV, VLMV, VLVM eller VMVL.

Körningsexempel 2

```
Antal mörka ? 6
Antal ljusa ? 2
Antal vita  ? 2

Det finns 0 sätt.
```

Kommentar: Det går inte att äta kulorna utan att ta två mörka i rad.

Körningsexempel 3

```
Antal mörka ? 8
Antal ljusa ? 0
Antal vita  ? 7

Det finns 1 sätt.
```

Körningsexempel 4

```
Antal mörka ? 2
Antal ljusa ? 4
Antal vita  ? 3

Det finns 79 sätt.
```

Körningsexempel 5

```
Antal mörka ? 11
Antal ljusa ? 12
Antal vita  ? 14

Det finns 7587605740 sätt.
```

UPPGIFT 4 – AVRUNDNING

Snåle Pål har varit i affären och fyllt sin kundvagn. Varje vara har ett pris som, uttryckt i ören, är ett godtyckligt heltal mellan 100 och 10000. När man betalar avrundas dock totalpriset till hela kronor som vanligt, observera att 50 öre alltid avrundas uppåt.

Pål har kommit på att om man lägger varorna i lämpliga grupper (där en grupp också kan vara en ensam vara) och betalar varje grupp separat kan man möjligen spara några kronor. Skriv ett program som beräknar det minsta totalpriset han måste betala för alla varorna om han fördelar dem optimalt. Han köper aldrig mer än 30 varor, och priserna på dessa är dessutom så jämnt fördelade att högst 15 av dem har slutsiffror i intervallet 00-49 och högst 15 har slutsiffror i intervallet 50-99.

Indata

På första raden i filen `avrundning.dat` står ett heltal N, antalet varor. På andra raden står N heltal, priset i ören för varje vara.

Utdata

Programmet ska skriva ut det lägsta möjliga totalbeloppet som Pål måste betala.

Exempel 1

```
5
284 390 182 278 259
```

Svar

```
13
```

Kommentar: Pål bör först betala varorna 2 och 5 ($6.49 \approx 6$ kr) och sedan varorna 1, 3 och 4 ($7.44 \approx 7$ kr).

Exempel 2

```
10
837 371 284 1183 192 209 193 273 109 302
```

Svar

```
38
```

Fotnot för framtida läsare: Ören är en gammal myntenhet som användes mellan 1522 och 2010 och utgjorde en hundradels krona.

UPPGIFT 5 – FYRTORNEN

Vi har ett kvadratiskt rutnät på $n \times n$ rutor. Det finns m stycken fyrtorn utplacerade. Till en början är alla "släckta", men när du tänder ett av dem så tänds samtliga andra fyrtorn som har en x- eller y- koordinat gemensam med det redan tända tornet, och detta fortsätter sedan rekursivt. Antag nu att du själv får tända k torn, och att du ska tända tornen så att så många som möjligt lyser på slutet. Hur många torn kan du få att lysa?

Indata

På första raden i filen `fyrtornen.dat` står heltalen n, m och k, där $1 \leq n \leq 300000$, $1 \leq m \leq 500000$ och $1 \leq k \leq m$. Därefter följer m rader med koordinaterna (x_i, y_i) för varje fyrtorn, som uppfyller $0 \leq x_i, y_i < n$. Du kan inte förutsätta att fyrtornen står jämnt utspridda.

Utdata

En rad med ett heltal, hur många fyrtorn du totalt kan lysa upp.

Exempel 1

```
4 5 2
0 0
1 1
0 1
2 2
3 3
```

Svar

```
4
```

Exempel 2

```
20 15 4
19 2
6 15
18 6
18 11
0 10
9 12
17 8
14 2
17 10
3 9
12 11
5 19
0 4
13 11
1 5
```

Svar

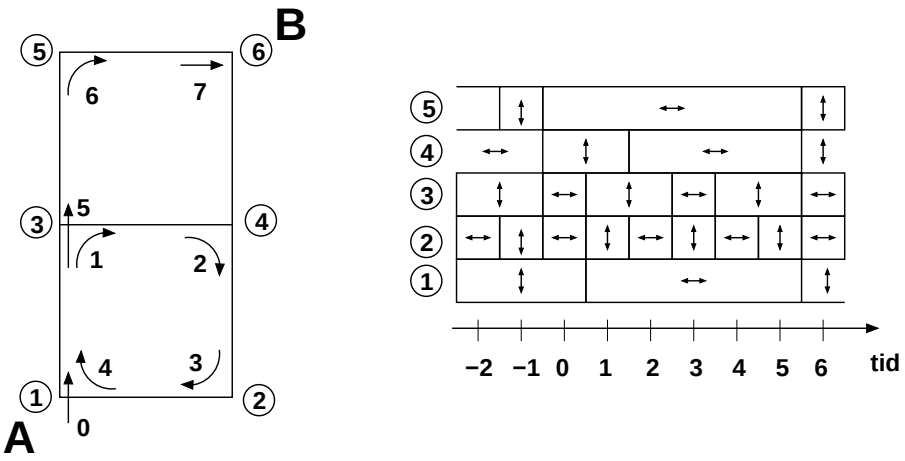
```
11
```

UPPGIFT 6 – TRAFIKLJUSEN

Estelle hatar att vänta vid trafikljus. När hon åker från sitt arbete (A) till sin bostad (B) försöker hon därför undvika långa perioder vid rödljus och tar hellre en extra omväg, förutsatt att hon fortfarande kommer hem inom 1000 tidsenheter (hon hatar nämligen också att missa sin middag). Skriv ett program som, givet fullständig information om alla stadens trafikljus, minimerar den längsta sammanhängande väntetid hon måste tillbringa vid något trafikljus, och dessutom beräknar den kortaste tid det tar att åka hela sträckan med denna minimala väntetid.

Staden består av ett rektangulärt rutnät av korsningar, där alla vägar mellan korsningarna är farbara i båda riktningarna och tar exakt en tidsenhet. I varje korsning finns trafikljus, som alla fungerar på samma sätt: Under en tidsperiod, P tidsenheter, är det grönt ljus för bilar som kommer från de vertikala (nord-sydliga) vägarna (och rött för de horisontella), därefter följer Q tidsenheter med grönt ljus för bilar som kommer från de horisontella vägarna (och rött för de vertikala). Perioderna P och Q samt tidpunkten Z som ljuset senast slog om till grönt för de vertikala vägarna kan dock variera för de olika korsningarna.

Eftersom de flesta bilolyckor sker vid vänstersvängar och U-svängar (180-graders svängar) har man beslutat att förbjuda dessa båda svängtyper. Följaktligen kan man endast köra rakt fram eller svänga till höger i en korsning. Det är inte tillåtet att stanna vid något tillfälle annat än när man har rött ljus.



FIGUR 1. Till vänster visas en karta över staden i första exemplet. De inringade siffrorna visar ordningen som korsningarna anges i filen: nerifrån och upp, från vänster till höger. Pilarna visar Estelles optimala väg och siffrorna bredvid anger vid vilken tidpunkt hon passerar korsningarna. I det här exemplet har hon grönt ljus hela vägen och behöver därför inte vänta alls. Till höger visas en tidslinje och vilken riktning som har grönt ljus för vart och ett av de fem första trafikljusen.

IndataPå första raden i filen `trafik.dat` står två tal X och Y, antalet korsningar i rutnätet ($2 \leq X, Y \leq 100$). Därefter följer $X \cdot Y$ rader, där varje rad innehåller de tre talen P,Q och Z för en viss korsning. Dessa tal uppfyller $1 \leq P \leq 10$, $1 \leq Q \leq 10$ och $-(P + Q) < Z \leq 0$. Talet Z anger alltså en av de oändligt många tidpunkter då det just slagit om till grönt ljus för de vertikala vägarna, närmare bestämt den av dessa tidpunkter som ligger närmast före tiden 0. Ordningen på korsningarna visas i figuren ovan. Vid tidpunkten 0 kommer bilen till A "nerifrån", observera att den kan behöva vänta på grönt ljus redan vid denna korsning. Resans mål är det övre högra hörnet B, där spelar trafikljuset ingen roll eftersom bilen inte ska igenom korsningen utan stanna där.

Utdata

Programmet ska skriva ut två heltal W och M. W ska vara det minsta heltalet som uppfyller att det finns ett sätt att åka från A till B på högst 1000 tidsenheter utan att vänta en sammanhängande tidsperiod längre än W tidsenheter vid rött ljus. M ska vara den minsta (tidigaste) tidpunkten som Estelle kan nå fram till B när hon gör en sådan resa (d.v.s. utan att vänta en sammanhängande tidsperiod längre än W tidsenheter vid rött ljus).

Exempel 1

```
2 3
3 5 -2
1 1 -1
2 1 -2
2 4 0
1 6 -1
10 4 0
```

Svar

```
0 7
```

Exempel 2

```
5 4
8 3 -10
1 1 0
7 5 -9
4 3 -1
3 8 -5
5 4 -7
6 8 -5
5 3 -5
2 6 0
1 2 -1
4 8 -2
6 1 -1
6 5 -3
2 1 -2
6 8 -9
1 7 0
5 3 -4
5 3 -5
1 4 0
2 2 0
```

Svar

```
4 18
```

UPPGIFT 7 – GRIDVOLLEYBOLL

Gridvolleyboll spelas med två spelare i varje lag (som beachvolley) men spelarna kan bara slå på bollen när de står på någon av de förutbestämda gridpunkterna. Ett slag går till så att en spelare som står på en "tillräckligt närliggande" gridpunkt flyttar sig till den gridpunkt där bollen befinner sig. Endast den spelare som just ska slå bollen får röra sig. Spelaren bestämmer en gridpunkt dit hen vill slå bollen, antingen på den egna planhalvan (en "passning") eller på motståndarlagets planhalva (ett "överslag"). Bollen hamnar alltid på just den tilltänkta gridpunkten och kan inte snappas upp på vägen. Om det inte finns någon spelare som kan ta sig till gridpunkten där bollen är, så förlorar det lag där bollen befinner sig denna poäng. Liksom i beachvolley får bollen slås *högst* tre gånger på en planhalva (två passningar och ett överslag), och samma person får inte slå bollen två gånger i rad. Det första slaget (serven) ska alltid ske från ett av planens hörn och måste alltid vara ett direkt överslag. Det är dessutom förutbestämt att den första spelaren (A1) i lag A ska serva.

Skriv ett program som, givet planens storlek och de fyra spelarnas egenskaper, beräknar vilket lag som vinner poängen om båda lagen spelar optimalt. Om inget av lagen kan tvinga sig till en vinst kommer bollen att pågå oändligt länge och vi räknar resultatet som oavgjort.

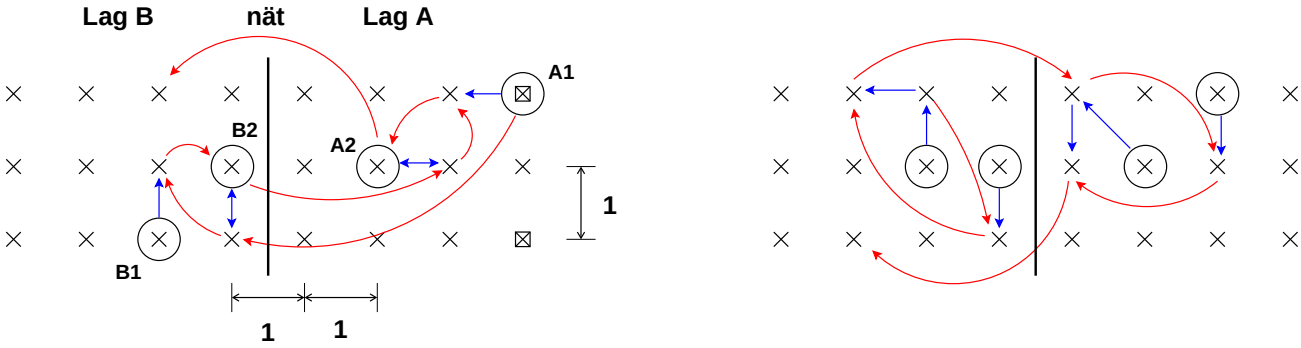
Nätet är genomskinligt, så spelarna kan se motståndarlagets placering (men kom ihåg att man inte får röra sig förrän man ska slå bollen). Spelet börjar på följande sätt: Först intar lag A sina platser: spelare A1 i ett hörn och spelare A2 på valfri plats. Därefter intar spelarna B1 och B2 sina (helt valfria) platser, medan lag A inte får flytta sig. Slutligen servar spelare A1 och spelet är igång. Du kan anta att spelarna rör sig oberoende av varandra, de kan t.o.m. befinna sig på samma gridpunkt. Däremot får de inte gå utanför planen eller in på motståndarnas planhalva.

Indata

På första raden i filen `volley.dat` står två tal $2 \leq X \leq 4$ och $2 \leq Y \leq 3$ som anger antalet gridpunkter i x-led och y-led på varje planhalva (se figuren ovan). Därefter följer fyra rader som vardera beskriver en spelare, i ordningen A1, A2, B1 och B2. Varje rad består av två heltal F och S, där $\sqrt{F}$ är det maximala avståndet spelaren kan slå bollen och $\sqrt{S}$ är det maximala avståndet spelaren kan röra sig under ett slag, d.v.s. det maximala avståndet spelaren kan befinna sig från den gridpunkt dit bollen är på väg för att kunna nå den i tid. Talen uppfyller $0 \leq F \leq 25$ och $0 \leq S \leq 5$.

Utdata

Först en rad med en sträng som anger resultatet för lag A: "Vinner", "Förlorar" eller "Oavgjort". Om lag A vinner ska programmet därefter skriva ut det minsta antalet överslag (båda lagens räknade) som krävs för att lag A garanterat ska vinna (alltid udda antal). Om lag A förlorar ska programmet istället skriva ut det minsta antalet överslag (båda lagens räknade) som krävs för att lag B garanterat ska vinna (alltid jämnt antal). Om det blir oavgjort behöver inget tal skrivas ut.



FIGUR 2. En möjlig spelsekvens i exempel 4, där båda lagen spelar optimalt, d.v.s. lag A spelar på ett sådant sätt att de garanterat vinner på högst 5 överslag medan lag B spelar på ett sådant sätt att lag A inte kan vinna på mindre än 5 överslag. Däremot finns det för båda lagen massor av likvärdiga spelsätt. Till vänster visas planens utformning med gridpunkterna markerade som kryss och de två möjliga servepunkterna markerade med kvadrater runt kryssen. Dessutom visas en utgångsställning för de fyra spelarna som är vinnande för lag A och optimal för lag B. Vidare visas bollens väg med böjda röda pilar, medan spelarnas rörelser visas med blåa raka pilar. För tydlighets skull har spelsekvensen delats upp så att den vänstra figuren visar sekvensen till och med det tredje överslaget, och den högra figuren visar resten av sekvensen, som slutar med att lag A slår över en boll som lag B omöjligt kan ta.

Bedömning: En lösning som korrekt anger resultatet men inte det minsta antalet överslag är värd 3 av 5 poäng. Observera att testfallen kommer att grupperas, så att programmet måste ge korrekt svar på en mängd testfall (med samma planstorlek) för att erhålla några delpoäng.

Exempel 1
```
2 2
8 0
3 0
1 0
2 0
```
Svar
```
Förlorar
2
```

Exempel 2
```
3 2
10 1
6 2
7 1
5 1
```
Svar
```
Oavgjort
```

Exempel 3
```
3 2
8 5
10 2
7 1
5 1
```
Svar
```
Förlorar
0
```

Exempel 4
```
4 3
20 1
15 2
15 1
15 1
```
Svar
```
Vinner
5
```

Kommentar: I exempel 1 kan ingen av spelarna röra sig. Lag B vet att serven inte når till bakre raden och kan därför sätta sina spelare på punkterna närmast nätet. Även om deras slag inte heller når bakre raden så vinner de eftersom lag A bara kan täcka upp en av punkterna vid nätet då spelare A1 är fast i hörnet efter serven.