Programmeringsolympiadens final 2011

TÄVLINGSREGLER

- Tävlingen äger rum den 4 eller 8 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 som exekverbara filer i Windows-format (EXE). Dessutom ska källkoden bifogas. För Linux/Mac-användare som använder gcc kan vi göra undantag från regeln med EXE-filer och istället kompilera källkodsfilen själva. Inkludera en kommentar i källkoden med den fullständiga kommandoraden för att kompilera programmet. Detsamma gäller om du använder ett interpreterande språk, t.ex. PHP (ange även version). Om du använder Java, så fungerar class-filen som exekverbar.
- 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. Självklart får din dator inte vara kopplad till vare sig internt eller externt nät.
- 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). **Avancerat: 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.**
- 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 – STRUMPMATCHNING

Hos familjen Svensson är det barnen som fixar tvätten idag. Lilla David har fått den allra svåraste uppgiften: att para ihop strumporna. Skriv ett program som hjälper David att bestämma vilken strumpa som ska paras ihop med vilken.

Det finns totalt N strumpor ($2 \leq N \leq 1000$), och varje strumpa har en färg F_i. Två strumpor i och j kan paras ihop om skillnaden i färg strikt understiger ett givet tal D , d.v.s. $|F_i - F_j| < D$.

Antalet strumpor kan vara udda, och strumporna kan ha vilken färg som helst (strumpor följer som bekant sina egna naturlagar och kan t.ex. försvinna spårlöst). Vidare så passar alla strumpor båda fötterna. Du ska räkna ut det maximala antalet strumppar som kan bildas, med ovan givna data.

Första raden i filen `strumpor.dat` innehåller två heltal, N och D, åtskilda med blanksteg. Sedan följer en rad med N heltal: F_1, F_2 o.s.v. till F_n. Talen F_i och D ligger mellan 1 och $1'000'000'000$ (inklusive). Programmet ska skriva ut ett heltal: maximalt antal par som kan bildas.

Exempel 1

```
5 3
3 8 1 5 9
```

Svar

```
2
```

Förklaring: Här paras 3 ihop med 5 och 8 med 9.

Exempel 2

```
4 1
100 101 102 103
```

Svar

```
0
```

Förklaring: Perfektionisten tillåter inga färgskillnader men får då gå barfota.

Exempel 3

```
10 20
81 92 42 45 62 5 4 85 73 22
```

Svar

```
4
```

UPPGIFT 2 – IPV6

Idag identifieras datorer på internet med en 32-bitars IP-adress (t ex 83.233.162.29). Antalet tillgängliga adresser håller dock på ta slut. För att råda bot på det så har IPv6 introducerats. IP-adresserna är där 128 bitar lång och kan se ut så här (med hexadecimala siffror):

```
2001:0db8:85a3:0000:0000:8a2e:0370:7334
```

Denna representation kan komprimeras genom att ta bort några eller samtliga inledande nollor i en grupp (men minst en siffra ska vara kvar). Adressen ovan kan t ex förenklas till följande:

```
2001:db8:85a3:0:00:8a2e:370:7334
```

Dessutom får en eller flera sammanhängande grupper av nollor ersättas med dubbla kolon, '::'. Då blir adressen ovan:

```
2001:db8:85a3::8a2e:370:7334
```

Dubbla kolon får endast användas på ett ställe i adressen.

Skriv ett program som läser in en giltig IPv6 adress från filen *ip.dat* (som består av en enda rad) och skriver ut en rad innehållande motsvarande adress okomprimerad.

Exempel 1
```
25:09:1985:aa:091:4846:374:bb
```

Svar
```
0025:0009:1985:00aa:0091:4846:0374:00bb
```

Exempel 2
```
::1
```

Svar
```
0000:0000:0000:0000:0000:0000:0000:0001
```

UPPGIFT 3 – POSTILJONER

Företaget Posten AB ska lägga schema för sina anställda postiljoner. Dessa jobbar alltid heltid, 4 dagar i följd. Däremot är det schemaläggarens ansvar att säga vilken veckodag postiljonen ska börja jobba. En anställd kan t.ex. jobba måndag–torsdag, en annan anställd kan jobba lördag–tisdag (det finns inget helg-begrepp i uppgiften).

Eftersom mängden post varierar, kräver varje veckodag ett visst antal anställda som är tillgängliga.

Skriv ett program som frågar efter de 7 veckodagarnas belastning och beräknar det minsta antalet postiljoner som måste vara anställda. Belastningen för en arbetsdag kommer aldrig överstiga 200.

Exempel 1

```
Måndag ? 1
Tisdag ? 2
Onsdag ? 2
Torsdag ? 2
Fredag ? 2
Lördag ? 2
Söndag ? 1

Antal postiljoner: 3
```

Förklaring: Postiljonerna kan t.ex. börja på tisdag, onsdag och lördag.

Exempel 2

```
Måndag ? 7
Tisdag ? 10
Onsdag ? 2
Torsdag ? 4
Fredag ? 5
Lördag ? 2
Söndag ? 1

Antal postiljoner: 11
```

Förklaring: Du kan t.ex. lägga 5 postiljoner på måndag, 4 på tisdag och en vardera på fredag och lördag.

Exempel 3

```
Måndag ? 48
Tisdag ? 81
Onsdag ? 75
Torsdag ? 76
Fredag ? 76
Lördag ? 59
Söndag ? 73

Antal postiljoner: 122
```

UPPGIFT 4 – OLIKHETER

På hur många sätt kan följande olikhet satisfieras?

$$\frac{A_1}{A_2} < \frac{?}{B} < \frac{?}{C} < \frac{?}{D} < \frac{E_1}{E_2}$$

där A_1, A_2, B, C, D, E_1, E_2 alla är givna heltal mellan 1 och 1000 och $?$ ska ersättas med heltal. Svaret ryms alltid i ett 64-bitars heltal.

Exempel 1

```
A1 ? 14
A2 ? 5
B  ? 1
C  ? 9
D  ? 7
E1 ? 10
E2 ? 3

Antal lösningar: 3
```

Förklaring: Här är lösningarna utskrivna:

$$\frac{14}{5} < \frac{3}{1} < \frac{28}{9} < \frac{22}{7} < \frac{10}{3}$$

$$\frac{14}{5} < \frac{3}{1} < \frac{28}{9} < \frac{23}{7} < \frac{10}{3}$$

$$\frac{14}{5} < \frac{3}{1} < \frac{29}{9} < \frac{23}{7} < \frac{10}{3}$$

Exempel 2

```
A1 ? 1
A2 ? 5
B  ? 4
C  ? 3
D  ? 2
E1 ? 5
E2 ? 1

Antal lösningar: 369
```

Exempel 3

```
A1 ? 900
A2 ? 950
B  ? 900
C  ? 950
D  ? 980
E1 ? 950
E2 ? 900

Antal lösningar: 177631
```

UPPGIFT 5 – FRIENDBOOK

FriendBook är en internetsite där man kan chatta och skriva till sina vänner. Under en lång tid har de använt sig av ett simpelt "vänsystem", varje användare har en lista över sina "vänner" bland de övriga användarna. På sistone har däremot en mycket kontroversiell feature dökt upp, nämligen att man även har en lista över sina "fiender". Medan vänrelationen alltid är ömsesidig (man bekräftar att man känner varandra) så behöver fienderelationen inte vara det: person A kan ha en fiende B som av ren fiendskap vägrar acceptera A som fiende.

Du är poet och funderar ofta över visdomsord och citat! Nyligen har du fått upp ögonen för följande citat:

Med en vän menar man en som tycker illa om samma människor som man själv.

Du vill veta i hur stor utsträckning citatet stämmer på ett givet Friendbook-nätverk. Mer formellt, för hur många par av användare gäller att de antingen är vänner och har identiska fiende-listor eller att de inte är vänner och inte har identiska fiende-listor?

Indata

På första raden i filen friendbook.dat står ett heltal N, antalet användare ($2 \leq N \leq 5000$). Sedan följer N rader, där varje rad består av N tecken. Beteckna tecknet på rad y och kolumn x för S_{yx}. S_{ij} anger vilket förhållande person i har till person j. De möjliga tecknen är 'V', 'F' och '.', de står för vän, fiende samt neutralt förhållande. Om exempelvis S_{ij} ='F' innebär det att person j finns på person i:s fiendelista. S_{ii} är alltid '.' och om S_{ij} ='V' så är även S_{ji} ='V' (men detta gäller inte alltid för de övriga tecknen).

Utdata

Programmet ska skriva ut ett heltal: antalet par av användare (utav de totalt $\frac{N(N-1)}{2}$ paren) för vilka citatet stämmer.

Exempel 1

```
3
.VV
V.V
VV.
```

Svar

```
3
```

Här är alla vänner med alla, så de har alltså alla samma fiender (trots att de inte har några fiender).

Exempel 2

```
3
.FF
F.F
FF.
```

Svar

```
3
```

Denna gång är alla fiender och då allas fiendelistor är olika stämmer citatet återigen för alla tre paren.

Exempel 3

```
5
.VFFF
V.FFF
FF.VF
FFV.F
FFFF.
```

Svar

```
10
```

De två paren som är vänner har identiska fiendelistor, övriga 8 par har olika fiendelistor.

Exempel 4

```
6
.VV.F.
V.V...
VV..FF
....VV
...V.F
FF.VF.
```

Svar

```
9
```

Här har de 5 vänparen olika fiendelistor. Men av de 10 övriga paren så har bara ett identiska fiendelistor, så 9 par uppfyller citatet.

Exempel 5

```
20
.VVVFVVVVV.VVVVVVVVV
V.VVVVVV.VVVVVVVVVV
VV.VVVFV.VVVVFVVVVV.
VVV.VVVV.VVFVFVVV.VV
FVVV..VVVVVVVVVVVVV
VVVVF.VVVV.VVVVVVFV
VV.VVV.VVV.VVV.VVVV.
VVVVVVV.VF.VVV.V..FV
VF.FVVVV.VVVVF.VV.VV
VVVVVVVFV.VVFVVFVVVV
FVVVV.FFVV.VFV.VVVVV
VVVFVVVVVVV.FFVVV.VV
VVVVVVVVVF.F.VVV.VVV
VV.FVVVV.VVFV.VVVVF.
VVVVVVFF.V.VVV.V.VVV
VVVVVVVVVFVVVVV.VVVV
VVVVVVVFVVVVFV.V.VVV
VVVFVVVF.VVFVVVVV.VV
VVVVV.VFVVVVVFVVVV.V
VV.VVVFVVVVVVFVVVVV.
```

Svar

```
37
```

UPPGIFT 6 – KINESISKA MUREN

Den kinesiska kejsaren måste försvara sitt land från de angripande mongolerna. Givet en förenklad karta över det aktuella området (ett $M \times N$ rutnät), där vissa rutor är hinder (markerade med '#'), bestäm det minsta antal rutor där en mur måste byggas som gör det omöjligt att ta sig från den övre raden i rutnätet (mongolernas område) till den nedersta raden i rutnätet (kinesiskt område). De mongoliska marktrupperna kan bara förflytta sig från en ruta till någon av sina fyra grannrutor om ingen av dessa är hinder eller mur.

Indata

Första raden i filen `muren.dat` innehåller två heltal, M och N ($3 \leq M, N \leq 1000$), storleken på rutnätet. Därefter följer N rader, var och en innehållande M tecken. Varje tecken är antingen '.' eller '#'. Den första och sista raden innehåller enbart '.' .

Utdata

Programmet ska först skriva ut en rad med ett tal, det minimala antalet rutor där det måste byggas en mur. Därefter ska det skriva N rader som visar på vilka rutor muren har byggts. Använd samma format som i indatat, men bokstaven 'M' på de rutor där mur har byggts. Det är tillåtet att bygga mur även på första och sista raden.

Exempel 1

```
5 4
.....
..#..
.#..#
.....
```

Svar

```
2
.....
..#..
M#.M#
.....
```

Exempel 2

```
12 9
............
....#.......
........#...
.##..#......
.....#......
....##..#...
............
.#....#....#
............
```

Svar

```
6
............
....#.......
........#...
M##..#......
...M.#......
....##MM#M..
..........M.
.#....#....#
............
```

UPPGIFT 7 – EKOKÖRNING

De flesta bilister vet att nyckeln till låg bränsleförbrukning är att köra med jämn och låg fart. Tyvärr tillåter ofta inte vägen att man kör helt jämnt, det finns dels skiftande hastighetsbegränsningar, dels platser där farten begränsas naturligt, t.ex. när man svänger. Dessutom har man ofta någon tid att passa, så man kan inte köra hur långsamt som helst.

Skriv ett program som, givet sådana begränsningar, beräknar den minimala bränsleförbrukningen för en viss vägsträcka.

Vägsträckan består av N hundrametersintervall ($1 \leq N \leq 100$). Vi antar att man vid start- och slutpunkten för varje intervall har hastigheter v_1 respektive v_2 km/h som *måste* vara jämnt delbara med 10. Vid varje sådan punkt är den maximala hastigheten begränsad till ett givet värde. Vidare antar vi [1] att bränsleförbrukningen (i milliliter) för intervallet är

$$f = \max\left(0.06v + 0.002(v_2^2 - v_1^2), 0\right)$$

där

$$v = \frac{v_1 + v_2}{2}$$

som vi antar är medelhastigheten för intervallet (för att slippa bry oss om exakt hur acceleration och bromsning sker inom intervallet). Tiden i sekunder det tar att köra intervallet blir då förstås

$$t = \frac{360}{v}$$

Den totala tiden för hela sträckan får inte överskrida T sekunder ($1 \leq T \leq 5000$). Vid sträckans början har du hastigheten 0. Vi bortser från begränsningar i bilens förmåga att accelerera och bromsa. Observera att du inte kan hålla hastigheten 0 i ett helt intervall (då kommer du aldrig framåt), däremot kan du vid en viss position komma ner till hastigheten 0.

[1] Detta är en grov modell men för den intresserade följer här en motivering: Den första termen beror på att många krafter (luftmotstånd, friktion m.m.) är ungefär proportionella mot bilens hastighet. Konstantens storlek är naturligtvis beroende av bilmodell etc. men 0.6 liter/mil vid 100 km/h är en rimlig siffra. I verkligheten spelar också växlarnas lägen och motorns verkningsgrad vid olika varv in, så förhållandet är sublinjärt. Den andra termen är ökningen av bilens rörelseenergi vid acceleration, beräknat för vikten 1660 kg och energivärdet 32 MJ/liter för bensin. I idealfallet får man tillbaka hela denna rörelseenergi som minskad förbrukning när man bromsar (därför har vi samma formel även för $v_2 < v_1$), men man kan naturligtvis aldrig få negativ förbrukning.

Indata

Första raden i filen `eko.dat` innehåller talen N och T, separerade med blanksteg. Därefter följer en rad med N blankstegsseparerade tal som anger maxhastigheten i slutet av varje intervall (maxhastigheten för startpunkten är irrelevant eftersom vi har hastigheten 0). Talen är mellan 0 och 120 (inklusive) och jämnt delbara med 10. Det första talet är aldrig 0 och två på varandra följande tal är aldrig 0.

Utdata

Programmet ska skriva ut den minimala bränsleförbrukningen i milliliter, d.v.s. summan av f för de N intervallen när du kör optimalt och ändå kommer fram senast vid tiden T sekunder (för givna testdata är det alltid möjligt att komma fram i tid). Svaret ska vara exakt angivet (men formatet spelar ingen roll).

Exempel 1

```
2 40
70 70
```

Svar

```
3.2
```

Förklaring: På första delsträckan accelererar du till 30 km/h och förbrukar 2.7 ml. På andra delsträckan bromsar du till 20 km/h och förbrukar 0.5 ml. Den totala tiden är 38.4 sekunder.

Exempel 2

```
8 61
50 80 80 80 50 50 100 0
```

Svar

```
33
```

Förklaring: På första "landsvägen" är det värt att accelerera upp till maxhastigheten, men efter 50-sträckan är det inte lönt. Din hastighet efter varje intervall är: 50, 80, 80, 70, 50, 50, 60, 0

Exempel 3

```
15 218
120 0 120 0 10 10 10 50 50 60 70 80 90 90 90
```

Svar

```
104.6
```