Programmeringsolympiaden 2008
Kvalificering

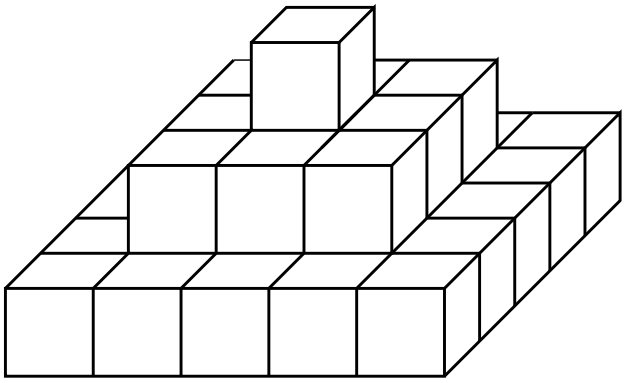
TÄVLINGSREGLER

- Tävlingen äger rum på ett av skolan bestämt datum under sex timmar effektiv tid.
- Tävlingen består av sex uppgifter som samtliga ska lösas genom ett datorprogram.
- Uppgifterna ska lösas i valfritt programmeringsspråk. Vi rekommenderar något av språken C, C++, Pascal (Delphi), eller Java. Du får till och med byta språk mellan olika uppgifter.
- Dina lösningar kommer att testköras med förpreparerade indata. Klarar ditt program alla testerna får du 2 poäng för uppgiften. Delpoäng (1 poäng) kan komma att utdelas då programmet inte klarar alla testerna. Ingen närmare bedömning av programkoden görs.
- Samtliga uppgifter leder fram till program vars exekveringstid normalt bör understiga 1 sekund. Skulle en lösning leda fram till ett program vars exekveringstid överstiger 10 sekunder bedöms programmet för det testexemplet som felaktigt.
- Rättningen utförs på samma eller likvärdig dator. Om, vid rättningen, fel uppstår vid kompilering bedöms programmet direkt som felaktigt och lösningen ger 0 poäng.
- 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.
- Deltagandet är individuellt vilket bland annat innebär att inget utbyte av idéer eller filer får ske under tävlingen.
- Hjälpmedel: Programspråkens manualer, minst ett exemplar tillgängligt i datorsalen. Formelsamling och räknedosa för varje deltagare.
- Tävlingsbidraget ska lämnas in i form av källkodsfiler som läggs i roten på utdelad diskett eller i en av läraren angiven hårddiskkatalog. Filerna ska döpas till uppg1...uppg6 med passande filtillägg. Ingen hänsyn tas till andra filer. Var noga med att lämna in den korrekta versionen av ditt program.

Årets International Olympiad in Informatics anordnas i Egypten i augusti. Kanske blir du en av dem som representerar Sverige där.

Lycka till!

UPPGIFT 1 – PYRAMIDBYGGE



FIGUR 1. *Denna pyramid med höjden 3 innehåller 35 stenblock. Cheops-pyramiden i Egypten har höjden 210.*

När man ska inleda ett större projekt, exempelvis bygga en pyramid, är det bäst att tänka efter en gång extra. Du ska skriva ett program som beräknar hur hög pyramid man kan bygga om man har tillgång till ett visst antal stenblock.

Vi antar att pyramiden är kompakt, d.v.s. det finns inga hålrum inuti. Vidare byggs den enligt principen i figur 1. Varje lager är alltså kvadratiskt med en sidlängd som är två block mindre än det underliggande lagrets. Det översta lagret består alltid av ett ensamt block.

Programmet ska fråga efter antalet tillgängliga block (högst hundra miljoner) och skriva ut höjden (i block räknat) för den största pyramid som kan byggas. Det gör ingenting om det blir block över, men det får inte saknas ett enda block.

Körningsexempel:

```
Antal block ? 83
Höjd: 3
```

UPPGIFT 2 – SFINX-SEX

En sfinx är en varelse med ett lejons kropp och en människas huvud. Liknande kombinationer återfinns i andra kulturer, t.ex. faunen (get+människa) och gripen (örn+lejon). Man skulle kunna tro att den ringa förekomsten av sådana arter i naturen beror på viss svårighet i fortplantningen, men i denna uppgift antar vi raka motsatsen.

Med hänvisning till genetikens lagar antar vi att när två kombinationsvarelser parar sig så ärver avkomman framkroppen från den ena föräldern och bakkroppen från den andra. En grodmus som parar sig med en fiskhöna kan alltså ge upphov till antingen en grodhöna eller en fiskmus.

Skriv ett program som, givet en grupp kombinationsvarelser (somliga hannar och andra honor) beräknar antalet *olika* arter som kan existera i nästa generation om alla par av hanne-hona antas få ungar. Svaret ska också inkludera de arter som ingår i den ursprungliga gruppen.

Programmet ska fråga efter antalet hannar och vilken art varje hanne tillhör, samt likadant för honorna. Arten anges som en sträng med två bokstäver (valda bland A-Z), där första bokstaven beskriver framkroppen och andra bokstaven bakkroppen (t.ex. ML för människolejon). Programmet ska skriva ut det totala antalet olika arter som kan finnas när parning har skett. Observera att ordningen på bokstäverna spelar roll – MF och FM är olika arter.

Körningsexempel:

```
Antal hannar ? 2
Hanne 1 ? ML
Hanne 2 ? FG
Antal honor  ? 3
Hona 1 ? VM
Hona 2 ? FM
Hona 3 ? VF
Antal arter: 11
```

Kommentar: Förutom de fem arterna i ursprungsgruppen kan följande arter ha uppkommit: FF, FL, MF, MM, VG och VL.

UPPGIFT 3 – BEVATTNING

En bevattningskanal avledd från Nilen förser ett antal åkrar med vatten. Efter att i åratal ha trätit om hur mycket vatten varje åker ska få under torkperioden, beslutar sig bönderna för att samarbeta och istället minimera totalförlusten orsakad av torka. För varje åker vet man följande:

- Den *optimala volymen* vatten (det är ingen idé att överskrida denna).
- *Känsligheten*, d.v.s. hur stor ekonomisk förlust som görs för varje volymenhet vatten under den optimala volymen.
- Den *minimala volymen* vatten för att åkern alls ska ge skörd.

Bönderna vill fördela vattnet så att den sammanlagda förlusten för alla åkrarna blir så liten som möjligt, samtidigt som varje åker ska ge åtminstone någon skörd. Skriv ett program som beräknar den minimala förlusten.

Programmet ska fråga efter den totala volymen tillgängligt vatten, antalet åkrar (högst 5), och sedan för varje åker efter optimala volymen, känsligheten och minimala volymen. Alla tal ligger i intervallet 1..1000. Programmet ska skriva ut den minimala sammanlagda förlusten. Vattnet kan endast fördelas i hela volymenheter. Allt vatten behöver inte nödvändigtvis användas. I samtliga testfall finns det tillräckligt mycket vatten för att varje åker ska ge skörd.

Körningsexempel:

```
Totalvolym ? 106
Antal åkrar ? 2
Åker 1, optimal volym ? 54
Åker 1, känslighet    ? 4
Åker 1, minimal volym ? 41
Åker 2, optimal volym ? 75
Åker 2, känslighet    ? 6
Åker 2, minimal volym ? 56
Minimal förlust: 112
```

Kommentar: Den bästa fördelningen är 41 volymenheter till åker 1 och 65 volymenheter till åker 2. Detta ger en förlust på $13 * 4 + 10 * 6 = 112$.

UPPGIFT 4 – RÄKNA MED BRÅK

I egyptisk matematik hade de så kallade heltalsreciprokerna

$$\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$$

särskild betydelse. Övriga bråk skrevs som summor av dessa tal, t.ex.

$$\frac{21}{20} = \frac{1}{2} + \frac{1}{4} + \frac{1}{5} + \frac{1}{10}$$

Skriv ett program som tar emot ett bråk (förkortat så långt som möjligt) och avgör om det kan skrivas som en sådan summa, under villkoren att bara de tio första heltalsreciprokerna (alltså t.o.m. $1/10$) får användas, och att varje tal endast får användas en gång. Om det går ska programmet skriva ut de ingående termerna i summan, annars ska det skriva ut *Omöjligt*. Om det finns flera lösningar ska programmet skriva ut vilken som helst av dem.

Två körningsexempel:

```
Täljare ? 67
Nämnare ? 60
Termer: 1/2  1/4  1/5  1/6

Täljare ? 2
Nämnare ? 5
Omöjligt
```

UPPGIFT 5 – GRAVPLUNDRING

Gravplundraren Rolf har hittat en bortglömd faraonisk grav som han tänker plundra. I graven finns en mängd föremål med varierande vikt och av varierande värde. Rolf vill naturligtvis få med sig en samling med så stort värde som möjligt. Kruxet är bara att han endast har två dromedarer att lasta skatterna på. Varje dromedar kan bära högst 150 kilo. Skriv ett program som, givet vikten och värdet för varje föremål, bestämmer det maximala värdet Rolf kan forsla bort på de två dromedarerna. Inget av föremålen går att dela upp mellan de två djuren.

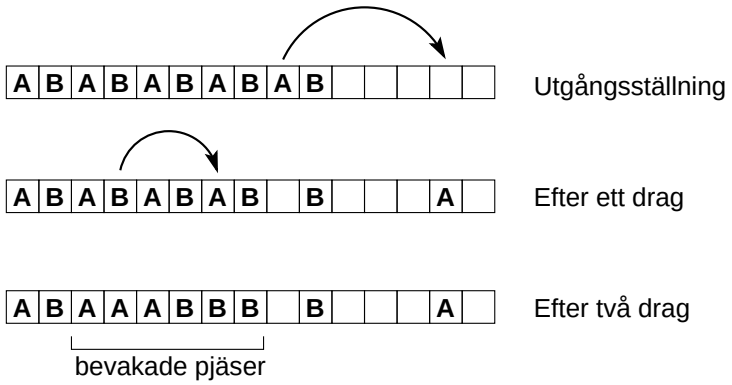
Programmet ska fråga efter antalet föremål N (där $1 \leq N \leq 14$), samt vikten M och värdet V för varje föremål. Både M och V är heltal och uppfyller $1 \leq M \leq 150$, $1 \leq V \leq 1000$). Programmet ska skriva ut det maximala sammanlagda värdet av de föremål som Rolf kan lasta på de två dromedarerna.

Körningsexempel:

```
Antal ? 3
Föremål 1, vikt ? 70
Föremål 1, värde ? 180
Föremål 2, vikt ? 90
Föremål 2, värde ? 150
Föremål 3, vikt ? 100
Föremål 3, värde ? 200
Maximalt totalvärde: 380
```

UPPGIFT 6 – SENET

Senet är ett uråldrigt egyptiskt brädspel som hittats i otaliga gravar. Reglerna är okända men en av de vanligaste rekonstruktionerna beskrivs här. Spelplanen kan betraktas som en serie med rutor där vi i denna uppgift endast bryr oss om de första 15. De två spelarna, som vi kallar A och B, har vardera 5 pjäser som vid spelets början alltid står uppställda enligt översta schemat i figur 2. Spelarna turas om att göra drag, A har första draget. I varje drag kastar spelaren några stickor, vilka tillsammans utgör en slags tärning som kan anta värdena 1, 2, 3, 4 eller 5. Därefter väljer spelaren en av sina pjäser och flyttar den framåt med så många steg som tärningen visade. För att vara ett godkänt drag måste pjäsen sluta på en ruta som antingen är tom eller innehåller en *obevakad* motståndarpjäs. I det senare fallet flyttas moståndarpjäsen tillbaka till den ruta varifrån den flyttade pjäsen kom, så att pjäserna i praktiken byter plats. En pjäs är obevakad om det *inte* finns en likadan pjäs i någon av de två angränsande rutorna.



FIGUR 2. *Utgångsställningen samt ställningarna efter dragen i exemplet nedan.*

Du ska skriva ett program som givet pjäsernas nuvarande positioner bestämmer det minimala antalet drag N som har spelats, samt en möjlig sådan dragsekvens.

Programmet ska fråga efter en sträng med längd 15, där varje tecken beskriver en ruta på spelplanen: A för en pjäs tillhörande spelare A, B för en pjäs tillhörande spelare B och . (punkt) för en tom ruta. Det finns alltid fem A:n, fem B:n och fem punkter i strängen.

Programmet ska skriva ut den kortaste dragsekvensen som leder från utgångsställningen till den inmatade ställningen. Närmare bestämt ska programmet skriva ut en rad med N tal i intervallet 1..5, där varje tal anger hur många steg som har tagits i respektive drag (första talet indikerar första draget etc.). Vilken pjäs som flyttats i varje drag behöver inte anges. Om det finns flera sekvenser med minimal längd kan programmet ange vilken som helst av dem. I samtliga testfall är $N \leq 6$. Notera att N kan vara udda, d.v.s. A kan ha gjort ett drag mer än B.

Körningsexempel:

```
Ställning ? ABAAABBB.B...A.
Dragsekvens: 5 3
```