

UPPGIFT 1 – TOMATER



FIGUR 1.

Ett intressant faktum är att omogna tomater mognar snabbare om man lägger in några redan mogna tomater bland dem. I denna uppgift ska du simulera denna process och räkna ut hur många tomater som är mogna efter en viss tid.

Antag att n tomater ligger i en lång rad och är numrerade från 1 till n . Tre av dessa tomater, nummer t_1 , t_2 och t_3 , är redan mogna när simuleringen startar vid dag 0. Varje dag mognar de tomater som ligger precis intill en redan mogen tomat. Efter dag 1 har alltså grannarna till de tre första mogna tomaterna mognat, efter dag 2 har även grannarna till de som mognade under dag 1 mognat och så vidare.

Skriv ett program som frågar efter antalet tomater n , $3 \leq n \leq 100$ numren t_1 , t_2 och t_3 (alla olika och i intervallet $1 \dots n$) på de tre tomater som är mogna från början samt en tid d , $1 \leq d \leq 100$. Programmet ska skriva ut antalet mogna tomater efter d dagar. Körningsexempel:

```
Antal tomater ? 12
Mogen ? 4
Mogen ? 5
Mogen ? 8
Tid ? 2
Antal mogna tomater: 9
```

Kommentar: Nummer 1, 11 och 12 är ännu omogna.

UPPGIFT 2 – TVÅVALSFRÅGORNA

Adam	j	j	n	j	j	n	n	n	j	n	2
Bertil	n	n	n	j	n	n	j	j	j	j	8
Caesar	n	n	j	n	n	n	j	n	j	j	5

I tabellen ovan ser vi resultatet från ett prov som Adam, Bertil, Caesar gjort. Just det här provet innehöll 10 frågor som alla skulle besvaras med *ja* (j) eller *nej* (n). Till höger i tabellen kan vi också se hur många rätt de fick. När läraren skulle rätta samma prov året därpå visade det sig att facit hade förkommit.

Skriv ett program som från givna svar och antalet rätt tar reda på svaren och återskapar facit. Ett körningsexempel:

```

Antal deltagare ? 3
Antal frågor ? 10
Deltagare 1 Svar ? jjnjnnjn
Deltagare 1 Antal rätt ? 2
Deltagare 2 Svar ? nnnjnnjjj
Deltagare 2 Antal rätt ? 8
Deltagare 3 Svar ? nnjnnjnjj
Deltagare 3 Antal rätt ? 5
Facit till provet: nnnjnjjnj
    
```

Kommentarer: Antalet deltagare $d, 3 \leq d \leq 5$. Antal frågor $f, 6 \leq f \leq 12$. Om det finns flera korrekta svar på uppgiften räcker det att ditt program ger ett av dessa. De enda svar som en elev kan avge på en fråga är ett j eller n.

UPPGIFT 3 – GRÄSKLIPPNING

Det finns nu automatiska gräsklippare ute på marknaden. Dessa rör sig slumpvis över gräsmattan, vars gräns har markerats med en nedgrävd kabel. Du ska skriva ett program som uppskattar hur lång tid det i genomsnitt tar innan hela gräsmattan är klippt.

För att definiera uppgiften precis får vi göra en rad mer eller mindre realistiska antaganden. Gräsmattan är rektangulär med längden l meter och bredden b meter, heltal där $1 \leq b \leq l \leq 10$. Vi tänker oss också en indelning i kvadratmeterstora rutor, $l \cdot b$ stycken. Vi antar att gräsklipparen klipper en sådan ruta fullständigt och sedan rullar till en av de fyra angränsande rutorna, slumpvis utvald. Det tar en minut att klippa en ruta, oavsett om rutan har klippts innan eller inte, men ingen tid att förflytta sig mellan rutorna. Om gräsklipparen försöker gå utanför gräsmattans utsträckning, hindras den så att förflyttningen aldrig utförs. Istället väljs på nytt slumpvis en av de fyra angränsande rutorna, och detta upprepas tills den väljer en ruta som ligger innanför gräsmattans gräns. Inte heller denna procedur tar någon tid i anspråk. Gräsklipparen startar alltid i en hörnruta och denna ruta klipps under första minuten. Gräsklippningen betraktas som klar när alla rutor har klippts. Detta måste alltså ta minst $l \cdot b$ minuter, men i praktiken tar det längre tid eftersom gräsklipparen inte har något minne, utan kommer att klippa samma ruta flera gånger.

Klippningen kan ta olika lång tid beroende på hur slumpen får gräsklipparen att röra sig. Vad man som gräsmattsägare är mest intresserad av är, hur lång tid det tar i genomsnitt. Därför ska ditt program utföra gräsklippningen 10000 gånger (med en ny oklippt gräsmatta varje gång) och beräkna ett medelvärde. Om dessa klippningar utförs efter varandra (i samma program) kommer datorns slumpgenerator att automatiskt börja på olika värden och därmed kommer gräsklipparen att röra sig på helt olika sätt varje gång.

Programmet ska fråga efter l och b och sedan skriva ut antal minuter som i genomsnitt går åt tills varje ruta har klippts minst en gång. 10000 klippningar är tillräckligt många för att felet normalt ska bli under 1%, men för säkerhets skull ger vi rätt för en avvikelse på upp till 10% (eller upp till fem minuter även om det är över 10%) från det korrekta svaret.

Två körningsexempel:

L? 4

B? 3

Det tar 53 minuter.

L? 6

B? 6

Det tar 294 minuter.

Hjälp: För att generera ett slumptal i intervallet $0 \dots 3$ kan du använda följande funktioner:

C: `r=rand()%4;`

Pascal: `r:=Random(4);`

Basic: `r=INT(RND*4)`

UPPGIFT 4 – NAMN

Enligt mycket opålitliga källor var det ytterst nära att alla svenskar födda 1986 fick vara del av ett namngivningsprojekt initierat av Riksförsäkringsverket. Man hade helt enkelt tröttnat på allt krångel med långa namn, folk som hade samma namn och andra tråkigheter. Istället konstruerade man ett sätt att namnge personer efter deras personnummer. På detta sätt skulle var och en få ett kort och unikt namn.

Man valde ett enkelt system för att "översätta" personnumret till namn. Man gjorde en lista över alla bokstavssekvenser av typen KVKKV, där K är en godtycklig konsonant och V är en godtycklig vokal. I en annan lista skrev man upp alla möjliga personnummer för personer födda 1986, det vill säga alla 365 möjliga födelsedatum följt av alla tal mellan 000 och 999. Man utelämnade sista siffran eftersom detta är en kontrollsiffra och därför inte behövs för att ge ett unikt nummer. Sedan sorterade man namnlistan i bokstavsordning och personnumren i stigande ordning. Början av listorna ser då ut så här:

BABBA	860101-000
BABBE	860101-001
BABBI	860101-002
BABBO	860101-003

Givet ett personnummer, ger man nu personen det namn som står på samma plats i namnlistan som personnumret står i nummerlistan. Du ska skriva ett program som, givet ett personnummer med sista siffran utelämnad och utan bindestreck, ger namnet på personen. Två körningsexempel:

Personnummer: 860817313
Namn: KALLE

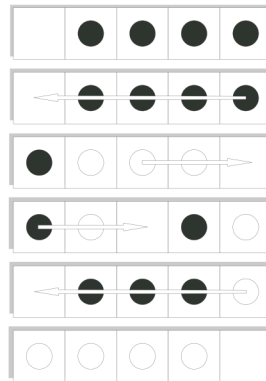
Personnummer: 861231999
Namn: PIKTU

Av det sista exemplet framgår att det finns fler namn än möjliga personnummer, följaktligen kommer inget namn som står efter PIKTU i bokstavsordning att användas.

Observera följande:

- Vokaler är AEIOUYÅÄÖ (9 stycken, i denna ordning)
- Konsonanter är: BCDFGHJKLMNPQRSTVWXZ (20 stycken, i denna ordning, W alltså inkluderat)
- Antalet dagar i varje månad är som bekant: 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31

UPPGIFT 5 – VÄNDA



FIGUR 2.

Högst upp i figur 2 ser vi ett av många möjliga utgångslägen i pusslet vi här kallar *Vända*. Vi ser fyra svarta brickor och en tom plats. De fyra brickorna är alla svarta på ena sidan och vita på den andra.

Målet. Att vända alla brickor så att de vita sidorna kommer upp. Visas längst ned i figuren.

Regler. En bricka, vilken som helst, *utom de som ligger närmast den tomma platsen*, får flyttas till den tomma platsen. Samtliga brickor som passerar på denna brickas väg dit, vänds så att brickorna ändrar färg – från svart till vit eller från vit till svart. Brickan som flyttas vänds inte. I figuren kan du följa de fyra drag som för just denna uppställning leder till målet. Dessutom på snabbaste sätt.

Skriv ett program som tar emot uppgift om brädets storlek n , i rutor räknat, $5 \leq n \leq 7$ och var den tomma rutan är belägen, ett tal $1 \dots n$, från vänster räknat. Samtliga rutor utom en innehåller alltså från start en svart bricka. Ditt program ska sedan bestämma det minsta antalet drag som krävs för att nå målet.

```

Brädets storlek ? 5
Den tomma rutan ? 1
Det krävs minst 4 drag
    
```

UPPGIFT 6 – CD-BRÄNNAREN

Kristin ska bränna ner innehållet på sin hårddisk på ett antal CD-skivor. Innehållet är ordnat i ett antal kataloger och hon vill inte splittra innehållet i en katalog över flera CD-skivor. Du ska skriva ett program som, givet storleken på varje katalog (i megabyte), beräknar hur Kristin ska fördela katalogerna så att det går åt så få CD-skivor som möjligt. Varje CD-skiva rymmer maximalt 650 megabyte.

Programmet ska fråga efter antalet kataloger n på hårddisken (där $1 \leq n \leq 15$). Sedan ska det fråga efter storleken på var och en av de n katalogerna. Storleken på en katalog är alltid ett heltal och ligger i intervallet $1 \dots 650$. Programmet ska beräkna och skriva ut det minsta antalet CD-skivor k , som behövs. Därefter ska det skriva ut ett möjligt sätt att fördela katalogerna på k skivor, genom att på var och en av k rader skriva storlekarna på de kataloger som läggs på respektive skiva, åtskilda med blanksteg. Ordningen på kataloger eller skivor spelar ingen roll. Körningsexempel:

```
Antal kataloger ? 6
Storlek på katalog 1 ? 25
Storlek på katalog 2 ? 500
Storlek på katalog 3 ? 600
Storlek på katalog 4 ? 110
Storlek på katalog 5 ? 40
Storlek på katalog 6 ? 25
Minimalt antal skivor: 2
25 600 25
500 110 40
```