UPPGIFT 1 – FORTSÄTT TALFÖLJDEN

Att fortsätta en påbörjad talföljd är en vanlig sorts uppgift i såväl matteböcker som IQ-tester. Men det smartaste måste väl ändå vara att skriva ett datorprogram som löser problemet en gång för alla? Betrakta till exempel de så kallade *triangeltalen*:

$$1, 3, 6, 10, 15, 21 \ldots$$

Ett mönster framträder om man tittar på skillnaden mellan varje tal och dess föregångare:

$$2, 3, 4, 5, 6 \ldots$$

Tydligen ökar skillnaden med 1 i varje steg. Men för någon annan talföljd kan skillnaden till exempel öka med 13, minska med 2 eller vara *konstant*. Det enda som är säkert om de talföljder som förekommer i denna uppgift är att *skillnaden mellan två intill varandra stående tal ändras lika mycket i varje steg*.

Skriv ett program som frågar efter de tre första talen i talföljden (positiva heltal mindre än 100) och sedan skriver ut de 10 första talen åtskilda med blanksteg.

Körningsexempel:

```
Första talet ? 1
Andra talet  ? 3
Tredje talet ? 6

Talföljden: 1 3 6 10 15 21 28 36 45 55
```

Här några fler talföljder att testa ditt program med:

```
1 4 9 16 25 36 49 64 81 100
2 4 6 8 10 12 14 16 18 20
3 5 3 -3 -13 -27 -45 -67 -93 -123
```

UPPGIFT 2 – TELEFONIPRISER

Vi översvämmas av erbjudanden från olika telebolag, som hävdar att just de har de bästa telefonipriserna. Vilket bolag som är billigast beror i många fall helt enkelt på hur mycket du ringer. I den här uppgiften ska du skriva ett program som tar emot information om ett antal bolags *fasta månadsavgifter* respektive *minutkostnader* och beräknar vilket bolag som är billigast, beroende på hur många minuter du pratar per månad.

Programmet ska börja med att fråga efter *antalet telebolag* (högst 5) och sedan efter den *fasta månadskostnaden i kronor* (högst 1000) samt *minutkostnaden i ören* (högst 100) för vart och ett av telebolagen. Minutkostnaden för två bolag är aldrig samma.

Programmet ska skriva ut en tabell där varje rad ger ett intervall $x - y$ och det bolag z som är billigast om man pratar mellan x och y minuter i månaden (inklusive x och y). Vi antar att samtalen debiteras i hela minuter så endast heltal x och y kan komma ifråga.

Intervallen får inte överlappa. Om två bolag skulle råka vara lika billiga (och billigare än övriga bolag) för ett visst minutantal så kan detta antal ingå i vilket som helst av de två aktuella intervallen, men alltså inte i båda. Varje bolag ska förekomma högst en gång (man kan lätt inse att om ett bolag slutar vara billigast efter något antal minuter så kan det inte bli billigast igen om antalet minuter ökar).

Intervallen ska vara sorterade i stigande ordning och det sista intervallet ska vara av formen x- vilket betecknar alla minutantal större än eller lika med x. Som du kanske redan har insett kommer det bolag som har lägst minutpris alltid att vara billigast i det sista intervallet.

Körningsexempel:

```
Antal bolag ? 5
Bolag 1, fast  ? 89
Bolag 1, minut ? 13
Bolag 2, fast  ? 70
Bolag 2, minut ? 16
Bolag 3, fast  ? 55
Bolag 3, minut ? 23
Bolag 4, fast  ? 240
Bolag 4, minut ? 0
Bolag 5, fast  ? 0
Bolag 5, minut ? 35

   0 -  368 : 5
 369 -  633 : 2
 634 - 1161 : 1
1162 -      : 4
```

Kommentar: Låt oss exempelvis kontrollera den första gränsen. Om man ringer 368 minuter per månad kostar det 128.80 med bolag 5 men 128.88 med bolag 2. Om man däremot ringer 369 minuter per månad är bolag 2 billigast (129.04 mot 129.15).

UPPGIFT 3 – RAMSAN

Ole Dole Doff
Kinke Lane Koff
Koffe Lane, Binke Bane
Ole Dole Doff
Ärtan Pärtan Piff,
Ärtan Pärtan Paff
Similimaka, Kuckelikaka
Ärtan Pärtan Poff

Ramsor av detta slag har under alla tider använts av barn för att välja ut någon slumpvis, till exempel vem som ska stå då man leker kurragömma. Barnen bildar en ring, där den som ”räknar” för varje ord i ramsan flyttar fingret, från ett barn till nästa, runt ringen. När ramsan tar slut får det utpekade barnet lämna ringen. Ramsan läses på nytt med början på det barn som står omedelbart efter det uträknade.

Denna procedur fortsätter tills endast ett barn återstår, det utvalda. Skriv ett program som frågar efter antalet ord i ramsan och antalet barn i ringen $n \leq 100$. Barnen är numrerade från 1 till n åt samma håll som räkningen sker. Nummer 1 är det barn man börjar räkna på. Programmet ska bestämma vilket barn som blir sist kvar.

Körningsexempel:

```
Antal ord ? 24
Antal barn ? 17
Barn nummer 3 blir sist kvar.
```

Kommentar: Det första barnet som åker ut är nummer 7 och därefter nummer 15, 8 och 2.

UPPGIFT 4 – MINISUDOKU

1			
			1
4			
			3

1	2	3	4
3	4	2	1
4	3	1	2
2	1	4	3

FIGUR 1.

I den här uppgiften ska du skriva ett program som löser en liten sudoku, nämligen en med 16 rutor istället för den vanliga med 81. Men vi börjar med att förklara vad en sudoku är, för dig som har lyckats undgå sudokuhysterin i tidningarna.

Vår variant av sudoku består av ett rutnät med 4×4 rutor, se figur 1. Rutorna ska fyllas med siffrorna $1-4$ så att följande gäller:

- Varje vågrät rad ska innehålla alla siffrorna $1-4$ (det vill säga endast en gång var).
- Varje lodrät kolumn ska innehålla alla siffrorna $1-4$ (det vill säga endast en gång var).
- Varje kvadrat markerad med feta linjer i figuren ovan ska innehålla alla siffrorna $1-4$ (det vill säga endast en gång var).

Från början är några rutor ifyllda och resten tomma (vilket anges med 0 vid inläsningen). Du ska skriva ett program som fyller i de tomma rutorna så att kraven uppfylls och skriver ut den fullbordade sudokun. Till givna indata finns det alltid exakt en lösning.

Körningsexempel:

```
Rad 1 ? 1000
Rad 2 ? 0001
Rad 3 ? 4000
Rad 4 ? 0003

1234
3421
4312
2143
```

UPPGIFT 5 – REGERINGSBILDNING

I september är det riksdagsval och därefter kan det bli hårda förhandlingar om hur regeringen ska se ut. I den här uppgiften ska du hjälpa till genom att skriva ett program som, givet valresultatet och partiernas positioner (till exempel på en höger-vänsterskala), ska bestämma vilket eller vilka partier som bör ingå i regeringen.

Programmet ska börja med att fråga efter antalet partier $n, 2 \leq n \leq 10$. Därefter ska det för varje parti fråga efter antalet riksdagsmandat $m, 1 \leq m \leq 1000$ och partiets position p på en hundragradig heltalsskala $1 \leq p \leq 100$. Programmet ska sedan ange den bästa regeringen genom att skriva de ingående partiernas nummer åtskilda med blanksteg.

Den regering som programmet ska välja ut ska ha två egenskaper:

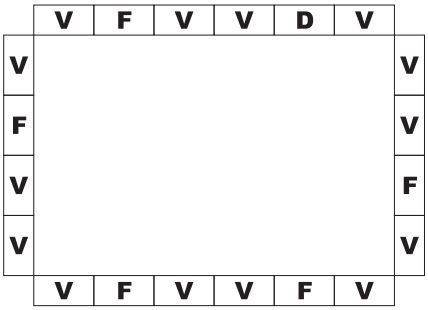
1. Det ska vara en majoritetsregering, det vill säga summan av antalet mandat för de ingående partierna ska vara större än hälften av det totala antalet mandat. För att göra programmet internationellt gångbart kan det totala antalet mandat variera men det är alltid ett udda antal.
2. Av alla regeringar som uppfyller kriterium 1 ska den ha den lägsta *oenigheten*. För enkelhets skull definierar vi oenighet som det minsta sammanlagda antalet steg på skalan som de ingående partierna måste ta för att kunna inta en gemensam position. Exempelvis, för en trepartiregering med partipositioner 4, 8 och 9 så är oenigheten 5 eftersom den bästa positionen som regeringen kan inta är 8, vilket gör att det första partiet måste ta 4 steg, det tredje måste ta 1 steg och det andra partiet inte behöver byta position alls. Om regeringen istället skulle inta position 7 eller 9 så måste partierna ta sammanlagt 6 steg så det är inte aktuellt. Om det finns flera regeringar med den lägsta oenigheten så kan programmet skriva ut vilken som helst av dem.

Körningsexempel:

```
Antal partier ? 6
Parti 1, mandat   ? 35
Parti 1, position ? 12
Parti 2, mandat   ? 26
Parti 2, position ? 11
Parti 3, mandat   ? 10
Parti 3, position ? 8
Parti 4, mandat   ? 15
Parti 4, position ? 9
Parti 5, mandat   ? 65
Parti 5, position ? 4
Parti 6, mandat   ? 20
Parti 6, position ? 16
Bästa regeringen  ? 3 4 5
```

Kommentar: Oenigheten för (3 4 5) är alltså 5 och det är en majoritetsregering eftersom de ingående partierna har 90 av 171 mandat. Näst bästa majoritetsregering är (1 2 3 4) med oenighet 6. Tre regeringar ger oenighet 7: (2 5), (2 3 5) och (2 4 5).

UPPGIFT 6 – BYGGSATSHUS



FIGUR 2. *Exempel på ett giltigt hus med storlek* 6×4*,* 5 *fönster och* 1 *dörr. Det finns* 216 *olika*

Ett byggsatshus består av tre olika komponenter: *dörrar, fönster* och *väggar*. Varje komponent är, för enkelhets skull, exakt 1 m lång.

Ett byggsatshus är rektangulärt och ska ha en viss *längd*, en viss *bredd*, ett visst *antal dörrar* och ett visst *antal fönster*. Dessutom ska följande gälla:

- Varje sida av huset måste ha minst ett fönster eller en dörr
- Det får finnas högst en dörr på varje hussida
- Fönster och dörrar måste omges på båda sidor av väggar
- Komponenterna närmast hörnen måste också alla vara väggar

Skriv ett program som tar emot husets längd $\mathtt{l}$ och bredd $\mathtt{b}$ i meter, $3 \leq \mathtt{b} \leq \mathtt{l} \leq 11$, antalet fönster $\mathtt{f}, 1 \leq \mathtt{f} \leq 20$ och antalet dörrar $\mathtt{d}, 1 \leq \mathtt{d} \leq 4$ och som sedan beräknar antalet olika byggsatshus som kan byggas enligt de givna specifikationerna. Två hus räknas som olika om minst en sida skiljer sig åt beträffande fönster- och dörrplacering. Även väderstrecken spelar roll, det vill säga två hus som skulle bli identiska om det ena roterades räknas fortfarande som olika hus.

Till givna indata finns alltid minst ett giltigt hus och det totala antalet överstiger inte 2 miljarder. Körningsexempel:

```
Längd ? 6
Bredd ? 4
Antal fönster ? 5
Antal dörrar ? 1
Antal hus: 216
```