UPPGIFT 1 – V75

FIGUR 1.

Varje lördag året om spelar tusentals svenskar på travspelet V75. Spelet går ut på att finna sju vinnande hästar i lika många lopp.

```
Lopp 1:  5  7
Lopp 2:  1  3  5  7  8 11
Lopp 3:  2  9
Lopp 4:  3
Lopp 5:  1  2  4  5  6 11
Lopp 6:  2  3  4  6  9
Lopp 7:  2  5  8 10 12
```

På kupongen kan man *strecka*, ta ut, flera hästar i ett och samma lopp. I systemet ovan har spelaren tagit ut hästarna 5 och 7 i första loppet och inte mindre än sex hästar $1, 2, 4, 5, 6, 11$ i femte loppet.

Antalet rader som ingår i systemet beräknas genom att multiplicera antalet hästar i varje lopp:

$$2 \cdot 6 \cdot 2 \cdot 1 \cdot 6 \cdot 5 \cdot 5 = 3600$$

Systemet ovan består av 3600 rader.

Utdelning ges normalt för rader med $7, 6$ och 5 rätt. Den som har 7 rätt kan ofta förvänta sig ett antal 6:or och ännu fler 5:or, allt efter systemets storlek.

Ägaren till systemet ovan blev lycklig vinnare med 7 rätt, där den rätta raden var $7, 3, 2, 3, 4, 4, 5$. Men dessutom hade han 6 rätt på *bland andra* raderna $5, 3, 2, 3, 4, 4, 5$ och $7, 3, 2, 3, 4, 9, 5$ och 5 rätt *på många*, bland dem $7, 11, 2, 3, 11, 4, 5$ och $7, 3, 2, 3, 4, 2, 2$.

Skriv ett program som tar emot uppgifter om antalet streckade hästar i varje lopp ($1 \leq s_i \leq 15$) och utdelningen på $7, 6$ och 5 rätt, i hela kronor. Programmet ska sedan beräkna den totala vinsten för ett sådant system *där det finns en rad med 7 rätt.*

Indata: Från exemplet ovan får vi antalet hästar från varje lopp. Dessutom gav 7 rätt 50000 kr, 6 rätt 210 kr och 5 rätt 7 kr.

```
Antal hästar     : 2 6 2 1 6 5 5
Utdelning (7,6,5): 50000 210 7
```

Utformningen av layouten för inmatningen är av underordnad betydelse.

Utdata: Utdata består av en rad, den som anger den totala vinsten i kronor på systemet:

```
Den totala utdelningen (kr): 55306
```

UPPGIFT 2 – PALINDROMTAL

```
 9341    10780    19481
 1439    08701    18491
-----   -----    -----
10780    19481    37972

37972    65945   120901
27973    54956   109021
-----   -----    -----
65945   120901   229922
```

FIGUR 2.

I figuren ser du ett antal additioner. Vi startar med talet 9341, bildar ett nytt tal, 1439 genom att vända det första bak och fram och summerar, $9341 + 1439 = 10780$. I nästa steg tar vi denna summa, vänder den bak och fram och summerar. Vi får nu $10780 + 08701 = 19481$. Vi fortsätter på det sättet tills vi får ett tal som är lika med samma tal vänt bak och fram. Ett sådant tal kallar vi *palindromtal*. Efter 6 steg har talet 9341 nått fram till palindromtalet 229922

Undersöker vi talen i intervallet $[1, 9999]$ kommer vi att finna att alla tal utom 246 stycken nått fram till ett palindromtal senast efter 24 steg. Det märkliga är att inget av dessa tal har nått sitt mål ens efter 5000 steg och vi lämnar dem därför utanför i detta problem.

Du ska nu skriva ett program som tar emot en *undre* gräns, $\mathtt{l}$, och en *övre* gräns, $\mathtt{u}$, i form av två heltal, $1 \leq \mathtt{l} < \mathtt{u} \leq 9999$ och som beräknar vilket tal i detta intervall som kräver flest steg innan det når fram till ett palindromtal. Programmet ska förutom *talet* och *antalet steg* också skriva ut det *slutliga palindromtalet*.

Indata: Programmet startar med följande dialog där den undre gränsen förstås är mindre än den övre och där båda talen tillhör intervallet $[1, 9999]$.

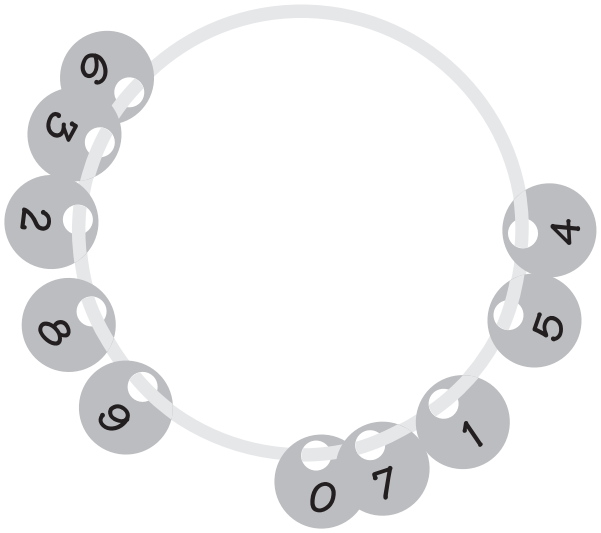
```
Undre gräns: 160
Övre  gräns: 180
```

Utdata: Programmet ska skriva ut det *tal* i det givna intervallet, som kräver flest steg för att nå fram till ett palindromtal, tillsammans med *antalet steg* och det slutliga *palindromtalet*. Från vårt exempel:

```
Talet         : 177
Antalet steg  : 15
Palindromtalet: 8836886388
```

Observera alltså att tal som kräver fler än 24 steg inte ingår i denna undersökning! Om det finns flera tal som kräver det maximala antalet steg räcker det att programmet anger ett av dessa.

UPPGIFT 3 – NUMMERBRICKORNA



FIGUR 3.

På en ring hade skolans vaktmästare trätt upp 10 brickor, se figuren. På varje bricka fanns en siffra. Var och en av de tio siffrorna $0 \ldots 9$ fanns med exakt en gång. När han satt och snurrade brickorna runt i ringen och samlade ihop dem i tre mindre grupper fick han, genom att läsa dem moturs, fram talen 715, 46 och 32890. Märkligt nog är

$$715 \cdot 46 = 32890$$

Han är nu intresserad av att ta reda på om det finns andra sätt att trä upp brickorna på ringen, där en liknande gruppering är möjlig och ber dig därför att skriva ett program.

Programmet ska ta emot de tio siffrorna, moturs, i den ordning de finns på ringen och bestämma tre tal, lästa moturs, där produkten av de två första är lika med det tredje. Observera att talen ska komma i denna ordning – först *faktorerna* och sedan *produkten* och att inget av talen får inledas med siffran 0.

Indata: Programmet frågar efter de 10 siffrorna, i moturs ordning, som de förekommer på ringen med början från valfri siffra.

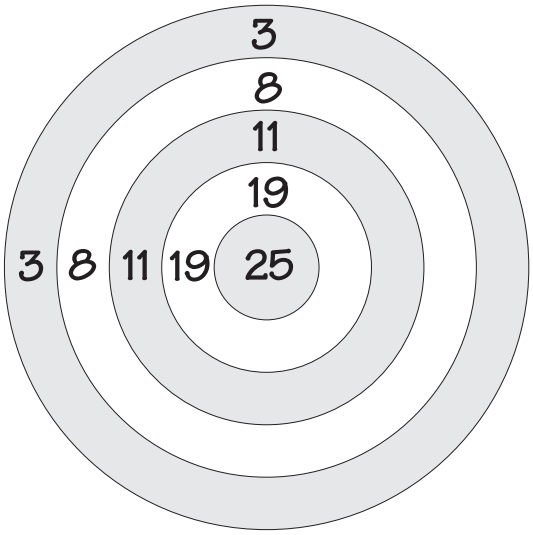
```
Siffrorna: 0715463289
```

Utdata: Programmet skriver ut en rad som innehåller de två faktorerna och produkten:

```
715*46=32890
```

Om det finns fler än en lösning räcker det att presentera en av dem.

UPPGIFT 4 – PILKASTNING



FIGUR 4.

När man får reda på att någon fått resultatet 36 med 6 pilar mot måltavlan i figuren kan man räkna fram att de 6 pilarna träffat ringarna: $3, 3, 3, 8, 8, 11$, eftersom det är enda sättet att uppnå detta resultat.

Eftersom alla sex pilarna måste ha hamnat på tavlan kan man ännu enklare säga i vilka ringar de måste finnas, om summan är 18 eller 150. 18 är den minsta summa man kan uppnå, alla pilar i 3 och 150 är det högsta resultatet, med alla pilar i 25.

Om, å andra sidan, någon påstår sig uppnått summan 37 vet man att denne inte talar sanning eftersom denna summa är omöjlig att uppnå. Om resultatet däremot blivit 34 finns det två möjligheter till detta resultat – antingen $3, 3, 3, 3, 11, 11$ eller $3, 3, 3, 3, 3, 19$.

Skriv ett program som tar emot uppgifter om poängen på måltavlan och som sedan bestämmer antalet poängresultat som kan erhållas på *exakt ett sätt*. Antalet pilar är alltid 6, och ingen missar tavlan.

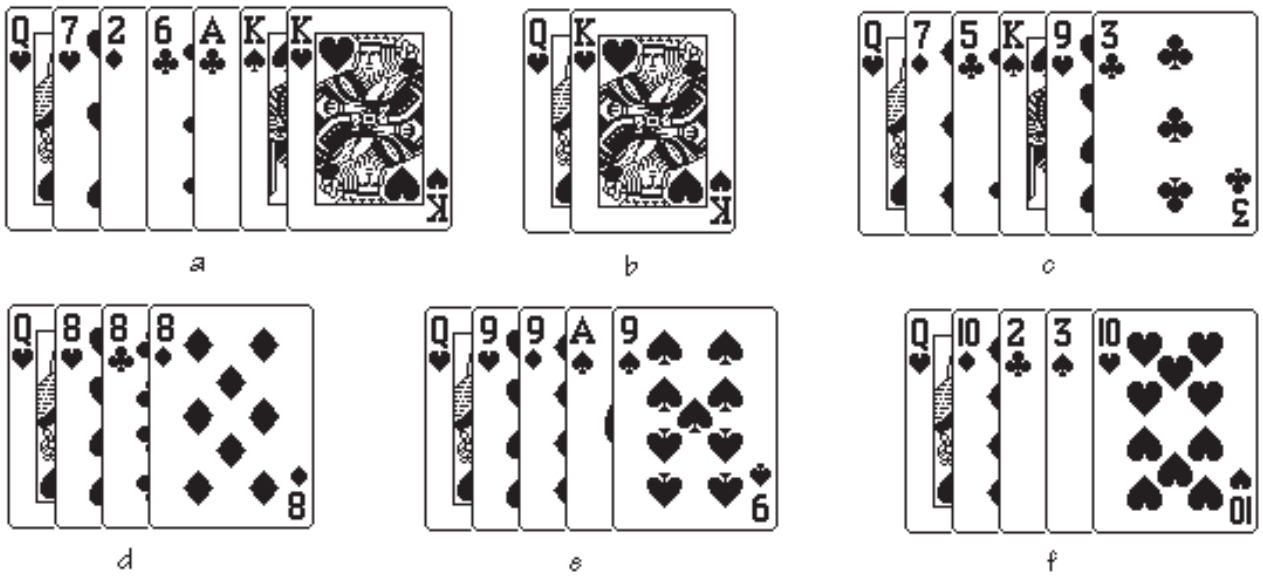
Indata: Programmet ska först ta emot uppgift om antalet ringar på måltavlan $n, 2 \leq n \leq 10$ och därefter poängen $p_i, 1 \leq i \leq n$ för de n ringarna, heltalen $p_i \in [1, 30]$

```
Antal ringar   : 5
Ringarnas poäng: 3 8 11 19 25
```

Utdata:

```
46 resultat kan bara nås på ett enda sätt
```

UPPGIFT 5 – KUNGEN MÖTER DAMEN



FIGUR 5.

I den här uppgiften ska Du låta datorn lägga en enkel patiens – många gånger – för att på det sättet uppskatta sannolikheten för att patiensen *går ut*.

En vanlig kortlek, med 52 kort används. Översta kortet i leken ska vara ♡*dam* och det understa ♡*kung*. De övriga 50 är blandade på vanligt sätt.

Ett kort i taget läggs ut på bordet med framsidan upp. Det första är då förstås ♡*dam*. När ett kort, som läggs ut, har samma *valör*, $(2, 3, 4, 5, 6, 7, 8, 9, 10, \mathsf{J}, \mathsf{Q}, \mathsf{K}, \mathsf{A})$, *eller* samma *färg*, $(\spadesuit, \heartsuit, \diamondsuit, \clubsuit)$ som det som ligger 2 eller 3 kort framför, så får mellanliggande kort plockas bort. Målet med patiensen är att endast ♡Q och ♡K ska finnas kvar på bordet – de har mötts!

Figuren förklarar vidare:

a Ett *slutläge* där inget mer kan göras. Patiensen har inte *gått ut*.
b Sluläget då patiensen har *gått ut*
c I detta läge kan ♠K och ♠9 plockas bort eftersom ♣5 har samma färg som ♣3. ♣3 flyttas sedan framåt så att det hamnar direkt bakom ♣5.
d Här kan ♣8 plockas bort eftersom ♢8 har samma valör som ♡8
e I de fall då det finns möjlighet att plocka bort både ett och två kort ska programmet alltid välja att avlägsna **två**. Alltså väljs ♢9 och ♠A av programmet eftersom ♡9 har samma valör som ♠9. Detta före enbart ♠A som också kunde ha varit tänkbart.
f När ett eller två kort plockas bort i raden uppstår en ny situation som kan leda till att fler kort kan plockas bort. Här kommer till sist ♡Q och ♡10 att bli kvar efter att först ♣2 och ♠3 kan tas bort. Den nya situationen leder till att ♢10 kan tas bort.
g Nästa kort läggs inte ut förrän alla borttagningar som är möjliga har utförts.

En del av problemet är att få till en funktion som blandar de 50 korten på ett korrekt sätt.
Indata: Programmet ska fråga efter hur många gånger patiensen ska läggas

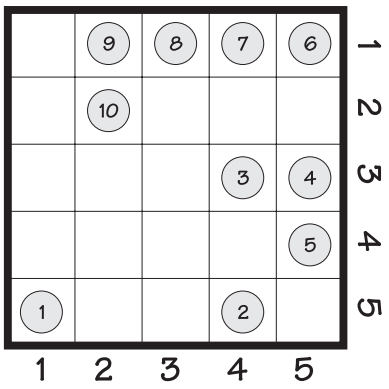
```
Hur många gånger ska patiensen läggas: 10000
```

Här får Du vara beredd på tal ända upp till 1 000 000

Utdata: Programmet ska ange, i procent med **två decimaler**, andelen lyckade försök, det vill säga hur ofta patiensen gick ut.

Under testen kommer fyra körningar att göras. Om minst tre av dessa ger ett resultat som ligger inuti, ett i svaret givet intervall, kommer programmet att betraktas som korrekt.

UPPGIFT 6 – BRICKSPELET



FIGUR 6.

I figuren ser Du ett bräde. Tio av de 25 rutorna innehåller brickor. Målet är att avlägsna dessa brickor, en i taget, efter speciella regler som följer här:

- Du får börja på vilken ruta som helst som innehåller en bricka.
- Förflyttning kan göras från en ruta till en annan endast horisontellt eller vertikalt (som ett torn i schack).
- Rutan till vilken man flyttar måste innehålla en bricka. Denna bricka tas bort från brädet.
- De rutor man passerar under en förflyttning får inte innehålla någon bricka.
- Efter nio drag ska brädet vara tomt från brickor.

Skriv ett program som tar emot uppgifter om var de 10 brickorna finns och som ger en dragföljd för hur problemet ska lösas.

Indata: Indata består av fem rader. Varje rad beskriver en rad på brädet där ”1” betyder att rutan innehåller en bricka och ”0” att rutan är tom. Enligt exemplet ovan

```
? 01111
? 01000
? 00011
? 00001
? 10010
```

Utdata: Utdata består av 10 koordinatangivelser, talpar som innehåller rad och kolumn för den bricka som ska tas bort.

För vårt exempel:

```
(5 1) (5 4) (3 4) (3 5) (4 5) (1 5) (1 4) (1 3) (1 2) (2 2)
```

Formateringen av utdata är på intet sätt viktig.