Uppgift 1

Askungen mötte den Goda Fen i skogen, och den Goda Fen (för att göra en lång historia kort) sade:

»Nå, min älskling, vad kan jag göra för dig?»

»Jag önskar att få bli evigt ung», svarade Askungen

»Så ska ske», sa den Goda Fen

»Du förstår», förklarade Askungen, »jag hatar att bli lika gammal som Begonia». (Begonia var den äldsta av Askungens två styvsystrar) »Begonia är åtta år äldre än jag»

Den Goda Fen vidrörde Askungen med sitt spö. »Förbli evigt vid den ålder du har idag»

När systrarna hörde om den Goda Fens gåva till Askungen, blev de mycket, mycket onda. »Din självisk a lilla slampa», sade Begonia. »Du glömmer att vår far varje år ger oss en check på det antal daler som motsvarar produkten av våra tre åldrar»

»Just det», sa Fuchsia, som hade blomkålsöron men var duktig i huvudräkning. »Låt mig se – under de följande två åren kommer din självisk het att kosta oss tillsammans 1382 daler»

Hur gammal är Askungen?

Skriv ett program som tar emot uppgift om antalet daler som de tre flickorna förlorar och med hjälp av detta belopp bestämmer deras åldrar. Eftersom alla beräkningar ska göras med heltal fungerar bara vissa indata, däribland 1382. Vid testning av ditt programförslag kommer endast indata, som ger heltalslösningar med åldrar i intervallet [1,30], att ges.

Indata: Antalet daler de tre flickorna förlorar tillsammans under de två åren.

Utdata: Askungens, Begonias och Fuchsias ålder.

Uppgift 2

Den nya myntreformen innebär att belopp avrundas till närmaste jämna 50-öring. Som till exempel:

Före	Efter
19,25	19,50
8,09	8,00
17,75	18,00
5,67	5,50

Fullborda programmet nedan genom att fylla i den påbörjade tilldelningssatsen i programmet, så att den utför en korrekt avrundning av det inmatade beloppet (fore). OBS Du får inte lägga till ytterligare satser i programmet!

```
program avrundning;
  var fore,efter:real;
  begin
    repeat
      write('Pris före avrundning ');
      readln(fore);
      efter:= ?
      writeln('Pris efter avrundning ',efter:7:2);
    until fore=0;
  end.
```

Indata: Belopp före avrundning (reellt tal med två decimaler)

Utdata: Belopp efter avrundning (reellt tal med två decimaler)

Uppgift 3

$\frac{22}{7}$ och $\frac{355}{113}$ är rationella tal, kända för att vara goda approximationer till π. Givetvis kan man finna hur många bättre approximationer som helst. Sök via ett program den bästa approximationen i form av ett rationellt tal där både täljare och nämnare är mindre än 100000.

Använd typerna DOUBLE eller EXTENDED för beräkningarna. π korrekt avrundat till 20 siffror: 3.1415926535897932385.

Indata: -

Utdata: täljare och nämnare för den bästa aproximationen

Uppgift 4

När den nye chefen för ortens elverk åkte ut för att besiktiga elledningarna blev han upprörd över elstolparnas placering.

»Hur kan man placera stolparna på det viset?», utbrast han, »det är ju olika avstånd mellan varje par. Flytta genast på dom, så att mellanrummet blir lika stort! Och se till att den totala förflyttningen samtidigt blir så liten som möjligt»

Chefens underordnade torkade svetten ur de pannor som redan låg i djupa veck och funderade och funderade ...

Skriv, för att hjälpa dessa tjänstemän, ett program som inleder med att fråga efter antalet stolpar n, ($3 \leq n \leq 15$) och sedan i tur och ordning efter stolparnas placering. Vi kan tänka oss stolparna utplacerade på x-axeln med *stolpe 1* i origo och övriga på den positiva delen av axeln.

Programmet ska sedan bestämma hur långt stolparna ska flyttas, så att den totala förflyttningen blir så liten som möjligt och så att avståndet, mellan två på varandra följande stolpar, blir konstant.

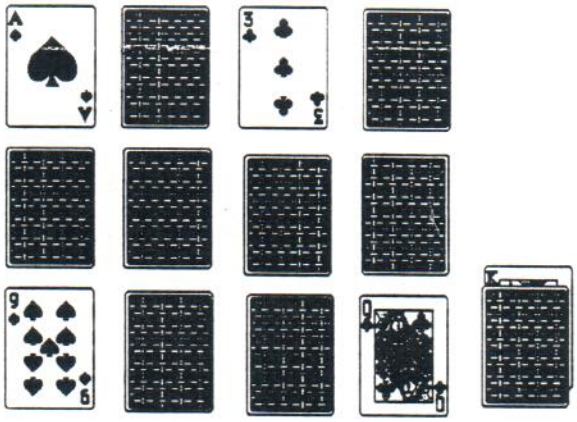
Utskriften ska bestå av en tabell innehållande stolpnummer och nya placeringen samt det nya, konstanta mellanrummet. Stolpar som hamnar till vänster om stolpe 1's ursprungsplacering anges med negativa tal.

Ledtråd: Du får använda de faktum, att sökt minimum alltid kan erhållas genom att låta två stolpar stå kvar på ursprungsplatsen.

Indata: Antal stolpar och stolparnas placering (ett reellt tal) på en tänkt x-axel där första stolpen befinner sig i origo.

Utdata: Tabell med stolpnummer och stolpens nya placering. Dessutom det nya, konstanta mellanrummet. Alla tal ges med två decimaler.

Uppgift 5



Denna patiens kallas, liksom många andra, "idioten", antagligen för att den gör skäl för namnet.

Spelet inleds med att lägga upp de 52 välblandade korten i 13 högar (se fig), med fyra kort i varje. Alla från början med baksidan uppåt. Det översta kortet dras från högen längst ned till höger (talongen). Valören bestämmer i vilken hög detta kort ska placeras (underst med framsidan uppåt). Är kortet ett Ess hamnar det i högen längst upp till vänster. Högen till höger är till för Tvåor, och så vidare ner till sista radens näst sista hög där Damerna ska samlas. Som nästa kort tas nu det översta i den hög som just utökades till fem kort. Valören på detta kort bestämmer i vilken hög det ska ligga, och så vidare. När en kung ger sig till känna läggs den underst i talongen, på samma sätt som för andra kort.

När patiensen pågått ett tag kommer troligtvis vissa högar att bli helt uppåtvända (som Ess-, Tre-, Nio- och Damhögen i figuren), alla korten i en valör har därmed samlats på rätt ställe. Målet med patiensen är att få alla kort på rätt plats innan den fjärde kungen dyker upp. Om detta sker har patiensen "gått ut". Även om inte alla högar är helt uppåtvända, efter att de fyra

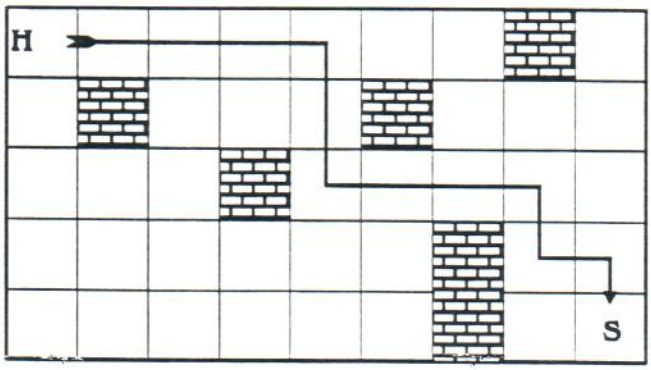
kungarna visat sig kan patiensen ändå ha "gått ut". Detta kräver dock att samtliga ouppvända kort redan ligger i rätt hög!

Skriv nu ett program som frågar efter antalet gånger patiensen ska läggas, lägger patiensen så många gånger och till sist skriver ut, i hur många procent av försöken patiensen "gått ut". Det är viktigt att Du har en korrekt blandningsalgoritm. Vid testning av programmet kommer flera körningar att göras och det är därför viktigt att du använder RANDOMIZE.

Indata: Antalet patienser som ska läggas (kan vara upp till 5000).

Utdata: Hur många procent av antalet försök, patiensen "gått ut" (ges med två decimaler).

Uppgift 6



När Kalle ska gå till skolan (rutan märkt S i figuren), startar han promenaden i rutan märkt H. Eftersom Kalle vill ha omväxling försöker han i det längsta att varje morgon ta en ny väg till skolan (se exempel i figuren på en av hans möjliga vägar). Han går aldrig längre än han behöver och väljer därför bara att gå i ostlig eller sydlig riktning (sydostlig riktning är omöjlig). På vägen kan finnas oframkomliga gator (tegelrutor i figuren).

Frågan är nu: Hur många olika vägar finns det för Kalle att ta sig till skolan?

Skriv ett program som först frågar efter stadens storlek (antal rader och antal kolumner, dessa tal kan vara olika men är ≤15). I nästa fråga ska man ange hur många oframkomliga vägar staden innehåller och därefter var någonstans, var och en av dessa är belägna genom att först ange rutans rad och därefter dess kolumn.

Programmet ska sedan bestämma antalet olika vägar mellan H och S där varje steg måste vara en förflyttning i antingen ostlig eller sydlig riktning och där man inte får beträda oframkomliga rutor.

Indata: Den rektangulära stadens storlek genom antalet rader och kolumner. Antalet oframkomliga gator och var dessa är belägna genom att först ange raden och sedan kolumnen.

Utdata: Ett tal som anger antalet olika vägar från H till S.