Uppgift 1 – Tvåpotenser.

$$2^{142} = 5575186299632655785383929568162090376495104$$

$n = 142$ är det minsta värde på n för vilket 2^n inleds med siffrorna 55. Uppgiften består i att skriva ett program som tar emot *en inledning* p, ett tal med upp till 20 siffror utan inledande nollor, som söker upp det *minsta* n sådant att 2^n inleds med det givna talet p.

Programmet kommer endast att testas med *inledningar* som verkligen finns och som har en exponent $n < 10000$.

Indata: Programmet frågar efter talet p

```
Talet ska inledas med: 55
```

Utdata: Den exponent n som ger denna inledning

```
Det sökta värdet på exponenten är 142
```

Uppgift 2 – Hissen i lustiga huset.

I *Lustiga Huset* finns $v, 1 \leq v \leq 100$ våningar. I den märkliga hissen finns bara två knappar. Dels en som förflyttar hissen u våningar *uppåt* och dels en som förflyttar hissen n våningar *nedåt*. Men med hjälp av en kombination av *resor*, *uppåt* och *nedåt* kan man ta sig till önskad våning m.

Du ska skriva ett program som tar emot uppgifter om v, u, n och m och som sedan beräknar *det minsta antalet resor* som behövs för att nå våning m, målet. En *resa* är alltså en knapptryckning som för hissen från en våning till en annan. Den första resan startar alltid på våning 1, som ligger i husets bottenplan. Huset saknar källare och vind, vilket betyder att hissen alltid måste befinna sig någonstans mellan våning 1 och v.

Indata: Programmet inleds med att fråga efter v, u, n och m

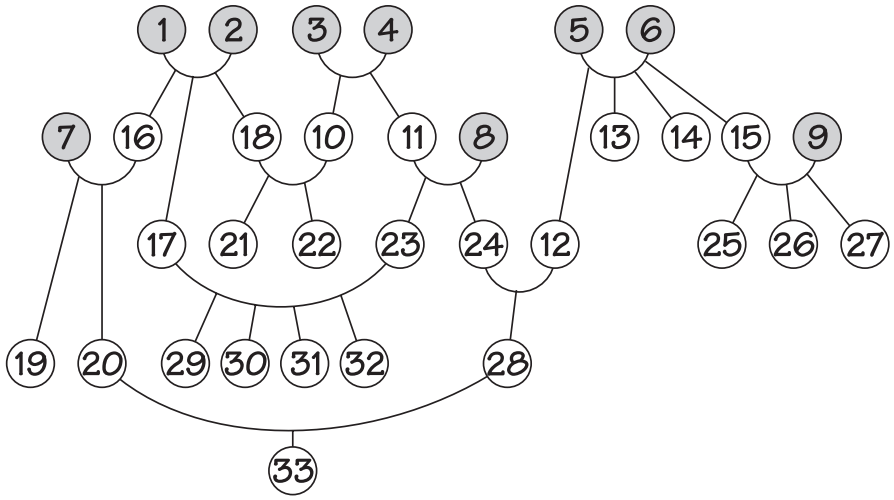
```
Hur många våningar har huset: 78
Förflyttning uppåt: 15
Förflyttning nedåt: 8
Till vilken våning ska du: 35
```

Utdata: En rad som talar om det minsta antalet resor som behövs för att nå målet:

```
Det behövs minst 13 resor för att nå målet
```

Endast testexempel där det finns en lösning kommer att användas.

UPPGIFT 3 – KUSINER.



FIGUR 1. *Ett exempel på släktträd. Bågar visar gifta par. Räta linjer som utgår från bågarna visar deras barn. Grå individer saknar förfäder i detta släktträd.* 32 *och* 28 *är kusiner därför att de har föräldrar som är syskon.*

I denna uppgift ska du med hjälp av information om *vem som har barn med vem* räkna ut hur många kusiner olika personer har. Två personer är kusiner om någon av deras föräldrar är syskon.

Vi utgår från N personer (numrerade från 1 till N), som inte är släkt och inte har några kusiner. Sedan inträffar ett antal *händelser*, där varje händelse innebär att två personer gifter sig och skaffar ett antal barn (K stycken). Dessa nya personer numreras med de lägsta lediga numren, för den första händelsen innebär detta $N+1, N+2 \ldots N+K$. Dessa kan sedan i sin tur gifta sig och så vidare. Två personer som har en gemensam förfader kan inte gifta sig och inga personer är kusiner på både faderns och moderns sida.

Indata: På första raden i filen `uppg3.dat` står två heltal, *antalet personer* från början N samt antalet händelser M. Sedan följer M rader med tre heltal P, Q och K, där P och Q är numren på de personer som gifter sig och K är antalet barn de får. P och Q finns bland de personer som antingen fanns från början eller som har definierats genom de tidigare händelserna. Det totala antalet personer överstiger inte 100.

Utdata: En lista över hur många kusiner varje person har. De som inte har några kusiner ska inte skrivas ut.

Exempel: Se även figuren

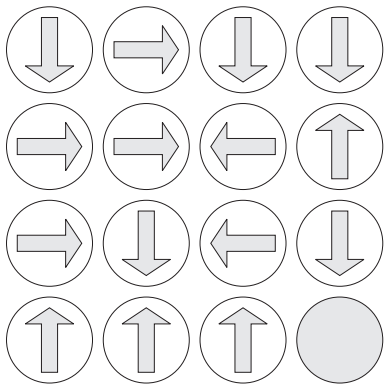
Indata

```
 9 10          9 personer från början, 10 händelser
 3  4 2        3 och 4 får barnen 10 och 11
 5  6 4        5 och 6 får barnen 12, 13, 14 och 15 och så vidare
 1  2 3
 7 16 2
18 10 2
 8 11 2
 9 15 3
12 24 1
17 23 4
20 28 1
```

Utdata

```
Nr 19 har 6 kusiner
Nr 20 har 6 kusiner
Nr 21 har 8 kusiner
Nr 22 har 8 kusiner
Nr 23 har 2 kusiner
Nr 24 har 2 kusiner
Nr 25 har 1 kusiner
Nr 26 har 1 kusiner
Nr 27 har 1 kusiner
Nr 28 har 7 kusiner
Nr 29 har 5 kusiner
Nr 30 har 5 kusiner
Nr 31 har 5 kusiner
Nr 32 har 5 kusiner
```

UPPGIFT 4 – PILPUSSLET.



FIGUR 2.

I figuren ser Du ett exempel på ett utgångsläge i *PilPusslet*, som ska lösas genom 15 *drag*. Starten sker alltid från ringen *högst upp i vänstra hörnet*.

Ett drag innebär en förflyttning i *pilens riktning* ett eller flera steg till en ring Du tidigare inte besökt. Pilen i den nya ringen bestämmer riktningen för nästa drag, och så vidare. I det 15:e draget ska Du nå den tomma ringen *längst ned i högra hörnet*. Samtidigt ska Du då ha besökt varje ring i pusslet precis en gång.

Skriv ett program, som tar emot en startställning och som bestämmer och skriver ut vägen genom pusslet.

Indata: består av fyra rader. På varje rad finns fyra tecken hämtade från $n, e, s, w, *$. När Du jämför körningsexemplet nedan med figuren förstår Du vad beteckningarna står för

```
rad 1: sess
rad 2: eewn
rad 3: esws
rad 4: nnn*
```

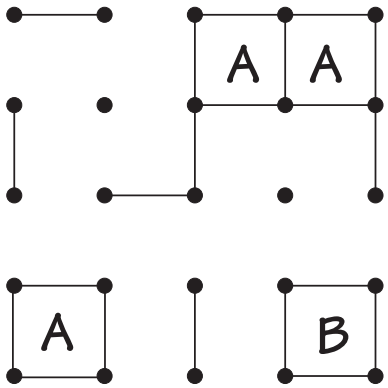
Det sista tecknet i den sista raden är alltid en asterisk.

Utdata: Resultatet ska presenteras som en tabell med fyra rader och fyra kolumner. Första talet på första raden ska alltid vara 0 och sista talet på sista raden ska alltid vara 15. Övriga tal anger ordningsnumret för när just ringen i den positionen besöktes. För exemplet ges följande resultattabell:

```
 0   6   7  13
10  11   9  12
 2   4   3  14
 1   5   8  15
```

Detta exempel har, liksom samtliga kommande tester, endast en lösning.

UPPGIFT 5 – ETT SPEL MED PENNA OCH PAPPER



FIGUR 3.

I figuren ser vi en situation i spelet *askar* (engelska *dots and boxes*). Från början finns bara ett antal punkter i ett kvadratiskt mönster (i denna uppgift alltid 5×5). Spelarna turas om att rita ett streck mellan två punkter. Strecket måste utgöra en sida i en av de 16 mindre kvadraterna.

Då en spelare lyckas rita det fjärde och sista strecket i en kvadrat har han vunnit denna och skriver därför sin bokstav inuti kvadraten. Dessutom behåller han turen och får fortsätta att dra streck så länge han kan vinna nya kvadrater. Han avslutar därför alltid sin omgång med ett streck som inte fullbordar någon ny kvadrat. (undantaget att spelet är slut – alla 40 strecken är ritade)

Din uppgift är nu att skriva ett program som, från en fil, tar emot drag och administrerar spelet. När listan av drag på filen är genomgången ska programmet ange ställningen i spelet. Skriva ut hur många kvadrater A respektive B har erövrat. A gör alltid första draget.

Indata: Programmet läser från filen `uppg5.dat`. Först ett tal som anger antalet drag $n, 1 \leq n \leq 40$, (partiet behöver alltså inte vara slutspelat).

På var och en av följande n rader finns ett drag angivet. De två första tecknen anger rad och kolumn för var strecket startar. Det tredje och sista tecknet anger riktningen. U, N, H eller V (uppåt, nedåt, höger eller vänster). Punkten $(1, 1)$ finns längst upp till vänster. Draget 55U startar alltså längst ned till höger och dras till till punkten ovanför $(4, 5)$.

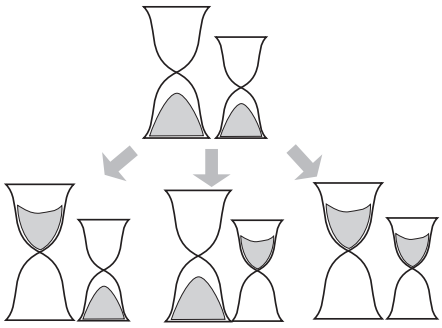
Indatafilen för vårt exempel innehåller alltså följande (**observera** dock att varje drag i filen tar upp en egen rad):

```
21
11H 21N 32H 33U 42V 43N 44H
55V 54U 55U 35U 25V 25U 14H
13H 13N 23H 14N 51H 51U 52U
```

Utdata: Programmet skriver ut ställningen på en enda rad:

```
A har 3 kvadrater och B har 1 kvadrater
```

UPPGIFT 6 – TIMGLAS



FIGUR 4.

Det program Du ska skriva för att lösa denna uppgift ska bestämma det minsta antalet *timglas* som behöver vändas för att mäta upp en given tid med hjälp av två olika timglas. Det större av de två timglasen behöver t_1 minuter för att rinna ut och det mindre behöver t_2 minuter. $t_1 > t_2$ och båda tiderna är heltal i intervallet $1 \ldots 20$. Båda timglasen har från start all sand på botten. Från detta läge kan ett eller båda timglasen vändas. Vända ett timglas, eller båda samtidigt, kan man sedan bara göra, exakt då ett (eller möjligtvis båda) runnit ut. Både det glas som just då fått all sand på botten och det andra kan vändas.

Två exempel förklarar det hela. Till vänster är $t_1 = 7$, $t_2 = 3$ och målet är 11 minuter. Till höger i tabellen är $t_1 = 7$, $t_2 = 4$ och målet 13 minuter.

tid	**Stora**	**Lilla**	**Vänd**	**Stora**	**Lilla**	**tid**	**Stora**	**Lilla**	**Vänd**	**Stora**	**Lilla**
0	0	0	Båda	7	3	0	0	0	Båda	7	4
3	4	0	Lilla	4	3	4	3	0	Lilla	3	4
6	1	0	Lilla	1	3	7	0	1	Båda	7	3
7	0	2	Lilla	0	1	10	4	0	Stora	3	0
8	0	0	Lilla	0	3	13	0	0			
11	0	0									

I båda dessa fall behövdes 6 vändningar. Observera att då båda timglasen vänds samtidigt räknas det som *två* vändningar. Det finns oftast många olika sätt att nå fram till samma minsta antal vändningar.

Indata: Tiderna t_1 och t_2 för de båda timglasen, två heltal i intervallet $1 \ldots 20$. Dessutom den tid man vill mäta upp, ett heltal i intervallet $1 \ldots 40$

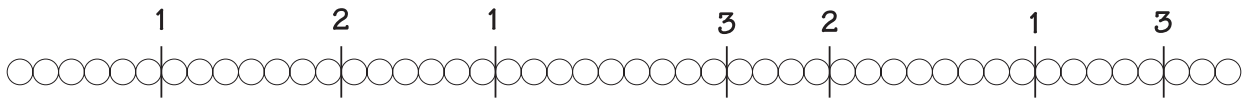
```
Det stora timglaset (minuter)? 7
Det lilla timglaset (minuter)? 5
Tid att mäta upp    (minuter)? 17
```

Tester kommer enbart att göras med tider som kan uppnås.

Utdata: består av ett tal – antal vändningar av mätglasen

```
Det minsta antalet vändningar som behövs är 3
```

UPPGIFT 7 – RESTRIKTIONSKLYVNING



FIGUR 5. *Figuren visar lösningen av exemplet i texten. Talet ovanför strecken, som visar de 7 klyvningsställena, anger vilket enzym som klyver på denna plats.*

Inom biokemin är det vanligt att man använder restriktionsenzymer när man arbetar med DNA. Som bekant är en DNA-molekyl uppbyggd av fyra sorters mindre byggstenar, nukleotider, som sitter ihop i två långa kedjor. I denna uppgift kan dock DNA-molekylen betraktas som en enda kedja.

Ett restriktionsenzym är ett enzym som klyver DNA-kedjan på vissa specifika ställen som bestäms av nukleotidsekvensen. Det klyver alltid mellan två nukleotider (se figur). Resultatet av en restriktionsklyvning blir ett antal kortare fragment. Om vi till exempel utgår från en DNA-kedja med 48 nukleotider kanske vi efter klyvning med ett visst restriktionsenzym har fått tre fragment med längderna 3, 17 och 28.

För att se vilka fragment som har bildats använder man gelelektrofores, som separerar DNA-fragmenten efter storlek. Man kan dock inte se i vilken ordning fragmenten satt i den ursprungliga molekylen. För att ta reda på detta behöver man göra flera klyvningar med olika restriktionsenzymer som klyver DNAt på olika ställen och sedan jämföra resultaten.

Din uppgift är att bestämma var i DNA-molekylen som ett antal restriktionsenzymer klyver när du får veta resultaten från gelektroforeserna.

Indata: På första raden i filen `uppg7.dat` finns två heltal, där det första, $L, 2 \leq L \leq 500$, är längden på det ursprungliga DNAt och det andra, $N, 2 \leq N \leq 10$ är antalet restriktionsenzymer du ska studera.

Sedan följer $N + 1$ resultat av restriktionsklyvningar (följt av gelelektrofores). Den första klyvningen har utförts med en blandning av alla N restriktionsenzymer, vilket gör att DNAt klyvs på alla möjliga ställen. De andra klyvningarna har utförts med endast ett restriktionsenzym, i tur och ordning från enzym nr 1 till enzym nr N. Två olika enzymer klyver aldrig DNAt på samma ställe.

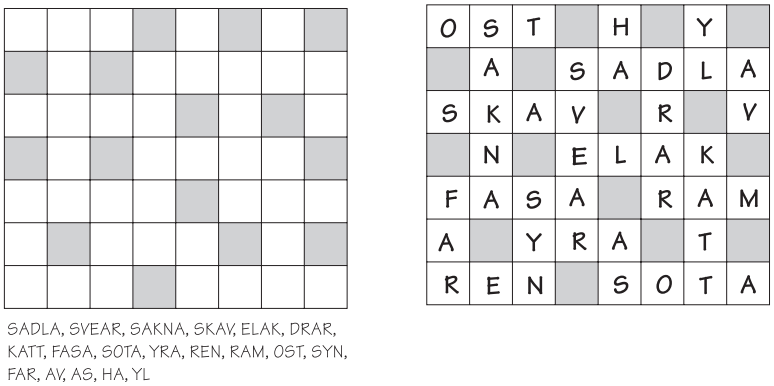
Vart och ett av resultaten anges med två rader. På första raden står ett heltal som anger hur många fragment, $K, 1 \leq K \leq 25$, som bildades. På andra raden står K heltal: längderna på fragmenten sorterade i storleksordning. Om två fragment har samma längd anges ändå båda fragmenten. Summan av de K talen är sålunda alltid lika med L. Exempel på en indatafil:

```
48 3
8                    klyvning med alla 3 enzymerna
3 4 5 6 6 7 8 9
4                    klyvning med enzym 1
6 8 13 21
3                    klyvning med enzym 2
13 16 19
3                    klyvning med enzym 3
3 17 28
```

Utdata: En rad med heltal som anger ordningen på klyvningsställena (se exemplet). Observera att det alltid finns en spegelvänd lösning. Det går givetvis bra att ange vilken lösning som helst av dessa. Utdata från exemplet blir

`1 2 1 3 2 1 3` *eller* `3 1 2 3 1 2 1`

UPPGIFT 8 – KORSORD



FIGUR 6. *Från det tomma korsordet med de givna orden ska ditt program nå fram till en lösning, som i detta exempel visas till höger.*

Du ska skriva ett program som löser ett litet korsord. Givet ett tomt korsord, samt de ord som ska placeras in. Ett ord är alltid minst två bokstäver långt och går antingen vågrätt (från vänster till höger) eller lodrätt (uppifrån och ner).

Indata: Filen `uppg8.dat` inleds med två heltal R och C ($2 \leq R, C \leq 8$), som anger antalet rader och kolumner i korsordet. Sedan ett heltal N, antalet ord som ska fyllas i. Sedan följer det tomma korsordet där '.' (punkt) betecknar ruta som ska fyllas i och '*' (asterisk) betecknar svart ruta. Därefter följer N rader med ett ord på varje (med tecknen A-Z). De längsta orden står först. Nedan början av den datafil, som ges av exemplet i figuren:

```
7 8 19
...*.*.*
*.*.....
....*.*.
*.*....*
....*...
.*...*.*
...*....
```

Direkt följt av 19 rader, som var och en innehåller ett ord. Vilka visas i figuren.

Utdata: Det ifyllda korsordet skrivs ut som nedan. Det finns alltid exakt en lösning.

```
OST*H*Y*
*A*SADLA
SKAV*R*V
*N*ELAK*
FASA*RAM
A*YRA*T*
REN*SOTA
```