UPPGIFT 1 – ÖVERSÄTTNING

Fikonspråket är ett hemligt språk med gamla anor som till och med har givit upphov till vissa svenska ord, till exempel *fimp* (fikonspråkets *fimpstukon* betyder *stump*). *Rövarspråket* användes av Kalle Blomkvist i Astrid Lindgrens böcker. Skriv ett program som översätter från rövarspråket till fikonspråket!

För båda språken gäller att man kan härleda varje ord från det motsvarande svenska ordet.

- För att översätta till fikonspråket delas ordet efter dess första *vokal* i två delar. Dessa delar sätts sedan ihop i omvänd ordning och dessutom tillfogas **fi** i början och **kon** i slutet av ordet. `GET` blir alltså `FITGEKON`, `LO` blir `FILOKON` och `ASTRONOM` blir `FISTRONOMAKON`.
- Översättning till rövarspråket sker bokstavsvis. Vokaler ändras inte alls, medan varje konsonant skrivs två gånger med ett **O** emellan. `MAT` blir alltså `MOMATOT` och `ODLA` blir `ODODLOLA`.

Programmet ska fråga efter ett ord ($\leq$ 30 tecken), skrivet på rövarspråk. Endast versaler används vid inmatningen. Ordet innehåller bara bokstäver A – Z och motsvarande svenska ord har minst en vokal. Du kan förutsätta att ordet följer reglerna för rövarspråket. Sedan ska programmet skriva ut ordet översatt till fikonspråk. Ett par körningsexempel:

```
Rövarspråk: COCYKOKELOL
Fikonspråk: FIKELCYKON

Rövarspråk:KOKALOLASOSFOFINONTOT
Fikonspråk: FILASFINTKAKON
```

UPPGIFT 2 – EN KAOTISK SEKVENS

Följande citat är hämtat från den välkända boken *Gödel, Escher, Bach* av *Douglas R. Hofstadter*:

"**En kaotisk sekvens** Ett sista exempel på rekursion inom talteorin leder till ett litet mysterium. Betrakta följande rekursiva definition av en funktion:

$$Q(n) = Q(n - Q(n-1)) + Q(n - Q(n-2)) \text{ för } n > 2$$
$$Q(1) = 1$$
$$Q(2) = 1$$

Den påminner om Fibonacci-definitionen därigenom att varje nytt värde är summan av två föregående värden – men inte av de *omedelbart* föregående två värdena. I stället anger de två omedelbart föregående värdena *hur långt bakåt man ska gå* för att komma till de tal vilkas summa skall utgöra det nya värdet! De första 17 Q-talen är:

$$1,\ 1,\ 2,\ 3,\ 3,\ 4,\ 5,\ \underset{\uparrow}{5},\ \underset{\uparrow}{6},\ 6,\ 6,\ 8,\ 8,\ 8,\ 10,\ \underbrace{9,\ 10}_{\text{antal steg åt vänster}},\ \ldots$$
$$5 + 6 = 11$$

För att få fram nästa värde går man respektive 10 och 9 steg åt vänster (räknat från de tre prickarna) och kommer då, som pilarna visar, till talen 5 och 6. Deras summa – 11 – är det nya värdet: $Q(18)$. Genom denna märkliga process kan listan på kända Q-tal användas för att utöka sig själv.

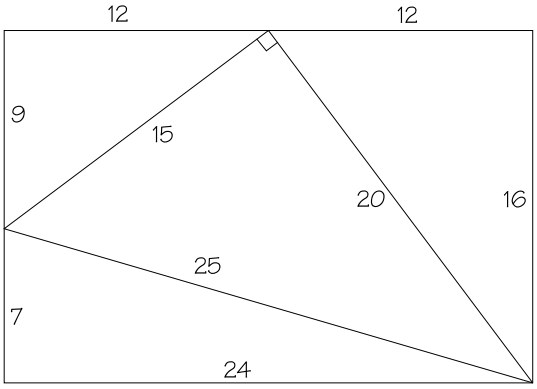
Den resulterande talföljden ger ett – milt uttryckt – planlöst intryck. Ju längre man kommer desto svårbegripligare ter den sig. Detta är ett av de mycket säregna fall där en till synes någorlunda naturlig definition leder till ett extremt förbryllande beteende – till ett under mycket välordnade former framkallar kaos. Man frågar sig naturligt nog huruvida detta skenbara kaos döljer någon subtil regelbundenhet. Definitionsmässigt finns regelbundenheten där givetvis, men det intressanta är om det finns något annat – i bästa fall icke-rekursivt – sätt att karaktärisera denna följd."

Skriv ett program som frågar efter ordningsnumret på det tal man vill beräkna och som skriver ut talet

```
Vilken tal ? 18
Talet har värdet 11
```

Vi kommer bara att testa ditt program upp till den 35:e talet.

UPPGIFT 3 – REKTANGEL MED TRIANGLAR



FIGUR 1.

I figur **??** visas en rektangel uppbyggd av fyra *pythagoresika trianglar*, det vill säga rätvinkliga trianglar med heltalssidor.

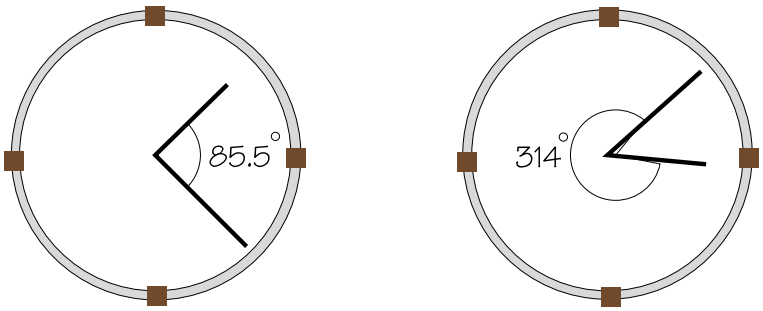
Skriv ett program som tar emot uppgift om rektangelns bas och bestämmer rektangelns höjd. Ingen sida i någon triangel överstiger till 300. Två körningsexempel:

```
Rektangelns bas: 117
Rektangelns höjd är 72
```

```
Rektangelns bas: 171
Rektangelns höjd är 180
```

Observera att den ”omslutna” triangeln alltid har ett hörn i rektangelns nedre högra hörn och de två andra på rektangelns motstående sidor. Observera också att det saknas lösning för de flesta baser, men du kan räkna med att för alla de baser som programmet testas med finns det precis en möjlig höjd.

UPPGIFT 4 – KLOCKAN



FIGUR 2. *Klockan till vänster är 1:21 och den till höger är 3:08*

Om någon frågar hur mycket klockan är, svarar de flesta ”kvart över fem”, 15 : 29 eller något liknande. Vill man göra det lite svårare så kan man annars svara med vinkeln mellan *tim*- och *minut*-visaren, eftersom man ur denna information entydigt kan bestämma klockslaget. Dock är det många människor som är ovana vid detta sätt att ange tider, så det vore bra att ha ett datorprogram som översätter till ett mer normalt format. Du ska skriva ett sådant program.

Vi förutsätter att vår klocka saknar sekundvisare och endast visar ett helt antal minuter (det vill säga: båda visarna hoppar framåt bara på hel minut). Vinkeln avläses genom att utgå från timvisaren och sedan mäta hur många grader medurs minutvisaren ligger (se figur 2). För att undvika decimaler anges vinkeln i tiondels grader (så att 85.5 grader skrivs som 855). Detta tal är alltid ett heltal mellan 0 och 3595 (inklusive) och är, som en följd av att endast hela minuter visas, alltid delbart med 5."

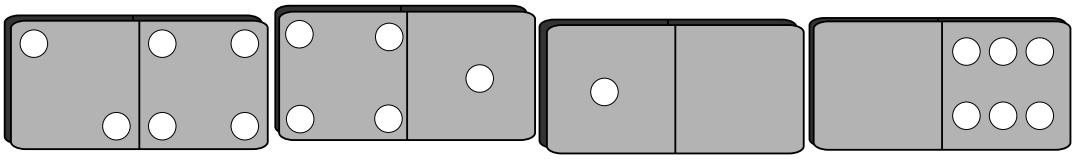
Programmet ska fråga efter en vinkel och sedan skriva ut tiden i vanligt digitalformat, alltså h : mm eller hh : mm, beroende på antalet timmar. Notera att minuterna alltid ska ges med två siffror. Vi förutsätter att det är morgon, så alla tider ska ligga mellan 0 : 00 och 11 : 59 (inklusive).

Två körningsexempel:

```
Vinkel: 855
Klockan är 1:21
```

```
Vinkel: 3140
Klockan är 3:08
```

UPPGIFT 5 – DOMINO



FIGUR 3.

En dominobricka består av två sammansatta kvadrater, där varje kvadrat har mellan 0 och 6 prickar. I en variant av domino har en spelare ett antal dominobrickor som han vill lägga ut i en lång rad, så att de ändar som vidrör varandra har samma antal prickar (se figur **??**).

Din uppgift är att skriva ett program som, givet utseendet på ett antal dominobrickor (som mest 15), lägger ut dessa i en rad enligt reglerna ovan. Du får rotera brickorna 180 grader, och alla brickor måste läggas ut. Du kan utgå ifrån att det finns minst ett sätt att lägga ut brickorna på, samt att de brickor du får är unika. En bricka beskrivs som ett tvåsiffrigt tal. Körningsexempel:

```
Antal brickor: 5
Bricka 1: 11
Bricka 2: 06
Bricka 3: 43
Bricka 4: 14
Bricka 5: 10

34 41 11 10 06
```

UPPGIFT 6 – KUBEN

En stor kub är indelad i $26 \times 26 \times 26$ kubiska celler. Positionen för varje cell kan anges med tre bokstäver, där en bokstav ger positionen i en dimension och är en versal i intervallet $A - Z$ (enligt ASCII-standard, det vill säga inklusive *W*). Sålunda har hörncellerna positionerna AAA, AAZ, AZA, AZZ, ZAA, ZAZ ,ZZA och ZZZ.

En liten radiostyrd robot rör sig inuti kuben, dock med vissa restriktioner. Varje kommando som roboten får innebär att den förflyttar sig ett antal celler i en av de sex riktningar som är parallella med kubens kanter (framåt respektive bakåt i var och en av de tre dimensionerna).

Antalet celler som roboten går för varje kommando är dock inte godtyckligt utan måste vara en av de godkända steglängderna, vilka är förutbestämda och givna från början. Om vi exempelvis antar att 3 är den enda godkända steglängden och roboten befinner sig i cell EHR, kan den med ett kommando förflytta sig till någon av cellerna BHR, HHR, EER, EKR, EHO eller EHU. Om dessutom 5 är en godkänd steglängd finns det ytterligare fem alternativ. Den sjätte riktningen (bakåt i den första dimensionen) är inte tillåten eftersom roboten då kommer utanför kuben.

Du ska skriva ett program som, givet *godkända steglängder*, en *startposition* och en *slutposition*, beräknar det minsta antalet kommandon som behöver ges för att förflytta roboten från startpositionen till slutpositionen.

Programmet ska fråga efter antalet godkända steglängder $n, 1 \leq n \leq 25$ och sedan låta användaren mata in dessa tal $k_i, i = 1 \ldots n$, $1 \leq k_1 \leq 25$. Därefter ska programmet fråga efter en startposistion och sedan en slutposition. Dessa matas in som strängar var och en bestående av exakt 3 versaler i intervallet $A - Z$. Programmet ska skriva ut det minsta antalet kommandon som behöver ges för att förflytta roboten från startpositionen till slutpositionen. Om det är omöjligt att förflytta roboten från startpositionen till slutpositionen med hjälp av de givna steglängderna ska programmet skriva ut `Omöjligt`. Två körningsexempel:

```
Antal steglängder: 3
Steglängd 1: 5
Steglängd 2: 6
Steglängd 3: 7
Startposition: AHT
Slutposition: JFY
Antal kommandon: 6
```

```
Antal steglängder: 2
Steglängd 1: 10
Steglängd 2: 14
Startposition: FFF
Slutposition: GGG
Omöjligt!
```

Kommentar: En av många möjliga förflyttningar är AHT – AAT – HAT – OAT – OFT – OFY – JFY.