UPPGIFT 1 – TVETYDIGA DATUM

Datum skrivs på olika sätt i olika länder. Till exempel skulle datumet 03/05/01 i Sverige betyda 1 maj 2003, medan det i USA skulle vara 5 mars 2001 och i en del andra länder 3 maj 2001 (vi kan utgå ifrån att årtalet alltid är under 2000-talet, dvs mellan 2000 och 2099).

Detta kan bland annat orsaka bekymmer när man tittar på bäst-före datumet på en gammal konservburk. Om man inte har en aning om vilket format ett datum har, kan man behöva pröva alla möjligha betydelser och, för att vara på den säkra sidan, välja det tidigaste giltiga datumet.

För att ett datum ska vara giltigt måste förstås månaden vara mellan 1 och 12 och dagen mellan 1 och antalet dagar i månaden. Antalet dagar i de tolv månaderna är i tur och ordning

$$31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31$$

utom under skottår (vilka, för perioden $2000 - 2099$, infaller om årtalet är jämnt delbart med 4) då februari har 29 dagar.

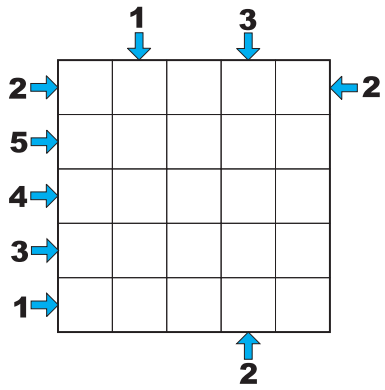
Skriv ett program som läser in de tre delarna av ett datum och skriver ut det tidigaste giltiga datumet som indata kan tänkas representera.

Körningsexempel:

```
Del 1: 3
Del 2: 5
Del 3: 1

År 2001, månad 3, dag 5.
```

UPPGIFT 2 – KVARTERET



FIGUR 1.

Figuren visar ett kvarter med totalt 25 hus uppförda på en kvadratisk tomt med 5×5 hus. Husen har höjderna $10, 20, 30, 40$ och 50 våningar, och måste vara placerade på ett sådant sätt att det i varje rad och kolumn finns exakt ett hus av varje höjd. Pilarna och talen anger hur många hus man kan se i den raden eller den kolumnen om man tittar i pilens riktning. Lägre hus skyms ju av högre!

Skriv ett program som med hjälp av den givna informationen bestämmer och skriver ut de 25 husens höjd i antal våningar som en kvadrat med 5 rader och 5 kolumner.

Indata är filen `kvarter.dat`, som innehåller 20 tal på en rad, separerade med blanksteg, antalet hus man kan se i en viss riktning, eller talet 0 som innebär att information i den riktningen saknas. Talen ges medurs med början högst upp till vänster.

Körningsexempel:
Filen `kvarter.dat` med innehållet:

```
0 1 0 3 0 2 0 0 0 0 0 2 0 0 0 1 3 4 5 2
```

ska ge svaret:

```
40 50 10 20 30
10 20 30 40 50
20 30 40 50 10
30 40 50 10 20
50 10 20 30 40
```

UPPGIFT 3 – ÖVERRASKANDE STRÄNGAR

Betrakta strängen "ABCBACC". På två ställen i strängen förekommer bokstaven 'A' två positioner före bokstaven 'C'. En sträng kallas *överraskande* om det **inte** finns två bokstäver, x och y, i strängen så att de förekommer två gånger (eller fler) i samma ordning och på samma avstånd från varandra. Den nämnda strängen är alltså inte en överraskande sträng. Ej heller är "ABACA" det, då bokstaven A förekommer två steg före bokstaven A (överlapp räknas alltså). Däremot är "ABACB" och "AABA" överraskande strängar.

Skriv ett program som tar som indata en överraskande sträng, och som utdata returnerar nästa överraskande sträng i alfabetisk ordning. Indatasträngen kommer endast bestå av stora bokstäver mellan A ... Z (max 20 tecken). Den nya strängen får inte innehålla några andra bokstäver än de som finns i indatasträngen (även om antalet av en viss bokstav kan vara fler eller färre).

Körningsexempel:

```
Överraskande sträng ? AABAC
Nästa överraskande sträng: AABC
```

UPPGIFT 4 – ANAGRAM

Ett ord är anagram till ett annat ord, om bokstäverna i det första har kastats om i det senare. *adrenalin* är anagram till *ledarinna* och *läsekrets* är anagram till *stärkelse* för att ta två ganska långa ord som exempel. Men ibland kan ett ord ha flera anagram, som till exempel *speldator, ledarpost datorspel* och *repsoldat*. Vi säger att vi funnit en grupp bestående av 4 anagram. Skriv ett program, som för givna ord tar reda på den största gruppen av anagram.

Programmet ska läsa listan med ord från filen `anagram.dat`. Filen inleds med ett tal n, ($n \leq 30000$), som anger antalet ord. Därefter följer n rader med ett ord på varje rad. Varje ord innehåller ≤ 16 bokstäver valda bland versalerna `A...Z`. Programmet ska först skriva ut hur många anagram den största gruppen innehåller, och därefter själva orden, i godtycklig ordning.

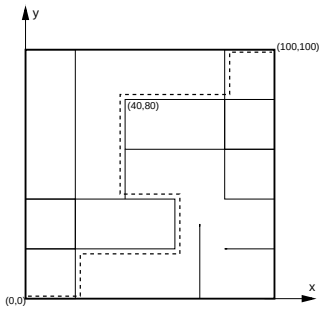
Körningsexempel:
Filen `anagram.dat` med innehållet

```
14
VELA
LEVT
ELEV
AVEL
LEVE
LAVE
ALV
ELVA
VAL
LEVA
VALL
LAV
VALE
TELE
```

ska ge svaret:

```
Den största gruppen innehåller 6 ord.
ELVA
AVEL
VALE
VELA
LEVA
LAVE
```

UPPGIFT 5 – STADEN



FIGUR 2. *Ett exempel på hur staden kan se ut. Om Boel följer den streckade linjen behöver hon bara korsa tre gator, vilket är det minsta möjliga antalet i detta fall. Observera att det aldrig kan göra någon skillnad om hon följer gatorna eller om hon sneddar.*

Boel bor och arbetar i den övertrafikerade staden Tutstad. Staden har en kvadratisk form och är placerad i ett koordinatsystem så att dess sydvästra hörn, där Boel bor, ligger i punkten $(0, 0)$ och dess nordöstra hörn, där Boel jobbar, ligger i punkten $(100, 100)$. Längs stadens fyra ytterkanter går så avskyvärda trafikleder att man omöjligt kan passera dem till fots. Men även inne i staden är trafiken besvärlig, vilket har fått Boel att undra hur många gator hon egentligen måste korsa när hon går till fots mellan hemmet och arbetet.

Skriv ett program som, givet stadens gatunät, bestämmer det minsta antalet gator man måste korsa för att ta sig från punkten $(0, 0)$ (innanför trafiklederna) till punkten $(100, 100)$ (innanför trafiklederna).

Indata läses från filen `tutstad.dat`. På första raden står ett heltal n ($1 \leq n \leq 5000$), antalet gator i staden. Därefter följer n rader med fyra heltal, x_1, y_1, x_2 och y_2, på varje rad. x_1 och y_1 är koordinaterna för startpunkten för en gata, medan x_2 och y_2 är koordinaterna för slutpunkten för gatan. För samtliga koordinater gäller att $0 \leq x_1 \leq x_2 \leq 100$ och $0 \leq y_1 \leq y_2 \leq 100$, samt att antingen $x_1 = x_2$ eller $y_1 = y_2$ (men inte både och). Varje gata är en rak sträcka mellan dessa punkter. Gatornas bredd är försumbar, så det går alltså att passera mellan två öst-västliga gator med y-koordinater y och $y + 1$.

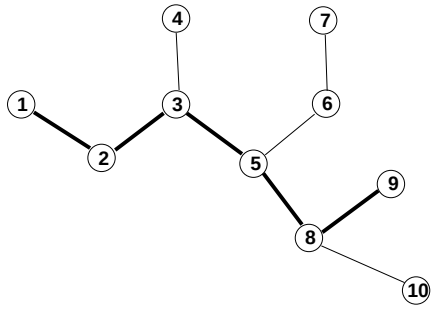
Körningsexempel: Filen `tutstad.dat` med innehållet:

```
11
20 0 20 100
0 20 60 20
60 20 60 40
0 40 60 40
70 0 70 30
80 20 100 20
80 40 100 40
80 40 80 100
40 40 40 80
40 60 100 60
40 80 100 80
```

ska ge svaret:

```
Minsta antal korsade gator: 3
```

UPPGIFT 6 – NÄTVÄRK



FIGUR 3. *Ett nätverk med 10 datorer. Om fiberkabeln dras via dator 1, 2, 3, 5, 8 och 9 (markerat med tjocka linjer) kommer alla datorer utom nummer 7 att ligga högst en länk ifrån fiberkabeln, vilket är optimalt.*

Bandbredden på företaget Caterpillar AB's intranät ska utökas. Deras datornätverk består av n datorer som är sammankopplade med $n - 1$ länkar, så att alla datorer kan kommunicera med varandra. IT-avdelningen har fått i uppdrag av styrelsen att ersätta en del av dessa länkar med en ny tjock fiberkabel. Kabeln får inte förgrena sig, men kan vara hur lång som helst.

Alla datorer som ligger längs kabeln eller högst en länk ifrån denna kan utnyttja den extra bandbredden. Eftersom Caterpillar AB är ett stort företag med många datorer är det inte helt enkelt att bestämma var den ny fiberkabeln ska dras, så att flest antal datorer kan utnyttja den högre bandbredden. Detta har orsakat stor huvudvärk hos IT-avdelningen, så de har därför outsourcat problemet åt dig.

Programmet ska läsa datornätverkets struktur från filen `network.dat`. Första raden i filen innehåller talet n (mellan 1 och 2500). Därefter följer $n - 1$ rader som beskriver länkarna mellan datorerna. Varje sådan rad består av två heltal mellan 1 och n, numren på datorerna som länken förbinder. Programmet ska skriva ut maximalt antal datorer som kan utnyttja den utökade bandbredden om fiberkabeln dras optimalt.

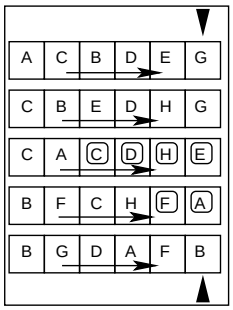
Körningsexempel:
Filen `network.dat` med innehållet:

```
10
1 2
2 3
3 4
3 5
5 6
6 7
5 8
8 9
8 10
```

ska ge svaret:

```
Maximalt antal datorer: 9
```

UPPGIFT 7 – GODISAUTOMATEN



FIGUR 4. *Ett exempel på hur godisautomaten kan vara laddad. För att få tag i varorna A, C, F och D måste Herbert minst köpa de sex inringade varorna.*

En viss godisautomat består av v ”våningar”, där varje våning består av en roterbar skiva. Varje våning är indelad i ett antal celler, där varje cell innehåller en vara. Figuren ovan visar hur det kan se ut. I automatens glas kan man se ett visst antal c av cellerna på varje våning, vi kan anta att övriga celler är tomma. Man kan köpa en vara från vilken våning som helst, men bara från den cell som för tillfället befinner sig längst till höger. När denna vara köpts roterar skivan så att den vara som fanns i cellen närmast till vänster om den köpta blir möjlig att köpa nästa gång.

Automaten kan innehålla upp till 26 olika sorters varor, var och en benämnd med en bokstav i intervallet $A \ldots Z$. Varorna kan vara utspridda hur som helst.

Herbert har glömt äta lunch och skulle vilja köpa ett antal olika varor. Men för att lyckas med detta måste han troligtvis köpa en del andra varor som han inte alls har något behov av. Skriv ett program som, givet varornas placering i automaten, beräknar det minimala antalet oönskade varor Herbert måste köpa för att få tag på de varor han verkligen vill ha. Observera att om han måste köpa flera exemplar av de önskade varorna så räknas alla utom ett exemplar som oönskade varor.

Indata läses från filen godis.dat. På första raden står heltalen v, c och n, ($1 \leq c, v \leq 100$ och $1 \leq n \leq 12$), där v är antalet våningar, c antalet synliga celler på varje våning och n är antalet varor som Herbert vill ha. Därefter följer v rader, vardera bestående av en sträng med c tecken i intervallet $A \ldots Z$. Var och en av dessa rader beskriver en våning, och ordningen på bokstäverna beskriver innehållet i cellerna på denna våning, från vänster till höger. Slutligen följer en rad med n olika tecken i intervallet $A \ldots Z$. Varje tecken är namnet på en vara som Herbert vill ha. Alla dessa varor finns i automaten.

Körningsexempel: Filen `godis.dat` med innehållet:

```
5 6 4
ACBDEG
CBEDHG
CACDHE
BFCHFA
BGDAFB
ACFD
```

ska ge svaret:

```
Antal onödiga varor: 2
```