UPPGIFT 1 – PRESIDENTVALET

När man ska välja president i det lilla landet på andra sida jorden, görs det av några få röstberättigade *väljare* v, ett udda antal i intervallet $1 \leq v \leq 999$. De har $k, 2 \leq k \leq 10$ *kandidater* att välja mellan. Varje kandidat har tilldelats en stor bokstav som beteckning, från A upp till J om det finns 10 kandidater.

Varje väljare har rangordnat kandidaterna efter eget gottfinnande. Vi tänker oss en följd av beteckningar, till exempel då antalet kandidater är 5, CADEB, där just den här väljaren anser att C passar bäst som president och att B passar sämst.

Själva valproceduren äger rum i ett antal steg, där man matchar två kandidater i taget mot varandra. Segraren i varje steg går vidare till nästa och efter $k-1$ steg har en president korats.

När till exempel en väljare med preferensordningen CADEB ska avge sin röst i matchen mellan kandidaterna A och E väljer han förstås kandidat A eftersom denne finns ”högre upp” (längre till vänster) på hans lista.

Nu visar det sig att det spelar stor roll i vilken ordning kandidaterna matchas i de olika stegen, för vem som till sist ska koras till president.

Skriv ett program som med hjälp av väljarnas preferenser på filen `VAL.IN` bestämmer en matchordning, som leder fram till seger för en given kandidat.

Indata: Första raden på filen innehåller bokstaven på den segrare som önskas. Den andra raden anger värdet på k, antal kandidater. Talet på tredje raden anger värdet på v, antal väljare. På efterföljande v rader finns en k tecken lång sträng, som anger preferensordningen för en väljare.

Utdata: En rangordning av kandidaternas beteckningar, som motsvarar en matchordning som leder fram till önskad president. Det finns förstås alltid flera matchordningar, som leder till samma resultat, om inte annat så kan ju de två första beteckningarna kastas om, utan att det påverkar utgången.

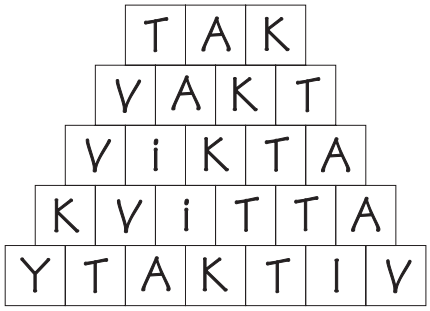
Exempel:

```
En matchordning, som leder till målet: FEDABC
```

Först möts alltså F och E. Segraren där går vidare till nästa steg där han möter D och så vidare.

UPPGIFT 2 – ORDJAKTEN

På TV4 har under våren ett nytt spel, kallat *Ordjakten* introducerats i programmet *Pussel*. Det handlar om en triangel av ord som ska fyllas i, som nedan:



FIGUR 1. *Ett exempel på Ordjakten*

Det första trebokstaviga ordet, TAK, är givet. Nästa ord, VAKT, innehåller alla bokstäver från ordet ovan, plus ytterligare en, V. På samma sätt ingår alla bokstäverna från VAKT i nästa ord, VIKTA. Här har bokstaven I lagts till. Och så vidare till ordet YTAKTIV (ämne, som sänker ytspänningen). Regeln är alltså: *Första ordet, med tre bokstäver, är givet. Därefter bildas varje ord med hjälp av bokstäverna i ordet ovanför, plus en valfri bokstav.*

Skriv ett program som tar emot ett ord och skriver ut samtliga lösningar tillsammans med en rad som anger antalet lösningar. Orden hämtas från filen `ORDEN.DAT` som innehåller 17285 ord från SAOL (Svenska Akademiens OrdLista) med tre till sju bokstäver. Vi har tagit bort ord som innehåller å,ä och ö och använder dessutom enbart gemener (små bokstäver).

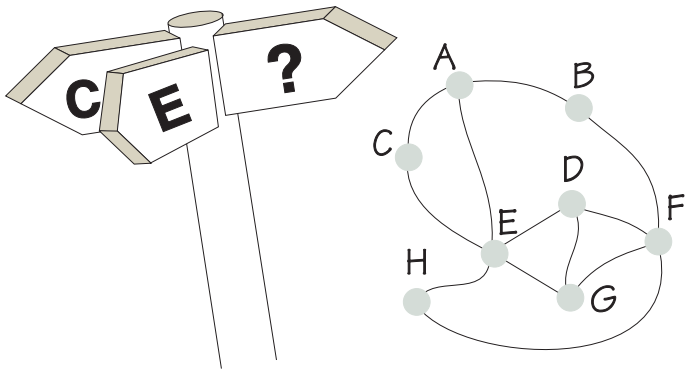
Indata: Ett ord med tre bokstäver (som för övrigt finns i filen).

Utdata: Alla lösningar som finns för det givna ordet. Varje lösning, med det givna ordet tillsammans med de fyra eftersökta, på en rad, samt en sista rad, som talar om hur många lösningar det finns.

Exempel:

```
Startord: abc
abc | back | backa | barack | backpar |
abc | back | backa | barack | byracka |
abc | back | backa | bracka | backpar |
abc | back | backa | bracka | byracka |
abc | back | black | blicka | anblick |
abc | back | black | blicka | billack |
abc | back | black | blicka | blackig |
abc | back | black | blocka | blockad |
abc | back | bocka | barock | badrock |
abc | back | bocka | barock | boricka |
abc | back | bocka | blocka | blockad |
Det finns 11 lösningar
```

UPPGIFT 3 – VÄGVISARE



FIGUR 2. *Till vänster ser du skylten som ska stå i stad A i exemplet, till höger motsvarande karta*

I landet finns 8 större städer med namnen, A, B . . . H, som alla ska få nya vägvisare. Även om de är nya, så är de inte fullständiga. Landet har helt enkelt inte råd att ge bilisterna den information de behöver för att säkert finna vägen till den stad de söker.

I fabriken, finns nu de 8 skyltarna. Dels saknar de alla varsin pil. De har alltså en pil mindre än det finns anslutande vägar till staden. Dels saknas de uppgifter om i vilken stad de ska placeras.

Skriv ett program, som med hjälp av informationen i filen `SKYLTAR.IN` bestämmer i vilken stad de olika vägvisarna ska placeras.

Indata: Filen består av 8 rader. På varje rad finns en sträng som anger vilka namn som finns på vägvisarens pilar.

Utdata: En sträng med namnen (bokstäverna) på de 8 städerna i vilka vägvisarna ska stå. Första bokstaven motsvarar namnet på den stad vars vägvisare beskrivs på första raden i filen och så vidare.

Exempel:
Indata: Filen har följande innehåll

```
CE
ACDH
F
DE
BGH
F
E
FG
```

Utdata:

```
Vägvisarna placeras i AEHGFBCD
```

UPPGIFT 4 – PERSONNUMMER

Personnummer är praktiskt att använda för att identifiera personer. Ibland kan det dock vara en fördel om man bara behöver ange en del av personnumret, för att unikt bestämma personen i en sluten grupp. Din uppgift är att skriva ett program som, givet en grupp människor och deras respektive personnummer, finner den kortaste sammanhängande sekvensen av siffror i varje persons personnummer, så att den sekvensen inte förekommer i någon annan persons personnummer.

Indata: Programmet ska läsa indata från filen `PERSON.IN`. Första raden i filen innehåller ett heltal $n, 1 \leq n \leq 10000$, som är antalet människor i gruppen. Därefter följer n rader i filen, som var och en innehåller ett personnummer. Varje personnummer består av exakt 10 siffror och kan ha inledande nollor (notera dock att personnumren inte behöver vara giltiga svenska personnummer).

Utdata: Programmet ska skapa en ny fil, `PERSON.UT`, till vilken utdata ska skrivas. Filen ska innehålla n rader, en rad för varje personnummer i indatafilen (i samma ordning). Varje rad ska innehålla den kortaste, sammanhängande sekvensen av siffror från personens personnummer, som inte förkommer i någon annans personnummer. Du kan anta att det aldrig krävs fler än 5 siffror från ett personnummer.

Exempel:
Indata:

```
5
0101010101
0123456789
0123456787
2348723434
8765432111
```

Utdata:

```
10
9
787
48
76
```

Kommentar: Det finns även andra giltiga lösningar.

UPPGIFT 5 – KOLLEKTIVTRAFIK

Anders planerar att bosätta sig i Lokstad, berömt för sin väl utbyggda kollektivtrafik med osedvanligt regelbundna avgångstider. Han har gjort en lista över möjliga bostad-arbetsplats-kombinationer, och din uppgift är nu att skriva ett program, som med hjälp av filen `LOKSTAD.IN`, avgör vilken av dessa kombinationer som är bäst ur kollektivtrafiksynpunkt.

Vi betraktar Lokstad som ett nätverk av n tågstationer numrerade från 1 till n. En tåglinje går direkt från en tågstation till en annan (endast en riktning). Varje tåglinje karakteriseras dessutom av tre heltal s, i och t. s är avgångstiden för det första tåget på linjen, given som antal minuter efter klockan $8:00$. i är intervallet mellan tågen och t är tiden för resan. Om till exempel en tåglinje från X till Y har $s = 6, i = 17$ och $t = 24$, betyder det att det första tåget avgår $8:06$ och kommer fram $8:30$, nästa tåg går $8:23$ och kommer fram $8:47$, nästa går $8:40$ och så vidare.

Vi räknar med att tågbyten sker snabbt, det vill säga, om man ankommer $8:40$ till en station kan man också hinna med ett annat tåg som avgår $8:40$. Anders kommer alltid till stationen närmast bostaden B klockan $8:00$ (det vill säga tiden tiden 0). Ditt program ska bestämma vid vilken tidpunkt han tidigast kan vara vid stationen vid arbetet A.

Indata: På första raden på filen finns två heltal $n, 2 \leq n \leq 500$ och $m, 1 \leq m \leq 5000$, där n är antal tågstationer och m antal tåglinjer. Därefter följer m rader med fem heltal $X, Y, s, (0 \leq s \leq 59), i, (1 \leq i \leq 60)$ och $t, (1 \leq t \leq 60)$ på varje rad. Varje rad beskriver en tåglinje från station X till station Y med s, i och t beskrivna ovan. Från varje tågstation utgår högst 10 tåglinjer. Sedan följer en rad med ett heltal $k, (k \leq 100)$, antal bostads-arbetsplats-kombinationer, och därefter k rader med två heltal B och A, start- och slutstation. Det är alltid möjligt att ta sig från B till A.

Utdata: Först talen B och A i den kombination som gav kortast restid och sedan denna tid i minuter.

Exempel:
Indata:

```
3 4
1 2 4 9 18
1 3 8 4 20
2 3 6 16 4
3 1 0 10 5
2
3 2
1 3
```

Utdata:

```
Bästa kombinationen: 1 till 3. Tid: 26 minuter
```

UPPGIFT 6 – BILJETTPLANERING

Algot och hans hustru Ritma brukar besöka Stockholm då och då för att handla. Väl i Stockholm använder de sig av Stockholms Lokaltrafik, SL, för att ta sig mellan olika platser. Eftersom paret är väldigt snålt planerar de sina besök så noga som möjligt för att minimera antalet resor med SL. Sedan måste de räkna ut vilka biljetter de ska köpa - det finns ju så många typer att välja mellan! De sorters biljetter som Algot och Ritma kan tänka sig att köpa är *rabattkuponger*, 1-*dygnskort*, 3-*dygnskort* och 30-*dagarskort*.

Planeringen sköter Ritma, och hon är redan klar. Hon har räknat ut exakt hur många resor de ska göra i Stockholm under vissa dagar för en tid framåt. Nu är det upp till Algot att planera inköpet av biljetter för dessa resor så att kostnaden minimeras. Detta är inte alls så lätt för stackars Algot, som nu ber om din hjälp.

Din uppgift är att skriva ett program som läser in hur många resor varje dag i Stockholm som paret planerar att göra, och skriver ut kostnaden för det billigaste sättet de kan köpa biljetter på. För enkelhetens skull antar vi att 1-*dygnskort*, 3-*dygnskort* och 30-*dagarskort* börjar gälla den dag kortet köps och de direkt följande 2 eller 29 dagarna (om man köpt ett 3-*dygnskort* eller 30-*dagarskort*). Om man använder *rabattkuponger* för en resa, så tar en resa exakt 3 kuponger. Eventuella överblivna *rabattkuponger* kan inte säljas. *Rabattkuponger* har obegränsad livstid (det vill säga, alla *rabattkuponger* som behövs kan köpas redan första dagen).

Prislista **Typ**	**Antal**	**Pris**
Rabattkuponger	20	110
Rabattkuponger	10	60
30-*dagarskort*	1	500
3-*dygnskort*	1	150
1-*dygnskort*	1	80

Indata: Programmet ska läsa indata från filen `BILJETT.IN`. Första raden i filen innehåller ett heltal $n, 1 \leq n \leq 100000$, som är antalet dagar framåt som Ritma planerat. Därefter följer n stycken tal mellan 0 och 10 som för varje dag anger antalet resor som planeras för denna dag. Talen separeras med blanksteg eller ny rad.

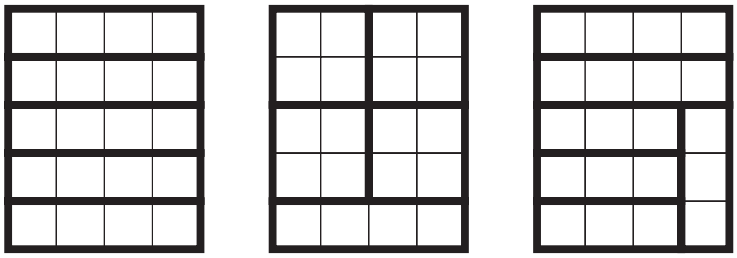
```
10
1 0 0 4 0 0 0 3 3 3
```

Utdata: Skriv ut, på skärmen, ett enda tal; kostnaden för det billigaste sättet att köpa biljetterna på, för att kunna genomföra alla planerade resor.

```
250
```

Kommentar: Resorna på dag 4, i exemplet, betalas med ett 1-*dygnskort*, de övriga resorna med *rabattkuponger* (total 30 krävs). Kostnaden blir $80+110+60 = 250$ kr. Notera om att sista dagen krävt 4 resor istället, hade den optimala lösningen varit ett 3-*dygnskort* och 20 *rabattkuponger* $150+110 = 260$ kr.

UPPGIFT 7 – TOMTINDELNING



FIGUR 3.

På grund av den exploderande inflyttningen till Älgträsk efter PO-kvalet behöver man fler villatomter. Ett rektangulärt träskområde ska därför (efter torrläggning) delas upp i ett antal mindre rektangulära tomter. Träsket är indelat i kvadratiska rutor och av någon byråkratisk anledning måste de nya tomtgränserna följa rutnätet. Din uppgift är att skriva ett program som utför uppdelningen på ett sådant sätt att *skillnaden i area mellan den största och den minsta tomten blir så liten som möjligt.*

Antag exempelvis att träskområdet är 5×4 rutor. Om det ska delas upp i 5 tomter kan detta göras som till vänster i figur 3. Varje tomt har här samma area, vilket naturligtvis är den bästa lösningen. En precis lika bra uppdelning visas i mitten i figur 3. Tomternas form spelar alltså ingen roll, bara de följer rutnätet och är rektangulära. Om vi däremot vill ha 6 tomter går det inte att göra dessa lika stora eftersom 20 inte ens är delbart med 6. En optimal lösning i detta fall visas till höger i figur 3. Här har tomterna areorna 3 och 4 vilket ger skillnaden 1. Även här finns flera likvärdiga lösningar.

Indata: Programmet ska läsa in två heltal $r, (1 \leq r \leq 14)$ och $c, (1 \leq c \leq 14)$ som är antalet rader respektive kolumner som träskområdet är inrutat i samt ett heltal $n, (1 \leq n \leq 26)$; antalet tomter som området ska delas upp i.

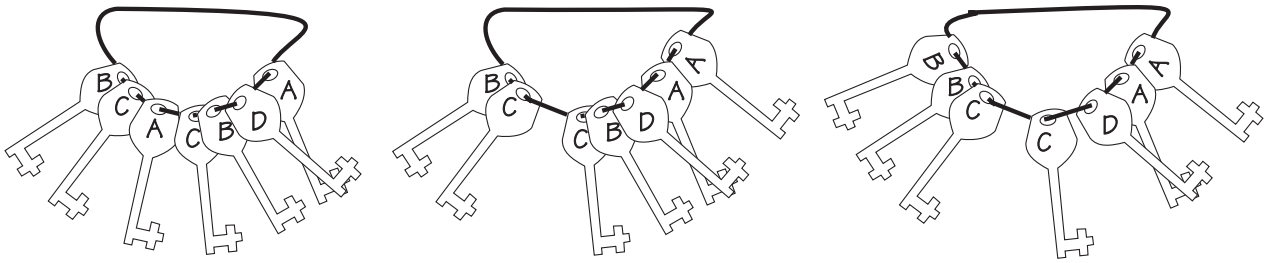
Utdata: Programmet ska först skriva ut minsta möjliga skillnad i area mellan den största och minsta tomten. Därefter ska en karta skrivas ut, vilken visar ett möjligt sätt att göra en sådan bästa uppdelning. Använd bokstäverna $A, B, C \ldots$ för att ”numrera” de olika tomterna. Ordningen på denna numrering är naturligtvis valfri.

Exempel:

```
Områdets storlek (rader): 9
Områdets storlek (kolumner): 8
Antal tomter: 5

Minsta skillnad: 2
AABBCCDD
AABBCCDD
AABBCCDD
AABBCCDD
AABBCCDD
AABBCCDD
AABBCCDD
EEEEEEEE
EEEEEEEE
```

UPPGIFT 8 – NYCKELRINGEN



FIGUR 4. *För att lösa det här problemet börjar vi med att ta ut den azurblå nyckeln mellan de två chockrosa (vänstra figuren) och sätter in den invid den andra azurblå (mittfiguren). Därefter tar vi ut den buskgröna som sitter intill den djupblå och sätter in den intill den andra buskgröna (högra figuren). Efter två förflyttningar är målet nått.*

Klas har många nycklar på sin nyckelring. För att hålla reda på dem har han märkt var och en med en tejpbit, som kan vara *Azurblå* (A), *Buskgrön* (B), *Chockrosa* (C), *Djupblå* (D), *Eldröd* (E), eller *Fibblegul* (F). En dag kom han på att det skulle vara mycket snyggare om alla nycklar med samma färg satt i följd på nyckelringen, och han började fundera på hur många nycklar som skulle behöva flyttas för att åstadkomma detta.

Din uppgift är att skriva ett program, som givet nycklarnas ursprungliga ordning på ringen, räknar ut det *minsta antalet* nyckelförflyttningar, som behövs för att uppnå en ordning där alla nycklar med samma färg sitter i följd. Vilken ordning färgerna kommer i spelar ingen roll. En nyckelförflyttning innebär att man plockar bort en valfri nyckel från ringen och sätter in den igen mellan två andra nycklar. Ett exempel visas i figur 4 ovan.

Indata: Programmet ska läsa in en rad med högst 60 bokstäver A...F, där varje bokstav anger färgen på en nyckel och nycklarna sitter i den ordning de kommer på raden. Den första och sista nyckeln sitter alltså bredvid varandra eftersom ringen är sluten.

Utdata: Programmet ska skriva ut det minsta antalet förflyttningar som behövs.

Exempel:

```
Nyckelsekvens: BCACBDA
Minsta antal flyttningar: 2
```