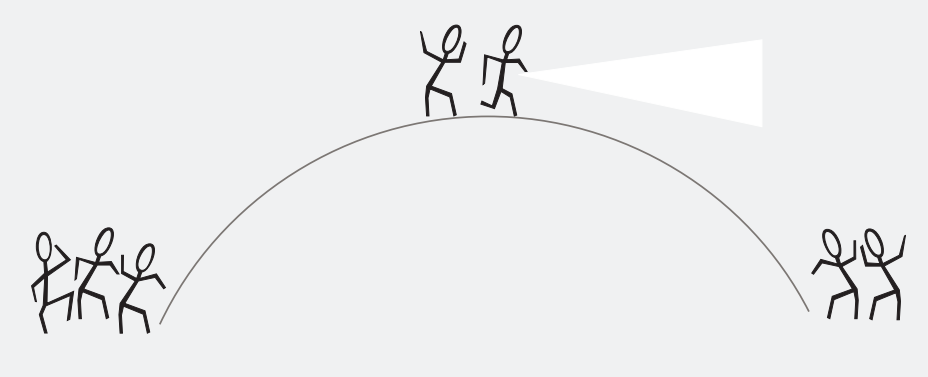


FIGUR 1.

UPPGIFT 1 – NATTLIG FÖRFLYTTNING

På ena sidan av en bro, en mörk höstnatt, befinner sig $n, 3 \leq n \leq 7$ personer, som alla vill till andra sidan av den smala bron. Eftersom det är mörkt, behövs en *ficklampa* vid övergången. Gruppen har bara **en** och då man vill göra passagen så säker som möjligt, kan högst två personer åt gången, ta sig över. Det kan ta olika lång tid för personerna att ta sig över bron. När två personer går tillsammans är det alltid den långsammaste som bestämmer farten.

Skriv ett program som tar emot data om gruppens storlek n och tiden (i minuter) för varje individ att ta sig över bron, $t_1, t_2 \ldots t_n$, och som sedan bestämmer den kortaste tid det tar för hela gruppen att ta sig över bron.

Indata: En dialog liknande denna.

```
Gruppstorlek : 4
Tid för person A : 1
Tid för person B : 5
Tid för person C : 2
Tid för person D : 10
```

Utdata: Endast en rad

```
Övergången tar 17 minuter
```

Kommentar: Det finns nästan alltid flera möjligheter, som leder till samma ”bästatid”. Så här kan det gå till i det givna exemplet med den förbrukade tiden efter varje steg inom parentes. A och C går över (2). A går tillbaka (3). B och D går över (13). C går tillbaka (15). A och C går över (17).

UPPGIFT 2 – ORDLEK

På planeten *Kompilatus* talas ett språk, vars ord har ovanligt väldefinierade *stavningsregler*. Utan att egentligen förstå språket kan man därför direkt avgöra om en bokstavskombination är ett giltigt ord eller inte.

Alfabetet består av 7 bokstäver (A–G) som är uppdelade i *pedaler* (A, D, G) och *folofjanter* (B, C, E, F). Följande fem stavningsregler finns:

- En *pedal* får inte följas av en *pedal*.
- Ett B måste följas av en *folofjant*, men denna får inte vara ett annat B.
- Ett C måste följas av F eller G.
- Ett E måste följas av en *pedal*.
- Ett ord måste sluta på A eller G.

Ett av de mest populära TV-programmen på Kompilatus är en ordlek där man får en rad med bokstäver som man sedan ska ändra ordning på, så att de bildar ett giltigt ord, det vill säga ett ord som följer de fem stavningsreglerna ovan. Om alla bokstäver inte kan användas ska man hitta det längsta giltiga ordet som kan bildas av de givna bokstäverna.

Din uppgift är att hjälpa de tävlande att hitta det längsta giltiga ordet. Om det finns flera ord med maximal längd kan du skriva vilket som helst av dem.

Indata: En sträng med de givna bokstäverna, maximalt 13 stycken. Varje bokstav kan vara A, B, C, D, E, F eller G (endast stora bokstäver).

```
Givna bokstäver: BBACFCB
```

Utdata: Det längsta, giltiga, ordet som kan skrivas (eller ett av dem)

```
Längsta ord: BCFA
```

UPPGIFT 3 – WILDCARDS

I kommandorader till *operativsystem* är det vanligt att man använder så kallade *wildcards* i namn, för att fånga upp en viss kategori eller typ av filer. Skriver man till exempel in *söksträngen* `A*.EXE` *selekterar* man alla filer, med ett namn som börjar på `A` och som har filtillägget `EXE`. Asterisken (`*`) är här ett wildcard, som ersätter alla tillåtna fortsättningar på ett filnamn, från 0 tecken och uppåt. Frågetecknet (`?`) förekommer också som wildcard, och står för *precis ett tecken*. *Asterisk* och *frågetecken* kan kombineras tillsammans med andra tecken i en söksträng.

Skriv ett program som, läser ord från en fil, och som för en given söksträng, listar de ord på filen som selekteras av söksträngen.

Indata: Programmet frågar efter en söksträng, bestående av $n, (1 \leq n \leq 20)$ tecken ur mängden $[A \ldots Z, \_, ?, *]$. Två asterisker står aldrig i rad. Orden som ska undersökas finns på filen `namn.dat`, som innehåller namnen på 257 större städer i världen. Varje rad innehåller ett ord med ≤ 20 tecken – versaler (stora bokstäver) $A \ldots Z$, samt eventuellt _ (underscore).

```
Söksträng: *A??C*
```

Utdata: En lista med de namn (ett på varje rad), i godtycklig ordning, som selekterats ut av programmet. Utskriften avslutas med ett tal som anger antalet ord, som återfunnits.

```
CARACAS
KARACHI
VATICAN_CITY
3
```

Kommentar: Genom några ytterligare exempel kommer Du att förstå hur programmet ska fungera. Inom parentes återfinns antalet selekterade namn:

`*`	Listar samtliga namn (257)
`S*`	Listar samtliga namn som börjar på S (28)
`*A`	Listar samtliga namn som slutar på A (54)
`*B*`	Listar samtliga namn som innehåller bokstaven B (61)
`*FF*`	Listar samtliga namn som innehåller två F intill varandra (1)
`*S*T*`	Listar samtliga namn där bokstäverna S och T förekommer i denna ordning (26)
`??I??`	Namn på fem tecken med I i mitten. (2)
`*A?`	Listar alla namn där *näst* sista bokstaven är A (31)
`*A?A*`	Någonstans i namnet ska finnas två A med exakt ett tecken emellan (42)
`?*A*`	Listar alla namn som innehåller A från och med *andra* tecknet (186)
`??*A??*`	Här måste bokstaven A vara väl inbäddad (103)
`????`	Listar alla namn med exakt *fyra* tecken (15)
`??*???`	Listar alla namn med ≥ 5 tecken (242)

FIGUR 2. *Den första moderna alfametiken, publicerad av H E Dudeney i juli 1924 ser ut så här. (Lösning:* $9567 + 1085 = 10652$*)*

UPPGIFT 4 – ALFAMETIK

En *alfametik* är ett matematiskt pussel där en antal ord är ordnade på den form man använder när man utför en addition ”för hand”, och där det går ut på att ersätta bokstäverna med siffror så att resultatet blir en giltig aritmetisk summa.

Det finns två tämligen självklara regler för en alfametik:

- Det ska finnas ett $1 - 1$ förhållande mellan bokstäverna och siffrorna i alfametiken. Det vill säga, samma bokstav står alltid för samma siffra, och samma siffra är alltid representerad av samma bokstav.
- Siffran 0 får inte förekomma längst till vänster i termerna eller summan.

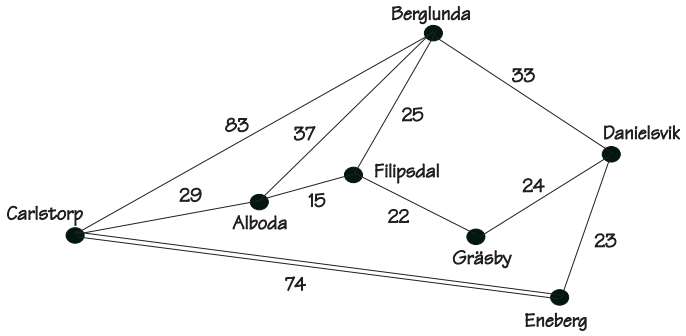
Dessutom, i den här uppgiften är det bara basen 10 (vanlig decimalräkning) som gäller.

Indata: Programmet ska först fråga efter antalet termer, ett tal mellan 2 och 10. Därefter läses termerna in, följt av summan. Inget av dessa ”tal” kommer att vara längre än 8 bokstäver. Bokstäverna kommer att vara uteslutande versaler, A till Z.

```
Antal termer: 4
Term 1: SATURN
Term 2: URANUS
Term 3: NEPTUNE
Term 4: PLUTO
Summa : PLANETS
```

Utdata: Programmet ska skriva ut en lösning till problemet. Om det finns flera lösningar räcker det att ange en av dessa. Du kan anta att det existerar minst en lösning.

```
127503+502351+3947539+46578=4623971
```



FIGUR 3. *Från början finns redan motorväg mellan Carlstorp och Eneberg*

UPPGIFT 5 – MOTORVÄGAR

Regeringen i *Motoristan* har beslutat att bygga om ett antal *riksvägar* till *motorvägar*, så att det är möjligt att ta sig mellan två godtyckliga städer i landet enbart genom att köra på motorvägar. Eftersom det är dyrt att bygga motorväg vill regeringen inte bygga om mer väg än nödvändigt, så målet är att minimera den totala *ombyggnadssträckan*.

Skriv ett program som läser in landets vägnät och beräkna vilka riksvägar som ska byggas om till motorvägar så att målet uppnås. En del av vägarna i landet kan dessutom redan vara motorvägar. Dessa behöver förstås inte byggas om. Det är inte tillåtet att bygga en motorväg mellan två städer som inte redan är direkt förbundna med en riksväg.

Indata: Programmet läser indata från filen `motor.dat`. Första raden i filen innehåller två heltal, $n, (2 \leq n \leq 200)$ som anger antalet städer i Motoristan, och $m, (1 \leq m \leq 5000)$ som anger antalet vägar. Därefter följer n stycken rader innehållande samtliga städer i Motoristan. Namnet på städerna är högst 15 tecken långt, och innehåller inga *blanksteg*.

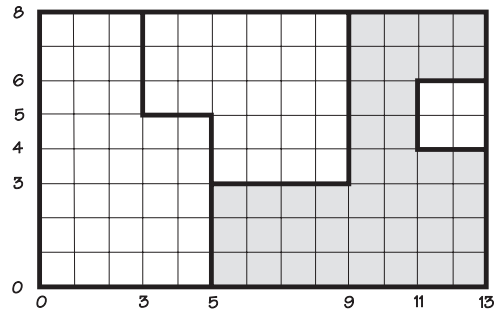
Sedan följer $4m$ rader som beskriver vägnätet. Varje grupp om fyra rader innehåller i tur och ordning namnet på två städer som är förbundna med en väg, längden på vägen (ett positivt heltal mellan 1 och 1000) samt ett tecken som anger om vägen är en riksväg, R, eller en motorväg, M.

Exempel på indatafil (här delad i fem spalter):

```
7 10          | 37          | Filipsdal   | 74          | Danielsvik
Alboda        | R           | Gräsby      | M           | Eneberg
Berglunda     | Alboda      | 22          | Berglunda   | 23
Carlstorp     | Filipsdal   | R           | Carlstorp   | R
Danielsvik    | 15          | Berglunda   | 83          | Danielsvik
Eneberg       | R           | Danielsvik  | R           | Gräsby
Filipsdal     | Alboda      | 33          | Berglunda   | 24
Gräsby        | Carlstorp   | R           | Filipsdal   | R
Alboda        | 29          | Carlstorp   | 25          |
Berglunda     | R           | Eneberg     | R           |
```

Utdata: Programmet ska skriva ut den minsta totalsträckan av riksväg som måste byggas om så att det går att ta sig mellan två godtyckliga städer genom att enbart köra på motorvägar.

```
Den minsta totalsträcka som måste byggas om är 109.
```



FIGUR 4. *Marken ovan är indelad i fyra områden med areorna 4, 26, 34 och 40*

UPPGIFT 6 – ETT MARKANT PROBLEM

Storbonden Sven äger ett stort staketinhägnat rektangulärt område mark. Han har låtit dela upp marken i flera mindre områden genom att bygga staket så att varje mindre område är helt inhägnat av staket. Samtliga staket är raka och placerade parallellt med ytterkanterna.

Nu ska Svens äldste son, Peter, gifta sig, och som bröllopspresent kommer han då få ett av dessa mindre områden. Peter vill självfallet ha det största området (sett till arean), men då Sven har varit klurig och delat in sin mark på ett komplicerat sätt är det inte helt enkelt att avgöra vilket område som är störst. Hjälp Peter att lösa problemet genom att skriva ett program som beräknar vilken del som har störst area.

Indata: Programmet ska läsa data från filen `mark.dat`. Filen inleds med tre heltal, $b, (0 < b \leq 40000)$, $h, (0 < h \leq 40000)$ och $n, (4 \leq n \leq 50)$ på första raden åtskilda av ett blanksteg. b och h anger bredden respektive höjden på Svens markområde och n anger antalet staket på Svens mark. Därefter följer n stycken rader med fyra heltal på varje rad, $x1, y1, x2, y2 (0 \leq x1, x2 \leq b, 0 \leq y1, y2 \leq h)$ som anger start- och ändpunkten på ett staket. Det nedre vänstra hörnet på Svens område har koordinaterna $(0, 0)$ och det övre högra koordinaterna (b, h).

Du kan anta att de fyra staketen, som inhägnar hela Svens område, är med i listan över staket, samt att inga staket korsar eller överlappar varandra. Exempel på fil (här delad i två spalter)

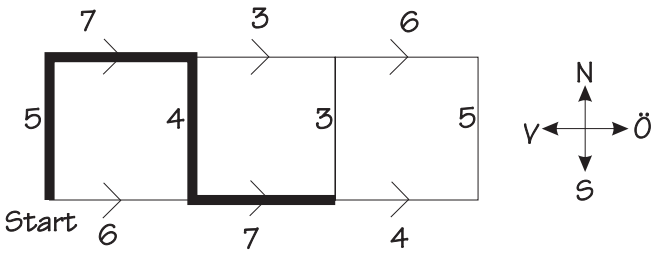
```
13 8 12
0 0 13 0
13 0 13 8
13 8 0 8
0 8 0 0
5 0 5 5
5 5 3 5
```

```
3 5 3 8
5 3 9 3
9 3 9 8
11 4 13 4
11 4 11 6
11 6 13 6
```

Utdata: Programmet skriver ut arean av det största området.

```
Det största området har arean 40 kvadratmeter.
```

Kommentar: (använd datatypen *long* i C/C++ eller *longint* i Pascal för att vara säker på att svaret ryms i variabeln)



FIGUR 5.

UPPGIFT 7 – SNÖPLOGNING

I sin vanmakt över det svenska vintervädret har politikerna i den långsmala staden *Rutnäs* givit upp målet att ploga alla gator efter ett snöfall. Istället satsar man på att ploga under en viss tid på ett sådant sätt att så många personer som möjligt har nytta av plogningen. Du ska skriva ett program som beräknar vilken rutt plogbilen ska åka.

Rutnäs har en mycket regelbunden struktur som visas i figuren ovan. Eftersom både stadens storlek och folks vanor förändras, vill man att programmet ska läsa in data om staden från en fil. Det kommer alltid att finnas exakt två öst-västliga gator, huvudgator, medan antalet tvärgator kan variera. Plogbilen startar alltid i sydvästra hörnet. På huvudgatorna får den bara köra österut, medan båda riktningarna är tillåtna på tvärgatorna.

Alla gator är från början oplogade. När plogbilen har kört en sträcka räknas den som plogad. Varje sträcka mellan två korsningar tar exakt en minut att köra och är försedd med ett heltal K som talar om hur många personer som använder den. Den totala nyttan för en rutt definieras som summan av K-värdena för de plogade sträckorna. Den bästa rutten (d.v.s. med högst total nytta) för fyra minuters plogning i exemplet är markerad i figuren. Den har total nytta 23.

Observera att det är tillåtet att köra tillbaka på en redan plogad tvärgata, men det ger naturligtvis inget bidrag till den totala nyttan.

Indata: Programmet ska läsa indata från filen `plog.dat`

Första raden i indatafilen innehåller två heltal $N, (2 \leq N \leq 80)$ och $T, (1 \leq T \leq 3N)$, där N är antalet kvarter (rutor) i staden och T är antal minuter som plogningen maximalt får ta. Sedan följer en rad med N heltal; K-värdena för sträckorna på den norra huvudgatan. Därefter en rad med $N + 1$ heltal; K-värdena för tvärgatorna. Slutligen en rad med N heltal; K-värdena för sträckorna på den södra huvudgatan. K-värdena ligger i intervallet $[1 \ldots 100]$

```
3 4
7 3 6
5 4 3 5
6 7 4
```

Utdata: Den totala nyttan för den bästa rutten plogbilen kan köra.

```
Bästa totala nyttan: 23
```

UPPGIFT 8 – FLYKTEN TILL VADSTÄLLET

Frodo, Sam, Merry, Pippin och *Vidstige* måste, så snabbt som möjligt, ta sig från staden *Bri* till alvernas boning i *Vattnadal*. De är på flykt undan de *Svarta Ryttarna*, som patrullerar landsvägen dag och natt. Därför vågar de inte ge sig ut på landsvägen, utan måste vandra genom skog och mark.

Hjälp sällskapet att finna den snabbaste vägen från *Bri* till *Vattnadal*, givet en karta över området. Markerat på kartan är, förutom *Bri* och *Vattnadal*, olika typer av terräng: *gräs, skog, kullar* och *träsk*. Att färdas över en viss typ av terräng tar en given tid (se tabellen nedan).

Sällskapet kan bara röra sig i riktningar *nord, syd, väst* och *öst*. De får inte vistas på landsvägen (inte ens bara för att korsa den) i rädsla för att stöta på de *Svarta Ryttarna*. Det är inte tillåtet att gå utanför kartans gränser.

Indata: Programmet ska läsa indata från filen `karta.dat` Filen inleds med två heltal på samma rad (åtskilda av ett blanksteg), $b, (2 \leq b \leq 100)$ och $h, (2 \leq h \leq 100)$, som anger bredden respektive höjden på kartan. Därefter följer h rader med b tecken i varje rad som beskriver kartans utseende enligt följande:

Tecken	Förklaring	Tid	Tecken	Förklaring	Tid	Tecken	Förklaring	Tid
B	Bri	0	g	gräs	1	k	kullar	3
.	landsväg	–	V	Vattnadal	0	s	skog	2
t	träsk	4						

Det tar 0 tidsenheter att lämna *Bri* och komma till *Vattnadal*. Endast ovanstående tecken kommer att finnas med i kartbeskrivningen. Det kommer att finnas exakt ett B och ett V utplacerat på kartan, och det kommer att vara möjligt att ta sig från *Bri* till *Vattnadal* utan att behöva korsa landsvägen.

Exempel på indatafil:

```
10 5
ggggttgggg
gB.......g
gggkkkgg.g
ggsksgggVg
ggssssgggg
```

Utdata: Programmet ska skriva ut den kortaste möjliga tid det tar för sällskapet att ta sig från *Bri* till *Vattnadal*.

```
Snabbaste vägen tar 12 tidsenheter.
```

Kommentar: En möjlig väg med tid 12 är SÖSÖÖÖÖÖÖ med tiderna (0+1+1+2+3+2+1+1+1+0 = 12)