

# NIO 2009/2010 — Oppgaver for innledende runde

## Oppgave 1: Kodekneking

*Merk:* Dette er ment som en forholdsvis enkel oppvarmingsoppgave. Tradisjonen tro skal hver oppgave gi 10 poeng, men siden oppgaveteksten i praksis inneholder beskrivelsen av løsningsmetoden, gis det ikke poeng for algoritme eller bevis. I stedet gis det inntil 7 poeng for fungerende program (kjøring mot testdata) og inntil 3 poeng for lesbar og kommentert kode. I de andre oppgavene regnes poengene på vanlig måte.

Gutteklubben Grei har nettopp oppdaget at rivalene Ugreie Unger driver og pønsker ut onde planer for å sette Gutteklubben Grei ut av spill. Noen tapre medlemmer av Gutteklubben Grei har klart å snappe opp flere kodete meldinger fra Ugreie Unger. Nå må meldingene dekrypteres slik at de onde planene kan avsløres! I og med at medlemmene av Ugreie Unger ikke er kjent for å være de skarpeste knivene i skuffen, regner Gutteklubben Grei med at de har brukt den enkleste krypteringstypen som finnes: *Cæsar-kryptering*.

Cæsar-kryptering går ut på at man velger seg et nøkkeltall,  $k$ , og forskyver hver bokstav  $k$  plasser bortover i alfabetet. Hvis man når slutten av alfabetet hopper man rundt til starten igjen og fortsetter tellingen der. For eksempel: hvis  $k = 3$  vil **a** bli til **d** og **w** bli til **z**, og hvis man begrenser seg til det engelske alfabetet vil **x**, **y** og **z** bli til henholdsvis **a**, **b** og **c**. Dekryptering gjøres ved enten å skyve hver kodete bokstav  $k$  hakk bakover eller  $26 - k$  hakk forover. (Merk at hvis  $k = 13$  er det det samme om man går fremover eller bakover, så kryptering og dekryptering vil fungere på eksakt samme måte. Cæsar-kryptering med  $k = 13$  kalles for *ROT13-kryptering* (*ROT* står for “rotate”).)

### Input

Input inneholder én linje. Den er maksimalt 255 tegn lang, og inneholder bare små, engelske bokstaver (a til z) — ingen mellomrom, sifre eller spesialtegn.

### Output

Programmet ditt skal skrive ut 25 linjer, én for hver mulig verdi for  $k$  (unntatt 0). Hver linje skal starte med et tall, fra og med 25 ned til og med 1, som tilsier hvilken  $k$  man har forsøkt å dekryptere med. Deretter kommer et kolon og et mellomrom, fulgt av resultatet av dekrypteringen. Første linje skal altså være resultatet av å skyve den krypterte teksten ett hakk til høyre i alfabetet, og så går det hele tiden ett hakk til høyre for hver linje.

### Hint

I C, C++ og mange andre språk gjelder følgende:

- Hvert tegn har en unik tallkode.
- I C og C++ kan man ta et tegn (altså en `char`-verdi) og gjøre om til det tegnets tallkode ved å skrive `(int)` foran, og et tall kan gjøres om til tegnet som har den tallkoden med `(char)`.
- Man kan addere og subtrahere tegn og tall. Hvis `c` er en `char`-variabel, vil `c - 'a'` fortelle hvor mange hakk bortenfor “a” tegnet i `c` ligger, og hvis `i` er en `int`-variabel, vil `(char)('a' + i)` gi tegnet som ligger `i` hakk bortenfor “a”.
- *Modulo-operatoren*, `%`, regner ut resten man ville ha sittet igjen med etter en divisjon. F.eks. vil `12 % 7` gi 5, og `-20 % 7` vil gi `-6`. (Merk: dette er forskjellig fra den *matematiske* modulo-operasjonen, som fungerer annerledes for negative tall; den ville ha gitt 1 for sistnevnte operasjon — men det burde ikke bli noe problem her.)

## Eksempel

### Input

```
pcvgxektsspvggn
```

### Output

```
25: qdwhyfluttqwwho
24: rexizgmvuurxxip
23: sfyjahnwvvsyyjq
22: tgzkbioxwvtzzkr
21: uhalcjpgyxxuaals
20: vibmdkqzyyvbmt
19: wjcnelrazzwccnu
18: xkdofmsbaaxddov
17: ylepgntcbbyeepw
16: zmfqhoudcczffqx
15: angripveddaggr
14: bohshjqwfeebhhsz
13: cpitkrxgffciita
12: dqjulsyhggdjjub
11: erkvmtzihhekkvc
10: fslwnuajiifllwd
9: gtmxovbkjgmmxe
8: hunypwclkkhnyf
7: ivozqxdmllioozg
6: jwparyenmmjppah
5: kxqbszfonnkqabi
4: lyrctagpoolrrcj
3: mzsduhbqppmssdk
2: natevcirqqnttel
1: obufwdjsrrouufm
```

## Kommentar

Meldingen var tydeligvis kryptert med  $k = 15$ , og det ser ut til at originalteksten var “angrip ved daggry”.

## Oppgave 2: Krystalltempelet

Den berømte eventyreren Ida Johnsen er i ferd med å utforske det fryktede Krystalltempelet på leting etter den minst like fryktede Dommedagshodeskallen. Hun har endelig kommet frem til døren som leder inn til det aller helligste. På døren er det tegnet inn mange merkelige symboler som ser ut som om de kan trykkes litt inn i døren, og det står også skrevet en tekst på et eldgammelt språk — heldigvis ett av de mange språkene Johnsen behersker. Hun tolker teksten omtrent slik: “For å komme inn til det aller helligste må alle disse steinsymbolene trykkes inn, men på en slik måte at følgende regler overholdes — hvis ikke vil du bli rammet av en grusom forbannelse!” Reglene som står skrevet er av typen “Stein 4 må trykkes inn før stein 2”, “Stein 9 må trykkes inn før stein 7” og så videre. Heldigvis er Johnsen en moderne eventyrer som alltid har med seg en ultraportabel laptop, så her er det bare å sette i gang kodingen.

### Input

Først kommer en linje med to heltall:  $n$  (antallet knapper) og  $k$  (antallet regler).  $n$  og  $k$  vil være mindre enn 1 000 000. Steinene er nummerert fra og med 0 til og med  $n - 1$ . Deretter kommer  $k$  linjer, hver med to tall  $s$  og  $t$ , som da betyr at stein  $s$  må trykkes inn før stein  $t$ . Merk at  $s$  og  $t$  ikke trenger å trykkes inn rett etter hverandre; det holder at  $s$  allerede er blitt trykket på ett eller annet tidspunkt før  $t$  trykkes inn. Ingen regler vil opptre mer enn én gang. Det kan finnes steiner som ikke nevnes i noen regler — disse kan da trykkes inn når som helst.

### Output

Én linje med  $n$  tall på — en trygg rekkefølge å trykke inn knappene på. Hvis det finnes flere trygge rekkefølger, er det likegyldig hvilken av dem som skrives ut. Merk at det ikke er lov å trykke inn noen steiner samtidig. Hvis det ikke finnes noen trygge rekkefølger (f.eks. hvis stein 1 må trykkes inn før stein 2, men stein 2 også må trykkes inn før stein 1), skal én linje med teksten **Kom deg ut!** skrives ut.

### Eksempel

#### Input

```
10 9
1 4
8 4
0 6
```

7 3  
9 7  
2 7  
4 2  
9 0  
4 9

### Output

1 8 4 5 2 9 7 3 0 6

### Kommentar

Det finnes mange gyldige rekkefølger her — eksempelvis ville 5 8 1 4 9 0 6 2 7 3 også ha blitt god tatt.

## Oppgave 3: Ticket to ride

I Hjulstad kommune drives alle bussrutene av samme firma, og de har et noe selsomt billettsystem — en billett gjelder for en viss distanse, f.eks. 700 meter, og det selges bare billetter for visse predefinerte distanser. Når man skal ta bussen, må man kjøpe en kombinasjon av billetter slik at den sammenlagte billett distansen blir minst like lang som den strekningen man har tenkt å kjøre. Avstanden mellom enhver bussholdeplass er delelig på 100 meter, og det finnes 100-metersbilletter, så det vil alltid være mulig å kjøpe billetter som “går opp” i strekningen man skal kjøre — men det er ikke alltid dét lønner seg, for billettene er selvfølgelig forskjellig priset. Innbyggerne i Hjulstad bestemmer seg for å leie inn en dyktig programmerer (deg) til å lage et program som kan hjelpe dem med å bestemme hvordan de skal kjøpe billetter på best mulig måte.

### Input

Første linje inneholder to tall  $n$  og  $k$  — henholdsvis antallet forskjellige billett distanser og antallet billettutregninger det vil bli spurt om.  $n$  vil være mindre enn 1 000, og  $k$  vil være mindre enn 100 000. Deretter kommer  $n$  linjer som hver inneholder to heltall: en billett distanse og prisen for den billetten. Alle distansene vil være delelige på 100, og vil være mindre enn 100 000. Ingen linjer vil ha samme distanse, og én av linjene vil ha distansen 100. Distansene vil stå i stigende rekkefølge. Det kan eksistere billetter som er totalt ubrukelige (på den måten at den samme distansen kan kjøpes billigere ved hjelp av andre billetter). Deretter kommer  $k$  linjer, hver med ett positive heltall som er delelig på 100 — dette er strekninger som du må finne den lavest mulige prisen for. Disse strekningene er ikke nødvendigvis sortert.

## Output

$k$  linjer, hver med ett heltall som forteller hva som er den lavest mulige prisen for billetter for den tilsvarende strekningen fra inputen (husk at det er lov å kjøpe billetter for en lengre strekning hvis det blir billigere enn den eksakte strekningen). Selve billettkombinasjonen som er billigst skal ikke skrives ut.

## Eksempel 1

### Input

```
3 3
100 4
400 11
500 15
800
200
700
```

### Output

```
22
8
22
```

### Kommentar

For å reise 200 meter er det billigere med 2 100-metersbilletter enn med en 400-metersbillett, mens for både 700 og 800 meter er det billigst med to 400-metersbilletter.

## Eksempel 2

### Input

```
5 4
100 5
300 14
400 22
900 45
1100 46
2000
3000
500
4700
```

### Output

```
88
130
24
198
```

## Oppgave 4: Ringenes herre

Petter er veldig glad i kjæresten sin — men i det siste har det dukket opp en mørk sky på horisonten (som han selvfølgelig ikke tør å nevne høyt): hun har fått veldig lyst på en ring. Studenter har som kjent begrenset økonomi, og Petter er ikke noe unntak. De skal snart ut på sin ukentlige spasertur gjennom byen, og Petter er redd for at kjæresten ved turens slutt vil uttrykke et ønske om den dyreste ringen hun har sett i gullsmedvinduene i løpet av turen. Derfor må Petter planlegge en rute gjennom byen som er slik at den dyreste gullsmeden de går forbi er billigst mulig.

### Input

Første linje i input inneholder to tall  $k$  og  $v$ : antall gatekryss og antall veier.  $k$  vil være mindre enn 10 000, og  $v$  vil være mindre enn 100 000. Gatekryssene er nummerert fra og med 0 til og med  $k - 1$ . Deretter kommer en linje med to tall  $s$  og  $t$ : nummeret på krysset hvor turen starter og nummeret på krysset hvor turen slutter. Deretter kommer  $v$  linjer som beskriver gatene. Hver linje inneholder tre tall: numrene på de to gatekryssene gaten går mellom (alle gater kan brukes i begge retninger, men bare den ene retningen er oppgitt), og prisen på den dyreste ringen i gullsmedforretningen i den gaten. Ikke alle gater har noen gullsmedforretning; da vil den oppgitte prisen være 0. Alle prisene vil være lavere enn 100 000. Veikartet vil være sammenhengende; det vil altså være mulig å nå frem til et hvilket som helst gatekryss. Det vil aldri være mer enn én direkte vei mellom de samme to kryssene.

### Output

Én linje som inneholder ett tall: prisen Petter er redd for å måtte betale — den dyreste prisen langs en rute som er slik at ingen andre mulige ruter har lavere makspris.

### Eksempel

#### Input

```
9 13
0 7
0 1 1000
1 2 0
1 4 1500
3 4 1700
3 0 2000
3 6 1900
6 7 2000
6 8 500
7 8 1500
4 8 2300
8 5 2000
4 5 2000
2 5 1900
```

## Output

1900

## Kommentar

Her illustreres bykartet som er beskrevet i inputen, og den beste ruten er tegnet inn:

