zadatak: Put

Perica treba da pređe autom put od grada A do grada B. Njegov auto ima rezervoar zapremine *p* litara i troši jedan litar goriva na 1 kilometar. Na putu od grada A do grada B se nalaze benzinske pumpe, raspoređene tako da Perica može stići na cilj (znači svake dve uzastopne pumpe su na rastojanju ne većem od *p*). Cene benzina na pumpama ne moraju biti iste, a predstavljaju cele brojeve između 1 i 1000. Perica želi da pređe put uz minimalni trošak kupujući benzin tamo gde je najjeftiniji, ali tako da može stići na cilj (tj. tako da ne ostane bez benzina negde između neke dve pumpe). Kako je put vrlo dugačak i broj usputnih pumpi velik, on to ne može dovoljno brzo proračunati. Pomozite Perici i napišite program koji će određivati koliko minimalno mora platiti da bi prešao put od A do B.

Ulaz:

Ulazni podaci se nalaze u datoteci **put.in**. U prvom redu datoteke su dva broja: ceo broj *p* ($1 \leq p \leq 100000000 = 10^8$) koji prestavlja zapreminu rezervoara i ceo broj *s* ($1 \leq s \leq 100000 = 10^5$) koji predstavlja broj pumpi na putu. U svakom od sledećih s redova se nalazi opis jedne pumpe i to u redosledu u kome se one nalaze na putu. Opis je zadat sa dva cela broja: broj *d* ($1 \leq d \leq p$) i to je rastojanje od te pumpe do sledeće (ako je to poslednja pumpa onda je *d* rastojanje od te pumpe do mesta B) i ceo broj *c* ($1 \leq c \leq 1000$), cena 1 litra benzina na toj pumpi. Ukupna dužina puta nije veća od $100000000 = 10^8$ kilometara. Prva pumpa se nalazi u mestu A.

Izlaz:

U prvom redu datoteke **put.out** treba ispisati jedan ceo broj koji predstavlja najmanji iznos novca koji treba da plati Perica da bi stigao iz mesta A u mesto B.

Primer:

put.in	**put.out**
40 3 10 2 15 1 5 2	

zadatak: Kurs

Perica je sasvim slučajno primetio da je jedna banka napravila sledeći previd: za jedan uloženi euro je davala 1.2 dolara, za jedan uloženi dolar je plaćala 0.6 funti, a za jednu uloženu funtu građanin je dobijao 1.45 eura. Perica je proračunao da će, ako promeni 1 euro u dolare, zatim dobijene dolare pretvori u funte, a onda sve dobijene funte u eure dobiti 1.044 eura. To je Perica i iskoristio, uložio 1000 eura i dobio 1044, tj. zaradio dragocenih 44 eura. Međutim, verovatno su i bankari primetili da im nešto ne štima pa su već narednog dana popravili kursnu listu i ova jednostavna konverzija euro-dolar-funta-euro nije donosila dobit. Međutim, kursna lista sadrži i druge valute te nije isključeno da negde ne postoji dobitni lanac. Pomozite Perici da ga nađe.

Ulaz:

Ulazni podaci se nalaze u datoteci **kurs.in**. U prvom redu je broj kursnih listi i to je ceo broj *nl* koji nije veći od 5. U sledećim redovima se nalaze opisi tih listi. U prvom redu svakog opisa je broj različitih valuta koji obrazuju tu listu i to je ceo pozitivan broj *nv* koji nije veći od 100. Valute su numerisane brojevima od 1 do *nv*. U narednih nv redova je opis jedne valute. Opis jedne valute (recimo da je to valuta broj *i*) se sastoji od *nv* mešovitih brojeva, tako da *j*-ti broj označava koliko se dobija *j*-te valute ako se uloži jedna jedinica *i*-te valute.

Izlaz:

U izlaznoj datoteci **kurs.out** treba ispisati *nl* redova, a u svakom redu po jedan ceo broj (0 ili 1). U *i*-tom redu treba upisati ceo broj 1 ako za *i*-tu kursnu listu postoji dobitni lanac zamena takav da se može ostvariti zarada, a broj 0, ako u *i*-toj kursnoj listi nema dobitnog lanca. Vodite računa da ne treba štampati dobitni niz konverzija.

Primer:

kurs.in	**kurs.out**
2 3 1.000 1.200 0.780 0.800 1.000 0.650 1.450 1.520 1.000 3	

```
1.000 1.200 0.680
0.800 1.000 0.650
1.250 1.520 1.000
```

zadatak: Agenti

Većina ljudi misli da se špijuni nalaze samo u filmovima o Džejmsu Bondu. Međutim, da to nije tačno mogli su se uveriti i domaći bezbedonosni agenti koji su u jednom skladištu pronašli mašinu za kriptovanje, koju koriste strani špijuni. Oni su i ranije persretali poruke, ali nisu mogli ni da shvate kako su kriptovane. Sada su, međutim, pronašli sistem i zamolili su vas da napišete program koji će automatski dekriptovati presretnute poruke.
Princip rada mašine je sledeći: ulazni podataka je poruka koja se sastoji samo od malih slova latinice S1 = s1s2...s*N* . Mašina zatim generiše sve ciklične permutacije S2 = s2s3...s*N*s1, S2 = s3s4...s*N*s1 s2, ..., S*N* = s*N*s1s2...s*N*-1 i sortira ih u leksikografskom poretku. Na ovaj način se dobija matrica dimenzija *N*x*N* (u *i*-toj vrsti matrice je *i*-ta po redu permutacija u sortiranom nizu permuacija, a u okviru jedne vrste elementi su pojedinačna slova u okviru permutacije koja se nalazi u toj vrsti). Kodirana poruka se sastoji od rednog broja vrste (označimo ga sa *K*) u kojoj se nalazi originalna poruka i spiska slova koja se nalaze u poslednjoj koloni matrice. Evo primera. Za poruku S1 = abracadabra imamo sledeći niz permutacija (nakon sortiranja):

```
 1. aabracadabr = S11
 2. abraabracad = S8
 3. abracadabra = S1
 4. acadabraabr = S4
 5. adabraabrac = S6
 6. braabracada = S9
 7. bracadabraa = S2
 8. cadabraabra = S5
 9. dabraabraca = S7
10. raabracadab = S10
11. racadabraab = S3
```

Prema tome, šifrovana poruka je: 3 rdarcaaaabb.
Ulaz:
U prvom redu tekstualne datoteke **agenti.in** nalazi se prirodan broj *K* (redni broj vrste matrice u kojoj se nalazi originalna poruka). U drugom redu nalazi se *N* (1 < *N* ≤ 100000) slova koji predstavljaju zadnju kolonu matrice. Između slova nema razmaka.
Izlaz:
U prvom redu tekstualne datoteke **agenti.out** treba ispisati originalnu poruku.
Primer:

agenti.in	**agenti.out**
3 rdarcaaaabb	

fajl: agenti.pas

```
program agenti;
var
  a:array[1..100000] of char;
  next:array[1..100000] of longint;
  i,k,n:longint;
  ch:char;
procedure sort(l,r:longint);
var
  i,j:longint;
  k,t:longint;
begin
  i:=l;
  j:=r;
  k:=next[(i+j) div 2];
  while i<=j do
    begin
      while (a[next[i]]<a[k]) or ((a[next[i]]=a[k]) and (next[i]<k)) do inc(i);
      while (a[k]<a[next[j]]) or ((a[next[j]]=a[k]) and (k<next[j])) do dec(j);
      if i<=j then
```

```
        begin
          t:=next[i];
          next[i]:=next[j];
          next[j]:=t;
          inc(i);
          dec(j)
        end
    end;
  if i<r then sort(i,r);
  if l<j then sort(l,j)
end;
begin
  assign(input,'agenti.in');
  reset(input);
  readln(k);
  n:=0;
  while not eof do
    begin
      read(ch);
      inc(n);
      a[n]:=ch;
      next[n]:=n
    end;
  close(input);
  sort(1,n);
  assign(output,'agenti.out');
  rewrite(output);
  for i:=1 to n do
    begin
      k:=next[k];
      write(a[k])
    end;
  close(output)
end.
```