

## ”Uzlīmes” (Latvija)

Kārlis ir automobiļu sacīkšu cienītājs, kurš nolēmis veidot savu modeļu kolekciju. Veikalā iespējams nopirkt modeļus aizlīmētās kastēs. Katrā kastē atrodas viena modeļa detaļas un noteikts skaits uzlīmju ar ciparu attēliem. Katrā kastē esošo uzlīmju kopa ir viena un tā pati. Saliktos modeļus Kārlis ir nolēmis numurēt pēc kārtas ar skaitļiem, sākot no 1, aplīmējot tos ar attiecīgo ciparu uzlīmēm. Piemēram, lai aplīmētu 2070.modelīti, būs nepieciešamas četras uzlīmes: divnieka, divas nulļu un septiņnieka uzlīme.

Neviena skaitļa pieraksts nesākas ar 0.

Kārlis katrreiz rīkojas sekojoši: atver jaunu modeļa kasti, saliek tajā esošo modeli un uzlīmē nepieciešamās (vienu vai vairākas) uzlīmes. Kārlis var izmantot vēl neizmantotās uzlīmes no kārtējās un jau iepriekš atvērtajām modeļu kastēm, bet nedrīkst atvērt jaunu kasti tikai tāpēc, lai paņemtu trūkstošās uzlīmes.

Uzrakstiet programmu, kas ievadītam katrā kastē esošo katra cipara uzlīmju skaitam nosaka, cik modeļus šādā veidā Kārlim izdosies aplīmēt!

### *Ievaddati*

Teksta faila STI.IN pirmajā rindā doti desmit viencipara naturāli skaitļi  $i_0 i_1 i_2 i_3 i_4 i_5 i_6 i_7 i_8 i_9$ , kur  $i_j$  norāda, cik uzlīmes ar cipara  $j$  ( $0 \leq j \leq 9$ ) attēlu atrodas katrā kastē. Katri divi blakus skaitļi ir atdalīti ar tukšumsimbolu.

### *Izvaddati*

Teksta faila STI.OUT pirmajā rindā jāizvada viens naturāls skaitlis –iepriekšaprakstītajā veidā aplīmēto modeļu skaits.

### *Piemēri*

Ievaddati (fails STI.IN)

1 1 1 1 1 1 1 1 1 1

Izvaddati (fails STI.OUT)

199990

Ievaddati (fails STI.IN)

3 4 5 4 3 4 5 4 3 4

Izvaddati (fails STI.OUT)

49999999499999999949999999973

## "Mutekši" (Igaunija)

Modernās programmēšanas valodas ļauj rakstīt programmas, kas sastāv no vairākiem izpildes *pavedieniem*. Tas līdzinās vairāku programmu paralēlai izpildei vienā un tajā pašā adresu telpā, izmantojot vienus un tos pašus mainīgos. Bieži rodas nepieciešamība šos pavedienus savā starpā sinhronizēt. Piemēram, vienam pavedienam varētu rasties vajadzība pagaidīt, kamēr kāds cits pavediens veic noteiktas darbības un ieraksta rezultātu kādā noteiktā mainīgajā.

Visvienkāršākais pavedienu sinhronizācijas līdzeklis ir *muteksis*. Muteksis ir speciāls objekts, kas var būt *aizņemts* vai *brīvs*. Aizņemts muteksis vienmēr pieder tieši vienam pavedienam. Ir divas operācijas, ko pavediens var pielietot muteksim: LOCK un UNLOCK.

Kad pavediens pielieto LOCK operāciju brīvam muteksim, šis muteksis tiek aizņemts un kļūst piederīgs attiecīgajam pavedienam. Ja pavediens  $x$  mēģina pielietot LOCK operāciju muteksim, kas jau pieder kādam citam pavedienam  $y$ , pavediens  $x$  tiek bloķēts, līdz kamēr attiecīgais muteksis tiek atbrīvots.

Kad pavediens pielieto UNLOCK operāciju muteksim, kas tam pieder, muteksis kļūst brīvs. Ja citi pavedieni gaida, lai pielietotu šim muteksim LOCK operāciju, tad vienam no tiem tas izdosies un muteksis atkal tiks aizņemts. Ja uz mutekša atbrīvošanu gaidīja vairāki pavedieni, tad viens no tiem tiek izvēlēts patvaļīgi, bet pārējie turpina gaidīt.

Vispārēja daudzpavedienu programmu problēma ir *strupceļš*. Strupceļš ir tad, kad divi vai vairāk pavedieni gaida viens uz otru, lai tiktu atbrīvots muteksis, un neviens pavediens nevar turpināt izpildi. Strupceļš iestājas arī tad, ja pavediens gaida uz tāda mutekša atbrīvošanu, kas pieder pavedienam, kurš izpildi jau beidzis bez mutekša atbrīvošanas.

**Uzdevums:** Dotajiem pavedienu aprakstiem nosakiet, vai to izpildē iespējams strupceļš. Šajā gadījumā jāatrod pilnīgs strupceļš, kad neviens pavediens nevar turpināt izpildi - pavedienam jābūt savu darbu beigušam vai bloķētam ar kādu muteksi. Protams, ka strupceļš nav tad, ja visi pavedieni ir darbu pabeiguši.

Katrs no pavedieniem sastāv no instrukciju virknes, kur katra instrukcija ir vienā no formām:

```
LOCK <muteksis>  
UNLOCK <muteksis>
```

Jūs varat uzskatīt, ka

- katra mutekša vārds ir latīņu alfabēta lielais burts (A..Z);
- neviens paveliens nemēģina pielietot LOCK muteksim, kas tam jau pieder;
- neviens paveliens nemēģina pielietot UNLOCK muteksim, kas tam nepieder.

**Ievaddati:** Teksta faila MUT.IN pirmā rinda satur pavelienu skaitu  $M$  ( $1 \leq M \leq 5$ ), nākošajās faila rindās seko  $M$  bloki, kas katrs apraksta vienu pavelienu. Bloka, kas apraksta  $i$ -to pavelienu, pirmajā rindā ir šajā pavelienā esošo instrukciju skaits  $N_i$  ( $1 \leq N_i \leq 10$ ), kurai seko  $N_i$  instrukciju rindas. Instrukcijas nesatur liekus tukšumsimbolus.

**Izvaddati:** Izvaddatu faila MUT.OUT pirmajai rindai jā satur viens skaitlis  $D$ .  $D$  jābūt 1, ja strupceļš ir iespējams, vai 0, ja nav.

Ja strupceļš ir iespējams, otrajai rindai jāapraksta programmas stāvoklis, kurā iestājas pilnīgs strupceļš. Ja strupceļš iespējams vairākos gadījumos, jāapraksta jebkurš no tiem.

Programmas stāvokli apraksta norādot 0-bāzētu instrukcijas indeksu (pirmajai instrukcijai atbilst indekss 0, otrajai - indekss 1, ...) katrā pavelienā tādā secībā, kā pavelieni doti ievaddatu failā. Pavelienam, kas savu darbību beidzis, kā atbilstošo indeksu izvadiet -1. Visi indeksi jāizvada vienā rindā un starp katriem blakus indeksiem jāizvada viens tukšumsimbols.

Piemērs:

MUT.IN	MUT.OUT
2	1
1	-1 1
LOCK X	
2	
LOCK Y	
LOCK X	

## "Dalīšanas izteiksme" (Polija)

Dalīšanas izteiksme ir aritmētiska izteiksme formā:

$$x_1 / x_2 / x_3 / \dots / x_k$$

, kur  $x_i$  ir vesels pozitīvs skaitlis visiem  $i$ , ( $1 \leq i \leq k$ ). Dalīšanas izteiksme tiek izpildīta no kreisās uz labo pusi. Piemēram, izteiksmes

$$1/2/1/2$$

vērtība ir  $1/4$ . Izteiksmē drīkst salikt iekavas, lai izmainītu tās vērtību. Piemēram, izteiksmes

$$(1/2)/(1/2)$$

vērtība ir 1.

Ir dota dalīšanas izteiksme E. Vai iespējams šajā izteiksmē salikt iekavas tā, lai iegūtu izteiksmi E', kuras vērtība ir vesels skaitlis?

**Uzdevums:** Uzrakstiet programmu, kas katrai datu kopai no vairākām dotajām datu kopām veic sekojošo:

- Nolasa izteiksmi E no teksta faila DIV.IN
- Noskaidro, vai no dotajā izteiksmē E iespējams salikt iekavas tā, lai iegūtu izteiksmi E', kuras vērtība ir vesels skaitlis,
- Rezultātu ieraksta teksta failā DIV.OUT

**Ievaddati:** Teksta faila DIV.IN pirmajā rindā dota naturāla skaitļa  $d$  ( $d \leq 5$ ) vērtība – datu kopu skaits. Tālāk seko datu kopu saturs. Katras datu kopas pirmajā rindā ir dota naturāla skaitļa  $n$  ( $2 \leq n \leq 10000$ ) vērtība – izteiksmē esošo skaitļu skaits. Katrā no nākošajām  $n$  rindām dota viena naturāla skaitļa vērtība, kas nepārsniedz 1000000000. Tad pēc kārtas  $i$ -tais skaitlis ir  $i$ -tais skaitlis izteiksmē.

**Izvaddati:** Katram  $i$  ( $1 \leq i \leq d$ ) izvaddatu faila DIV.OUT  $i$ -tajā rindā programmai jāizvada vārds YES, ja  $i$ -to izteiksmi iespējams pārveidot tā, ka tās vērtība ir vesels skaitlis, vai vārds NO pretējā gadījumā.

### Piemērs:

Ievaddatu failam DIV.IN:

2  
4  
1  
2  
1  
2  
3  
1  
2  
3

pareizais rezultāts ir izvaddatu fails DIV.OUT:

YES  
NO