Problema 1 - vraja 100 puncte

Vrăjitoarea cea rea a ferecat-o pe prințesă într-un castel fermecat spunându-i că va putea ieşi doar dacă trece prin **toate** camerele, **o singură dată**, şi îi mai rămâne suficientă putere să iasă din castel, considerând că pentru ieşire îşi consumă din putere atât câte camere este nevoită să sară până în afara castelului. Vrăjitoarea rămâne la intrarea în castel ca prințesa să nu poată ieşi pe acolo. Ea poate ajunge dintr-o cameră în alta printr-o vrajă pe care o găseşte în camera respectivă, vrajă care îi arată în ce cameră următoare ajunge. La fiecare salt într-o cameră cu număr mai mare câştigă putere şi la fiecare salt în cameră cu număr mai mic pierde putere, echivalent cu numărul de camere ”sărite”. Ajutați-o pe prințesă să afle dacă poate sau nu ieşi din castel.

Cerință
Scrieți un program care să afişeze valoarea puterii rămase (***x* după ieşirea** din castel) sau **-1** dacă nu iese, conform situației descrise mai sus.

Date de intrare
Fişierul de intrare **`vraja.in`** conține două linii. Pe prima linie sunt scrise numărul natural nenul **n** reprezentând numărul de camere din castel şi numărul natural nenul **x,** reprezentând puterea inițială a prințesei. Pe a doua linie sunt scrise cele **n** numere naturale c_0, c_1, c_2, c_3,…,c_{n-1}, separate prin câte un spațiu, reprezentând numărul de ordine al camerei în care poate ajunge prințesa.

Date de ieşire
Fişierul de ieşire **`vraja.out`** va conține o singură linie pe care se va scrie ***x*** dacă prințesa poate ieşi din castel, respectiv ***-1*** dacă ea rămâne blocată în castel deoarece nu mai are putere să iasă sau deoarece ajunge într-o cameră prin care a trecut deja.

Restricții şi precizări
- **n, x, c_i** sunt numere naturale nenule
- $2 \le n \le 3000$
- $1 \le c_0, c_1, \ldots, c_n \le 3000$
- Prințesa porneşte din camera **0 şi** iese doar prin dreapta
- O cameră poate avea numărul ei însăşi: dacă nu sunt toate camerele parcurse prințesa se blochează acolo, dacă sunt parcurse toate ea poate încerca să iasă

Exemple

`vraja.in`	**`vraja.out`**	***Explicații***
`3 1` `2 1 1`	`1`	Sunt 3 camere, puterea inițială e 1. Din camera 0 ajunge în 2, puterea creşte cu 2, din camera 2 ajunge în camera 1, puterea scade cu 1, din camera 1 nu e obligată să sară în altă cameră, a parcurs toate camerele, ca urmare iese din castel având putere 2, suficientă cât să sară peste camera 2, rămânând cu putere 1.
`3 1` `2 0 1`	`-1`	Sunt 3 camere, puterea inițială e 1. Din camera 0 ajunge în 2, puterea creşte cu 2, din camera 2 ajunge în camera 1, puterea scade cu 1, din camera 1 se întoarce în camera 0, unde a fost deja.
`5 1` `4 1 1 2 3`	`-1`	Din camera 0 ajunge în 4, puterea creşte cu 4, din camera 4 în camera 3, puterea scade cu 1, din camera 3 în 2, puterea scade cu 1, din 2 în 1, puterea scade cu 1, puterea finală e 2 şi nu poate ieşi din castel având putere 2, insuficientă cât să sară peste camerele 2,3,4 şi să iasă din castel

Timp maxim de executare/test: 1 secundă
Limita de memorie: 2 Mb

Problema 2 - *Markov* **100**

Strategia generală într-un algoritm Markov este de a transforma şirul de caractere de intrare **x** în şirul de caractere de ieşire **y** prin aplicarea repetitivă a unui număr de paşi**.**

Paşii sunt de forma **u → v**, unde **u** şi **v** sunt şiruri de caractere. Un astfel de pas este aplicabil şirului de intrare **x** daca există cel puțin o apariție a şirului **u** în **x**, altfel pasul nu este aplicabil. Dacă pasul este aplicabil, se înlocuieşte prima apariție a lui **u** cu **v** în şirul **x şi** se reiau paşii începând cu primul.

Un algoritm Markov se termină atunci când nu mai există paşi aplicabili.

Date de intrare
Fişierul de intrare **markov.in** conține:
- pe prima linie şirul de intrare **x;**
- pe a doua linie numărul de paşi **n**;
- pe următoarele **n** linii perechi de şiruri (**u, v)** separate prin câte un spațiu.

Date de ieşire
Fişierul de ieşire **markov.out** va conține pe prima linie şirul **x**, transformat conform algoritmului descris..

Restricții şi precizări
- $1 \leq \mathbf{n} \leq 100$;
- **x, u, v** sunt şiruri de maxim 1000 de caractere din alfabetul englez;
- Datele de intrare sunt corecte :
  1. **nu există** pas **u → v** în care **u** să fie inclus în **v** ;
  2. lungimea şirului **v** este mai mică sau egală cu lungimea şirului **u**;
  3. **nu există** perechi de paşi circulari, de forma **ab → ef** şi **ef → ab.**

Exemplu

markov.in	**markov.out**	**Explicație**
abcdefg 5 ab ef ef ad ad d cd ef dg d	ddd	Transformările succesive sunt: abcdefg → efcdefg → adcdefg →adcdadg → dcdadg → dcddg →defdg →daddg → dddg → ddd

- Timpul limită de execuție: 1 secundă/test (1000ms)
- Memoria totală: 2MB
- Dimensiunea maximă a stivei: 1 MB
- Dimensiunea maximă a sursei: 5 KB

Descrierea solutiei
Problema "vraja"

Se citesc valorile date, conform enunțului, numărul de camere, puterea şi valorile din vectorul c.

Se parcurge vectorul, cu while (condiție pe pozițțe şi switch) pentru a putea sări dintr-o cameră în alta:

- pentru a verifica salt înainte sau salt înapoi şi a calcula puterea,
- pentru a verifica dacă mai poate intra în camera respectivă sau nu,
- se pun camerele parcurse pe o valoare neutră (d.p.d.v al datelor de intrare)

si se foloseşte un switch pentru vector parcurs (verifică valoarea neutră).

Se testează switchul şi puterea rămasă (x>=n-i, unde i este pozitia curentă a prințesei) şi dacă e ok se calculează puterea rămasă, dacă nu, se afisează -1.

Problema - *Markov* **Descrierea soluției** **100**

Se reţin cei n pasi.
Cât timp mai există paşi aplicabili
    Se caută primul pas **u** aplicabil
    Dacă există
        se inlocuieşte cu perechea lui **v**
        se reia procedura de înlocuire **de la primul pas**