Sortări — 100 puncte

Fişier sursă: sortari.pas, sortari.c sau sortari.cpp

Balaurului Arhirel nu îi plac prea mult şirurile care nu sunt ordonate. Din acest motiv, nu poate să suporte permutările de `N` elemente, aşa că se decide să le sorteze şi pentru asta inventează o metodă proprie de sortare.

El ia iniţial un şir `S` care reprezintă o permutare de ordin `N`. Caută în şirul `S` cel mai mic (`min`) şi cel mai mare element (`max`). Să considerăm că `min` se află în şirul `S` pe poziţia `pmin`, iar `max` pe poziţia `pmax`. Să notăm cu `x` minimul dintre `pmin` şi `pmax`, iar cu `y` maximul dintre `pmin` şi `pmax`. Şirul `S` a fost astfel partiţionat în alte trei şiruri, `S1`, `S2`, `S3` care pot avea fiecare zero elemente, un element sau mai multe elemente. Şirul `S1` începe la prima poziţie din şir şi se termină la poziţia `x-1`. Şirul `S2` începe la poziţia `x+1` şi se termină la poziţia `y-1`. Şirul `S3` începe la poziţia `y+1` şi se termină la ultima poziţie din şir.

Balaurul Arhirel mută valoarea min la capătul din stânga al lui `S`, iar valoarea max la capătul din dreapta al şirului `S` şi reia sortarea pentru fiecare din şirurile `S1`, `S2`, `S3`.

De exemplu să considerăm `N=6` şi şirul `S=(3 4 2 1 6 5)`. La primul pas, `min=1` şi `max=6`. Deci `S1=(3 4 2)`; `S2=()`; `S3=(5)`. Se mută `min` şi `max` la capetele şirului şi se obţine `S=(1 3 4 2 5 6)` şi se sortează în acelaşi mod `S1`, `S2` şi `S3`. `S2` şi `S3` au `0`, respectiv `1` element, deci sunt deja sortate. Pentru `S1`, se găseşte `min=2` şi `max=4` şi vom avea şirurile `(3)`; `()`; `()`. Se mută `min` şi `max` la capete şi se obţine `S1=(2 3 4)`. În final, vom avea şirul `S=(1 2 3 4 5 6)`.

Evident, această metodă nu va funcţiona întotdeauna pentru sortarea unei permutări.

Spre exemplu, pentru şirul `S=(3 4 1 6 5 2)`, se găseşte `min=1` şi `max=6`, iar `S1=(3 4)`, `S2=()`, `S3=(5 2)`. Se mută `min` şi `max` la capetele lui `S`: `S=(1 3 4 5 2 6)` şi se procedează la sortarea pe rând a şirurilor `S1`, `S2`, `S3`. `S1` este sortat, `S2` nu are elemente, iar `S3` va deveni `S3=(2 5)`. În final, `S=(1 3 4 2 5 6)`.

Cerinţă

Ajutaţi-l pe Balaurul Arhirel să afle câte dintre permutările de `N` elemente pot fi sortate prin metoda sa.

Date de intrare

Fişierul `sortari.in` conţine o singură linie pe care se află numărul `N`.

Date de ieşire

Fişierul `sortari.out` va conţine o singură linie pe care se află numărul de permutări de ordin `N` ce pot fi sortate prin metoda balaurului modulo `19573` (restul împărţirii numărului de permutări la `19573`).

Restricţii

`0<N<201`

Exemplu

sortari.in	sortari.out	*Explicaţie*
4	18	În cazul permutărilor de câte 4 elemente, şirurile `(1 3 4 2)`; `(3 1 2 4)`; `(3 1 4 2)`; `(3 4 1 2)`; `(3 4 2 1)`; `(4 3 1 2)` nu vor putea fi sortate crescător. Celelalte `24-6=18` permutări pot fi sortate crescător. De exemplu, pentru şirul `(1 3 4 2)`, după găsirea `min=1`, `max=4`, se obţin şirurile: `S1=()`; `S2=(3)`; `S3=(2)`, care, având câte un element sunt sortate. Astfel, după mutarea la capete a lui `min` şi `max` vom avea: `(1 3 2 4)`

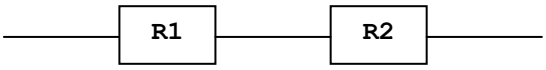
Timp maxim de execuţie/test: 0.2 secunde pentru Linux şi 1.5 secunde pentru Windows.

Rez — 100 puncte

Fișier sursă: rez.pas, rez.c sau rez.cpp

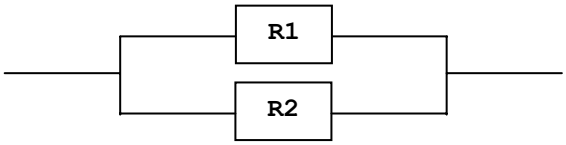
Gigel este electronist amator. El afirmă că a inventat o nouă componentă electronică denumită reziston. În mod ciudat totuşi rezistonii au nişte proprietăți care nouă ne sună foarte cunoscut:

1. Orice reziston este caracterizat printr-o mărime fizică numită rezistonță. Aceasta poate avea ca valori numai numere naturale.
2. Rezistonii pot fi legați între ei în serie sau în paralel, formând astfel circuite.
3. Fie doi rezistoni având rezistonțele `R1`, respectiv `R2`. Legarea în **serie** a rezistonilor se realizează astfel:



Rezistonța acestui circuit va fi `R1+R2`.

4. Legarea celor doi rezistoni în **paralel** se realizează astfel:



Rezistonța acestui circuit va fi `(R1*R2)/(R1+R2)`. Fiindcă rezistonțele pot fi numai numere naturale, împărțirea este întreagă (adică rezultatul este câtul împărțirii întregi a lui `(R1*R2)` la `(R1+R2)`).

5. Prin legarea oricâtor rezistoni în serie şi în paralel se obțin circuite. Circuitele pot fi legate în serie şi/sau în paralel după aceleaşi reguli. Rezistonța unui circuit se calculează aplicând regulile de mai sus.

Un circuit va fi codificat printr-un şir de caractere construit după următoarele reguli:

a. Dacă circuitul `C` este format dintr-un singur reziston şi acesta are rezistonța de valoare `x`, atunci codificarea circuitului `C` este `Rx`. Rezistonța circuitului `C` va fi `x`.
b. Dacă circuitul `C` este obținut prin legarea în serie a două sau mai multe circuite, codificate C_1, C_2, ..., C_k, atunci codificarea circuitului C se obține concatenând în ordine codificările circuitelor $C_1C_2...C_k$. Rezistonța circuitului `C` se obține prin însumarea rezistonțelor circuitelor C_1, C_2, ..., C_k.
c. Dacă circuitul `C` este obținut prin legarea în paralel a două sau mai multe circuite, atunci codificarea circuitului `C` se obține încadrând între paranteze rotunde codificările circuitelor din care este format şi separând aceste codificări prin virgulă: $(C_1, C_2, ..., C_k)$, $k>1$. Rezistonța circuitului `C` este egală cu câtul împărțirii produsului dintre rezistonțele C_1, C_2, ..., C_k şi suma rezistonțelor circuitelor C_1, C_2, ..., C_k.

Cerință

Scrieți un program care să determine rezistonța unui circuit.

Date de intrare

Fişierul de intrare `rez.in` conține pe prima linie un şir de caractere care reprezintă codificarea unui circuit conform regulilor de mai sus.

Date de ieşire

Fişierul de ieşire `rez.out` conține o singură linie pe care este scrisă rezistonța circuitului specificat în fişierul de intrare.

Restricții şi precizări

`0 <` lungimea codificării unui circuit `≤1000`
`0 <` rezistonța oricărui reziston `< 100`
`0 <` rezistonța oricărui circuit `< 2000000000` (două miliarde)
Şirul prin care se codifică un circuit nu conține spații.
Pentru datele de test nu se vor obține împărțiri la `0`.

Exemple

rez.in	rez.out	rez.in	rez.out	rez.in	rez.out	rez.in	rez.out
R12	12	R42R33R3	78	R2(R5,R69,R12)R80	130	(R5,R3(R12,R4),R3)	6

Timp maxim de execuție/test: 0.1 secunde pentru Linux şi 0.1 secunde pentru Windows.

Olimpiada Națională de Informatică
Clasa a X-a
Ziua 1

Gaina — 100 puncte

Fişier sursă: gaina.c, gaina.pas sau gaina.cpp

Găina Chucky 767 trebuie să străbată curtea sărind de pe un coteț pe altul, sau zburând peste cotețe, de la primul la ultimul coteț. Cotețele sunt reprezentate prin nişte dreptunghiuri de lățime 1m şi înălțimi date şi sunt numerotate în ordine de la stânga cu numerele de `1` la `n`. Două cotețe cu numere consecutive sunt lipite (adiacente). Inițial, Chucky are un număr de unități de energie. Dacă la un moment dat Chucky ajunge să aibă **energie strict negativă** sau se află în **imposibilitatea de a mai face un pas** (întâlneşte un coteț mai înalt decât cota la care se află), **atunci nu mai poate continua** acel drum.

Chucky poate să se deplaseze făcând următoarele tipuri de mişcări:

a) **Pas**. Găina păşeşte pe orizontală de pe un coteț de înălțime `H` pe următorul coteț de aceeaşi înălțime (în desen: de la poziția `h` la `i`). În acest caz **nu se pierde şi nu se câştigă energie.**

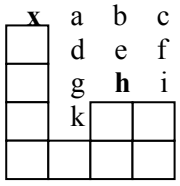
b) **Aterizare**. Găina aterizează pe un coteț de înălțime `H`, venind în zbor, tot de la înălțimea `H` (exemplu în desen: de la `g` la `h`). Nici în acest caz **nu se pierde şi nu se câştigă energie.**

c) **Decolare**. Găina zboară 1m pe orizontală de pe un coteț (exemplu în desen de la `x` la `a`). În acest caz găina **pierde o unitate de energie**.

d) **Zbor orizontal**. Găina zboară pe orizontală (exemplu în desen, de la `a` la `b`, de la `b` la `c`,…). În acest caz **pierde câte o unitate de energie** pentru fiecare metru parcurs pe orizontală.

e. **Picaj**. Găina coboară pe verticală (exemplu în desen de la `a` la `d`, sau de la `d` la `g`…). În acest caz **câştigă câte o unitate** de energie pentru fiecare metru coborât.

Mai jos, avem un drum format din 4 cotețe, de înălțimi `4`, `1`, `2`, `2`. Exemplificăm diferite tipuri de mişcări din poziția `x` (ceea ce nu reprezintă un traseu complet):



EXEMPLU: din poziția de start `x` se poate ajunge în poziția `h` pe 3 trasee:

TRASEU	**Bilanț energie pas cu pas**	**Necesar energie inițială minimă**
1) x, a, b, e, h	x→a(-1), a→b(-1), b→e(+1) ,e→h(+1)	2
2) x, a, d, e, h	x→a(-1), a→d(+1), d→e(-1), e→h(+1)	1
3) x, a, d, g, h	x→a(-1), a→d(+1), d→g(+1), g→h(0)	1

Cerință

Să se determine **energia minimă** (un număr natural) pe care trebuie să o aibă Chucky la începutul călătoriei (când se află pe primul coteț), astfel încât ea să poată ajunge pe cotețul `n`, **având în fiecare moment energia mai mare sau egală cu 0.**

Date de intrare

Fişierul de intrare `gaina.in` conține două linii. Pe prima linie se află numărul natural `n`. Pe linia a doua se află `n` numere naturale $h_1\ h_2\ ...\ h_n$ (unde h_i reprezintă înălțimea cotețului `i`), separate de câte un spațiu.

Date de ieşire

Fişierul de ieşire `gaina.out` conține o singură linie pe care se află un număr natural `K` reprezentând **energia minimă inițială** cu care găina Chucky poate să ajungă la cotețul `n`**,** având în fiecare moment energia mai mare sau egală cu `0`.

Restricții

$0 < n < 10001$
$0 \le h_i \le 30000$

Pentru datele de test există întotdeauna soluție (primul coteț are înălțimea mai mare sau egală cu înălțimea oricărui alt coteț).

Exemple

gaina.in	gaina.out	gaina.in	gaina.out
3 2 2 0	1	6 3 0 0 2 1 1	2

Timp maxim de execuție/test : 0.1 secunde pentru Linux şi 0.1 secunde pentru Windows.