Vhodna datoteka: `register.in` **100 točk**
Izhodna datoteka: `register.out` **Časovna omejitev:** 1.5 s
Izvorna koda: `register.pas/.c/.cpp` **Pomnilniška omejitev:** 16 MB

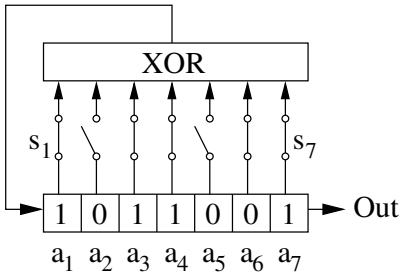
Shift Register

Mikroprocesorski register je prostor, kjer se shranjuje N bitov informacije. Poznamo tudi t.i. pomikalni oz. shift register, ki omogoča pomikanje vseh bitov za eno mesto v levo ali desno, pri čemer se na eni strani v register zapiše en nov bit, na drugi strani pa en bit pade ven iz registra.

Z uporabo pomičnega registra in nekaj dodatnih funkcij lahko naredimo generator psevdo-naključnih števil na sledeč način: N-biten pomikalni register na začetku napolnimo z določenimi bitnimi vrednostmi $a_1, a_2, \ldots, a_N$. V vsakem koraku register na izhod izpiše najbolj desni bit a_N, vse ostale pa pomakne za eno polje v desno. Pri tem na levi strani nastane prosto mesto a_1, v katerega se vpiše nova vrednost, ki se izračuna na sledeč način:

Vsak bit pomikalnega registra je povezan z vezjem XOR prek majhnega stikala, ki je lahko vključeno ali izključeno (glej sliko). Vključeno stikalo (označimo ga s številko 1) pomeni, da se stanje posameznega bita pomikalnega registra prenaša na vhod vezja XOR, izključeno stikalo (označeno z 0) pa pomeni, da bo na vhodu vezja XOR vedno logična 0. Stanja vseh stikal na sliki $(s_1, s_2, \ldots, s_N)$ lahko opišemo z zaporedjem $(1, 0, 1, 1, 0, 1, 1)$, kar pomeni, da se na vhodu vezja XOR pojavi stanje $(a_1, 0, a_3, a_4, 0, a_6, a_7)$. Stanje bitov v registru na sliki pa predstavimo z zaporedjem $(1, 0, 1, 1, 0, 0, 1)$.

Vezje XOR prešteje število enic, ki se pojavijo na vhodu in vrne logično 1, če je število enic liho, oz. logično 0, če je število enic sodo. Rezultat vezja XOR se ob pomiku bitov v desno, zapiše v prazno polje na levi strani registra. *(formalnejši zapis si lahko preberete v angleški verziji teksta.)*



korak	a_1	a_2	a_3	a_4	a_5	a_6	a_7	izhod
0	1	0	1	1	0	0	1	-
1	0	1	0	1	1	0	0	1
2	1	0	1	0	1	1	0	0
3	1	1	0	1	0	1	1	0
4	0	1	1	0	1	0	1	1
5	0	0	1	1	0	1	0	1
6	1	0	0	1	1	0	1	0
7	1	1	0	0	1	1	0	1
8	0	1	1	0	0	1	1	0
9	1	0	1	1	0	0	1	1
10	0	1	0	1	1	0	0	1
11	1	0	1	0	1	1	0	0
12	1	1	0	1	0	1	1	0
13	0	1	1	0	1	0	1	1
14	0	0	1	1	0	1	0	1

Ob danih začetni vrednostih bitov v registru in stanjih stikal, delovanje vezja ponazarja zgornja tabela. Izhodne vrednosti registra ponazarja stolpec 'izhod'.

Tvoja naloga je, da iz danih $2N$ izhodnih vrednosti tako povezanega registra določiš stanja stikal $(s_1, s_2, \ldots, s_N)$.

Vhodna datoteka

Prva vrstica vhodne datoteke `register.in` vsebuje velikost pomikalnega registra N v bitih ($1 \leq N \leq 750$). Druga vrstica vsebuje $2N$ številk 0 ali 1, ki brane po vrsti predstavljajo zaporedje prvih $2N$ izhodnih vrednosti registra.

Izhodna datoteka

Izhodna datoteka `register.out` naj vsebuje natanko eno vrstico, v kateri naj bodo izpisana stanja N stikal $(s_1, s_2, \ldots, s_N)$, ki povzročijo generiranje danih izhodnih vrednosti. Lahko se zgodi, da dano zaporedje izhodnih vrednosti vezja lahko povzročijo različne kombinacije stikal. V tem primeru naj vaš program izpiše poljubno izmed njih. Stanja stikal naj ponazarjajo številke 1 za vključena stikala in 0 za izključena stikala, številke pa naj bodo ločene s presledki.

V primeru, da ne obstaja nobena kombinacija stikal, ki bi povzročila generiranje danih izhodnih vrednosti, naj bo v edini vrstici izhodne datoteke izpisana vrednost -1.

Primeri

register.in	register.out
7 1 0 0 1 1 0 1 0 1 1 0 0 1 1	1 0 1 1 0 1 1

register.in	register.out
3 0 0 0 1 1 1	-1