Feladat első fele

A `feladat2.txt` szöveges fájl egy programkódot tartlamaz. Minden sorban egy utasítás van. Minden utasítás egy műveletből (`acc`, `jmp` vagy `nop`) és egy előjellel ellátott számból áll (mint például `+9` vagy `-10`). A programkód végrehajtása az első sortól indul és az utolsó sornál ér véget.

Az egyes műveletek a következőt csinálják:

- `acc`: Egy **belső változó értékék változtatja** az utasítást követő előjeles szám értékével. Például ha `+9` szerepel az `acc` után akkor a belső változó értéke 9-el nőni fog. A belső változó értéke a program indulásakor nulla. Az utasítás végrehajtása után a program a következő sorra ugrik.
- `jmp`: Egy **másik utasításra ugrik** a program és onnan folytatja a végrehajtást. Az adott sorhoz képest annyit ugrik a program amennyi a művelet utáni értékben szerepel. Például a `jmp +2` két sorral lentebb folytatja a végrehajtást (tehát átugorja a következő sort), a `jmp -10` pedig az adott sorhoz képest 10 sorral előbbi utasításra fog ugrani.
- `nop`: Ez a művelet **nem csinál semmit**, utána a program a következő sorra ugrik.

Vegyük a következő példaprogramot:

```
nop +0
acc +1
jmp +4
acc +3
jmp -3
acc -99
acc +1
jmp -4
acc +6
```

A végrehajtás lépései:

- Az első sorban szereplő `nop +0` nem csinál semmit. A következő (tehát a második) sorra ugrik a program.
- Az `acc +1` megnöveli a belső változó értékét ami 1 lett (0+1). A következő (harmadik) sorra ugrik a program.
- A `jmp +4` négyel lejjebbi (hetedik) sorra ugrik a program.
- Az `acc +1` szintén megnöveli a belső változó értékét ami 2 lett (1+1). A következő (nyolcadik) sorra ugrik a program.
- A `jmp -4` négyel feljebbi (negyedik) sorra ugrik a program.
- Az `acc +3` megnöveli a belső változó értékét ami az utasítás után 5 lett (2+3). A következő (ötödik) sorra ugrik a program.
- A `jmp -3` hárommal feljebbi (második) sorra ugrik a program.

Itt a végrehajtás végtelen ciklusba kerül, vagyis a utasítások végrehajtása sose ér véget és a program sose éri el az utolsó sort.

Kérdés

Mi a belső változó értéke mielőtt a `feladat2.txt` fájlban szereplő program végtelen ciklusba kerülne, tehát mielőtt egy olyan utasítást hajtana végre amit már egyszer végrejhajtott?

Feladat második fele

A programot ki lehet javítani úgy, hogy egyetlen utasítás műveletét `nop`-ról `jmp`-ra **vagy** `jmp`-ról `nop`-ra változtatjuk.

A példaprogramban ezt úgy tehetjük meg, hogy az utolsó előtti utasítás `jmp` műveletét `nop`-ra módosítjuk:

```
nop +0
acc +1
jmp +4
acc +3
jmp -3
acc -99
acc +1
nop -4  <==
acc +6
```

A javított program végrehajtásának lépései:

- Az első sorban szereplő `nop +0` nem csinál semmit. A következő (tehát a második) sorra ugrik a program.
- Az `acc +1` megnöveli a belső változó értékét ami 1 lett (0+1). A következő (harmadik) sorra ugrik a program.
- A `jmp +4` négyel lejjebbi (hetedik) sorra ugrik a program.
- Az `acc +1` szintén megnöveli a belső változó értékét ami 2 lett (1+1). A következő (nyolcadik) sorra ugrik a program.
- A **javított** `nop -4` nem csinál semmit. A következő (kilencedik) sorra ugrik a program.
- Az `acc +6` megnöveli a belső változó értékét ami az utasítás után 8 lett (2+6). Mivel ez volt az utolsó sor, a program futása véget ér.

A `feladat2.txt` fájlban szereplő programot szintén ki lehet javítani ugyanezzel a módszerrel.

Kérdés

Mi lesz a belső változó értéke miután a `feladat2.txt` fájlban szereplő program javított verziója lefutott, tehát az utolsó sorban szereplő utasítás végrehajtása után?