# Bitek listájának eliminálása

## Feladat ismertetése

A feladatban arra keressük a választ, hogy egy biteket tartalmazó sorozat eliminálható-e definiált szabályok mentén. Egy sorozatot akkor tekintünk eliminálhatónak, ha az azonos elemekből álló csoportokat (azonos szomszédos elemek) valamilyen sorrendben, fokozatosan elhagyva a sorozatból, a lépéssorozat végére egy üres sorozatot kapunk.

Például, ha az `10110001` sorozatból elhagyjuk az `11` csoportot, akkor az `100001` sorozatot kapjuk. Ebből az egyetlen eliminálható `0000` csoportot elhagyva az `11` sorozatot kapjuk. Elhagyva az `11` csoportot az üres sorozatot kapjuk. Azaz létezik olyan lépéssorozat, amely végrehajtásával a sorozat összes eleme eliminálható.

Természetesen ez nem minden esetben ennyire egyszerű, hiszen ha az `10001101` sorozatból indulunk ki, és hasonlóan az első eliminálható csoporttal indítjuk a redukálást, akkor az `11101` sorozatot, majd a `01` sorozatot kapjuk. Ez egy "zsákutcának" tekinthető, hiszen ez azt mutatja, hogy a sorozat nem eliminálható. Azonban ha a második eliminálható csoporttal indultunk volna, akkor az `100001`, majd az `11` és végül az üres sorozatot kaptuk volna.

Látható, hogy egy hosszabb sorozat esetén több különböző sorrendben elvégezhetőek az eliminálási lépések, de nem mindegyik vezet eredményre. Természetesen az is előfordulhat, hogy semmilyen módon nem eliminálható a sorozat.

A feladat során ezt a kérdést fogjuk alaposabban megvizsgálni.

A feladatban egy sorozatot `0` és `1`-es értékeket tartalmazó listaként adunk meg, ezért bevezetünk `BitList` típusszinonímát az egészek listájára.

```
type BitList = [Int]
```

### Eliminálható csoport

Eliminálható csoport alatt a legalább két elemű, azonos értékeket tartalmazó bitsorozatot értjük. Fontos, hogy az összes azonos szomszédos elem alkotja a csoportot, és az nem bontható fel.

Például, a `011110` sorozatban az `1111` bitsorozatot tekintjük eliminálható csoportnak és mást nem.

### Eliminálási lépés

Eliminálási lépésnek nevezzük azt, amikor egy eliminálható csoportot elhagyunk a sorozatból. Egy eliminálási lépés végrehajtásával egy csoport eltűnik a sorozatból, más csoportok egyesülhetnek egy hosszabb csoportot alkotva.

**Egy példa az eliminálásra**

A következő táblázatban látható egy példa az eliminálási lépés működésére.

| Bitlista | Stratégia |
|---|---|
| `[0,0,0,1,1,0,0,1,1,1]` | Az első csoportot eliminálják. |
| `[1,1,0,0,1,1,1]` | Az első csoportot eliminálják. |
| `[0,0,1,1,1]` | Az első csoportot eliminálják. |
| `[1,1,1]` | Az első csoportot eliminálják. |
| `[]` | Üres listához jutottunk, az eliminálás sikeres. |

## Első szint: Eliminálható csoportok száma (2 pont)

Adjuk meg azt a függvényt, amely egy `BitList`-ről megadja, hogy hány eliminálható csoport található benne.

```
countEliminable :: BitList -> Int
```

Néhány teszteset:

```
countEliminable [1,0,0,1] == 1
countEliminable [0,0,1,1,1,0,1,1] == 3
countEliminable [] = 0
```

## Második szint: n-edik eliminálható csoport elhagyása (5 pont)

Definiáljuk azt a függvényt, amely egy `BitList`-ből eliminálja az **N-edik** eliminálható csoportot. Az indexelés **egytől indul**, ha a kapott index **kisebb** mint egy, akkor az **elsőt** kell eliminálni, ha pedig **nagyobb** mint az eliminálható csoportok száma, akkor az **utolsót** kell eliminálni. Abban az esetben, ha a bitlistában **nem szerepel** eliminálható csoport, az **eredeti** bitlistával térjen vissza.

```
eliminateNth :: BitList -> Int -> BitList
```

Néhány teszteset:

```
eliminateNth [0,0,1,1,1,0,1,1] 2 == [0,0,0,1,1]
eliminateNth [1,0,0,1,1,0,1,1,1] 5 == [1,0,0,1,1,0]
eliminateNth [1,1,0,1,0,0,1] 0 == [0,1,0,0,1]
```

## Harmadik szint: Eliminálhatóság eldöntése (10 pont)

Adjuk meg azt a függvényt, amely eldönti egy `BitList`-ről, hogy teljesen eliminálható-e, azaz van egy olyan lépéssorozat, amely segítségével eliminálható a sorozat.

Egy bitlista eliminálhatósága nem minden esetben triviális. A csoportok eliminálása során új csoportok keletkezhetnek, így az eliminálási sorrend lényeges.

### Egy példa a sorrend fontosságára

Ha mindig az első adandó csoportot eliminájuk, akkor azt kapjuk az alábbi példában, hogy a lista nem eliminálható.

| Eliminálási folyamat | Hanyadik csoportot eliminájuk |
|---|---|
| `[1,1,0,0,0,1,0,0]` | 1. |
| `[0,0,0,1,0,0]` | 1. |
| `[1,0,0]` | 1. |
| `[1]` | Nem tudunk tovább eliminálni. |

Más sorrend mellett viszont az eliminálás sikeres lesz.

| Eliminálási folyamat | Hanyadik csoportot eliminájuk |
|---|---|
| `[1,1,0,0,0,1,0,0]` | 2. |
| `[1,1,1,0,0]` | 1. |
| `[0,0]` | 1. |
| `[]` | Az eliminálás sikeres. |

A feladat tehát meghatározni, hogy **van-e legalább egy** olyan eliminálási sorrend, amellyel a listát **teljesen** eliminálni tudjuk.

```
elimination :: BitList -> Bool
```

Néhány teszteset:

```
elimination [1,0,0,1] == True
elimination [] == True
elimination [1,1,0,1,0,0,1] == False
```

## Bónusz feladat

**A következő feladat nem kerül pontozásra, ez a feladat továbbgondolása!**

Bónusz feladatként definiálható egy olyan függvény, ami kiírja az eliminálható bitlista egy lehetséges eliminálási lépéseit. Tehát egy listába kigyűjti az eliminálási lépések utáni pillanatnyi állapotot.