Feladat:

Nyitrai Bence - Milliomos játék

Egy szöveges fájl egy milliomos játék kérdéseit és válaszait tartalmazza, minden kérdéset 7 sorban, az alábbi formátunmban:

#1423
Mikor volt a mohácsi csata?
A: 1523
B: 1643
C: Nem is volt ilyen csata
D: 1526
helyes válasz: D

Egy másik szöveges fájl játékokat tartalmaz, minden játékról az alábbiakat:

Játékos neve
Kérdés-válaszlista

pl.

Kovács Mariann
#1324 B #2435 D #5342 C ...
Horváth Miklós
#4235 A #6345 B #2435 D

A program olvassa be a két fájlt, és listázza ki azokat a helytelen kérdés-válasz párokat, melyek 50%-nál gyakrabban fordulnak elő.

Adatszerkezetek:

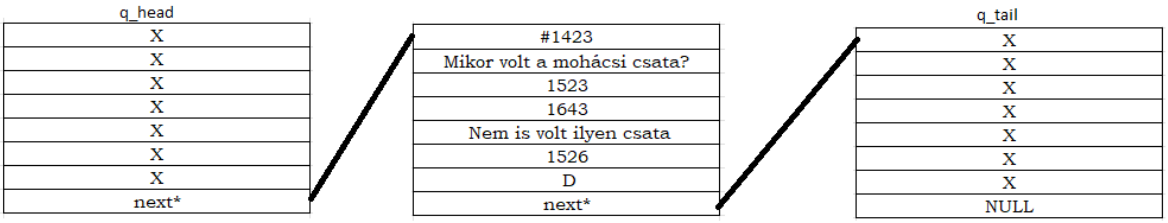
- Fájlok felépítése:
  - Input fájl1:
    - 1.sor: kérdés azonosítója
    - 2.sor: kérdés
    - 3.sor: A válasz
    - 4.sor: B válasz
    - 5.sor: C válasz
    - 6.sor: D válasz

        - 7.sor: helyes válasz azonosítója
    - Input fájl2:
        - 1.sor: játékos neve
        - 2.sor: kérdés azonosító, játékos válasza
    - Output fájl:
        - helytelen kérdés-válasz párok, amelyek 50%-nál gyakrabban fordulnak elő
- answer struktúra:
    - valaszid (char): a válaszopció azonosítóját tartalmazza, az első fájlból kerül beolvasásra
    - valasz (string): magát a választ tartalmazza, az első fájlból kerül beolvasásra
- question struktúra:
    - kerdesid (int): tartalmazza a kérdés azonosítóját, az első fájlból kerül beolvasásra
    - kerdes (string): a kérdést tartalmazza, az első fájlból kerül beolvasásra
    - A (answer): az A választ tartalmazza
    - B (answer): a B választ tartalmazza
    - C (answer): a C választ tartalmazza
    - D (answer): a D választ tartalmazza
    - helyesid (char): a helyes válasznak az azonosítóját tartalmazza, az első fájlból kerül beolvasásra
    - next (struct question*): a következő elemre mutató pointer

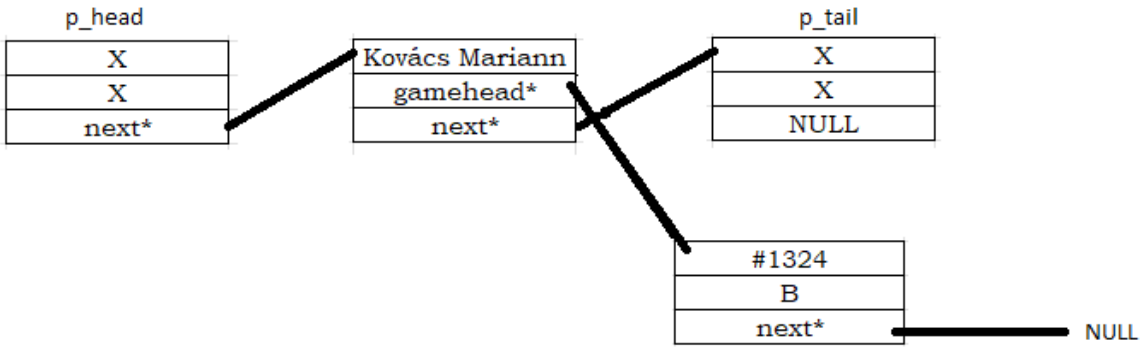
- game struktúra:
    - kerdesid (int): annak a kérdésnek az azonosítója, amit a játékos kapott, második fájlból kerül beolvasásra
    - valaszid (char): a játékos szerinti válaszopció, a második fájlból kerül beolvasásra
    - next (struct game*): a következő elemre mutató pointer
- player struktúra:
    - nev (string): a játékos nevét tartalmazza, a második fájlból kerül beolvasásra
    - g_head (game*): az adott játékos game listájának az elejére mutató pointer
    - next (struct player*): a következő elemre mutató pointer
- Globális változók:

- q_head (question*): a question lista első strázsája
- q_tail (question*): a question lista hátsó strázsája
- p_head (player*): a player lista első strázsája
- p_tail (player*): a player lista hátsó strázsája

1. Láncolt listák:
   - a kérdés adatait tároló láncolt lista: elől-hátul strázsás, feltöltése a program inicializálásakor



   - a játékos adatait és a játék menetét tároló fésűs lista: a player lista elől-hátul strázsás, a game lista nem rendelkezik strázsával, feltöltésük a program inicializálásakor



2. Dinamikusan foglalt tömb:
   - a q_id egy calloc()-al foglalt kezdetben 10 elem nagyságú tömb, funkciója: ha egy kérdést megvizsgálunk, beletöltjük a tömbbe az azonosítóját, hogy azt a kérdést ne vizsgáljuk meg mégegyszer a későbbiekben, mivel akkor hibás eredményt kapnánk
   - méretét realloc()-al exponenciálisan növeljük, ha szükséges

Függvények:

- `void read_to_question(question *head)`
  - az input1.txt -ből beolvassa az adatokat head által mutatott question listába
  - a kérdések és a válaszok hossza maximum 500 karakter lehet egyenként, mivel struktúrában előre meg van adva a string mérete
  - a beolvasott kerdesid nem lehet #0, mert a need_to_test függvény hibásan működik erre az értékre
- `void read_to_player_and_game(player *head)`
  - az input2.txt tartalmával feltölti a head által mutatott player – game fésűs listát
  - egy játékos neve maximum 100 karakterből állhat, azonos okból, mint a kérdés hossz
- `char good_answer(question *head, int gameid)`
  - megkapja a question lista fejét(head) és a jelenlegi kérdés azonosítóját(gameid)
  - visszatér az adott kérdés jó válaszának az azonosítójával
  - függvény jó működéséhez szükséges, hogy a keresett kerdesid benne legyen a question listában!
- `int need_to_test(int q_id[], int l, int kerdesid)`
  - bemenetként megkapja a q_id dinamikusan foglalt tömböt (tömb kezdőcíme: q_id[] és a hossza: l átvételével) és a vizsgálandó kérdés azonosítóját(kerdesid)
  - megnézi, hogy a kerdesid benne van-e a q-id tömbben, ha igen akkor igaz értékkel tér vissza (1-el), ha nem akkor hamissal(0-al)
  - kerdesid nem lehet nulla, mert a calloc-tól kinullázott tömbként kapjuk meg a q_id -t, erre az estetre hibásan működik a need_to_test függvény, mivel hamissal tér vissza #0 azonosítójú kérdésre, amikor még nincs letesztelve
- `char check_current(question *q_head, player *p_current, game *g_current)`
  - megkapja a question lista fejét(q_head), a jelenleg vizsgálandó player-re mutató pointert(p_current) és azt a pointert, ami megmutatja, hogy hol tartunk a jelenlegi player - game listájának vizsgálatában (g_current)
  - ha a vizsgált kérdésre a helytelen kérdés-válasz párok közül az egyik, meghaladta az 50%-ot, akkor visszaadja azt a válaszazonosítót, különben '0'-t ad vissza
- `void destroy_question(question *q_head)`
  - megkapja a question lista fejét(q_head) és felszabadítja a question lista által lefoglalt memóriát
- `void destroy_player_and_game(player *p_head, player *p_tail)`

- megkapja a player lista fejét(p_head) és a farkát(p_tail), visszaadja az operációs rendszernek a player – game fésűs lista által lefoglalt memóriát
- a player lista első és a hátsó strázsáját külön szabadítja fel

Algoritmusok:

1. Inicializálás:
   - létrehozni a question listát
   - létrehozni a player és a benne levő game listát
   - az első fájlból beolvasni az adatokat a listákba
   - a második fájlból beolvasni az adatokat a listákba
2. Bejárjuk minden emberhez tartozó game listát egy adott kérdésre, a bejárás során megnézzük a kérdésre adott választ, és hogy hány ember kapta meg az adott kérdést.
3. Ha az azonos rossz válaszok száma osztva az összes válasz számával több mint 50%, akkor ezt a kérdés-válasz párt kiírjuk a kimenet fájlba.
4. Ha van még kérdés, amit nem vizsgáltunk, akkor ugrás: 2. (egy másik kérdésre, ha már az adott game listában nincs több kérdés, akkor player lista következő elemére ugorjunk, és kezdjük el az ő game listáját bejárni)
5. Ha a game listákról az összes kérdést megvizsgáltuk: Vége.

Tesztelés:

1. test1.txt: minden játékos jól válaszol az összes kérdésre: helyes a kimenet ✅(semmi)
2. test2.txt: minden játékos ugyanazokkal a rossz válaszokkal válaszol az adott kérdésekre: helyes a kimenet ✅ (az összes kerdesid és a leggyakoribb rossz valaszid)
3. test3.txt: ugyanannyi játékos válaszol egy rossz valaszid -vel, mint amennyi jóval: helyes a kimenet ✅(semmi)
4. test4.txt: bizonyos játékosok ugyanazzal a rossz valaszid-vel válaszolnak, de ez kevesebb mint az összes játékos fele: helyes a kimenet ✅(semmi)
5. test5.txt: játékosok fele egy rossz valaszid-t jelöl, a másik fele egy másik rossz valaszid-t jelöl: helyes a kimenet ✅(semmi)
6. input1.txt: random teszt: helyes a kimenet ✅(#1423 C #420 C)