Programozási alapismeretek
beadandó feladat
Törzsek: H, P, TT

Készítette: Prorok Ernest
Neptun-azonosító: FILSPA
E-mail: ernest.prorok@gmail.com

Kurzuskód: IP-08PAEG/5
Gyakorlatvezető neve: Menyhárt László Gábor

2015. május 3.

Tartalom

Felhasználói dokumentáció 3
Feladat 3
Környezet 3
Használat 3
A program indítása 3
A program bemenete 3
A program kimenete 3
Minta bemenet és kimenet 4
Hibalehetőségek: 4
Fejlesztői dokumentáció 6
Feladat 6
Specifikáció 6
Környezet 6
Forráskód 6
Megoldás 7
Tesztelés 17
Fejlesztési lehetőségek 18

Felhasználói dokumentáció

Feladat

A jelenlegi Dél-Afrikai Köztársaság területén számos törzs élt, amikor a fehér gyarmatosítók megjelentek. Közülük egyesek békében éltek, mások időről időre ellenséges viszonyba keveredtek vagy éppen harcban álltak szomszédjaikkal. A háborúkat kezdőidő szerinti sorrendben ismerjük a XVII., XVIII., XIX. századból.

Sorszám	Feladat szövege
H	Add meg, hány 5 évnél tovább tartó háború volt!
P	Add meg mindhárom vizsgált évszázadra a háborúzó törzsek számát!
TT	Adj meg egy törzset, amely csak egyszer háborúzott!

Környezet

IBM PC, exe futtatására alkalmas operációs rendszer (pl. Windows XP). Nem igényel egeret.

Használat

A program indítása

A program a bead\bin\Debug\bead.exe néven található a tömörített állományban. A bead.exe fájl kiválasztásával indítható.

A program bemenete

A program az adatokat a billentyűzetről vagy fájlból olvassa be a következő sorrendben:

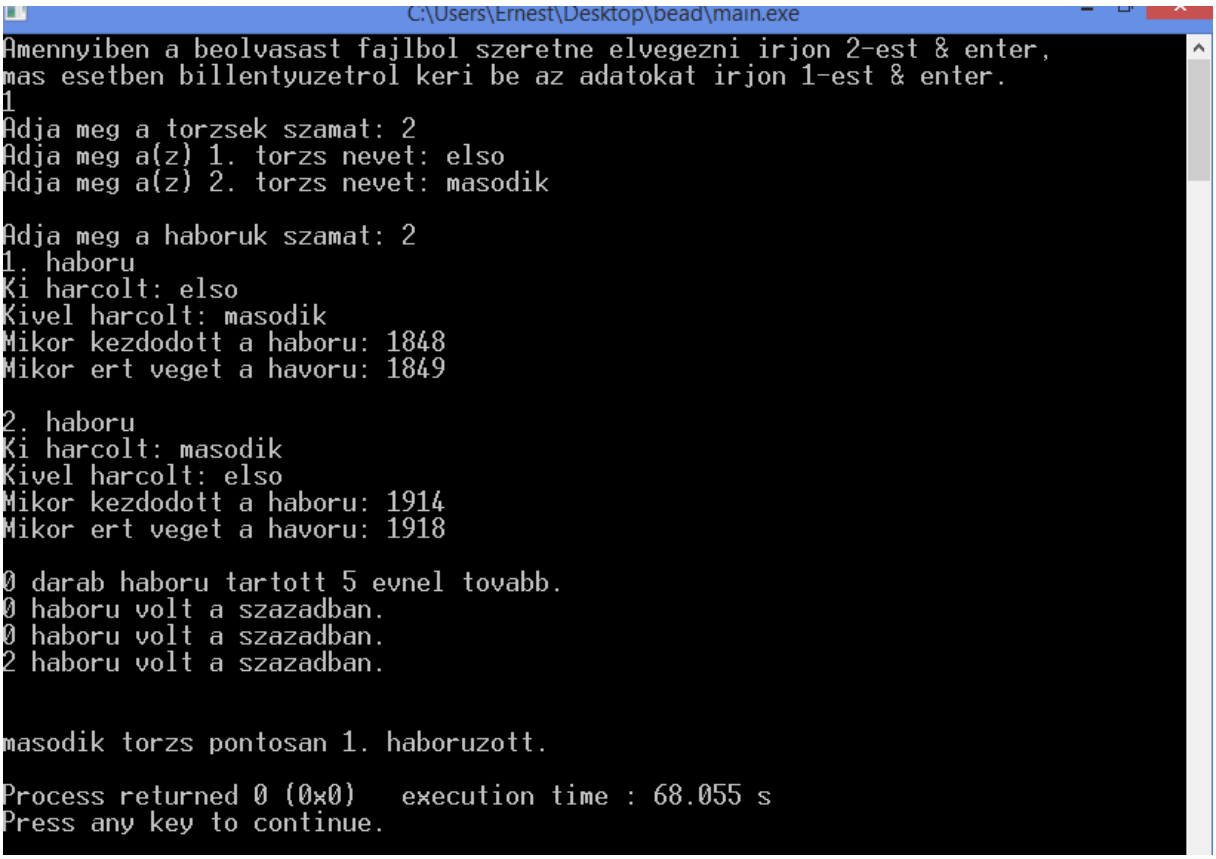
Sorindex	Adat	Magyarázat
0.	*bemód*	A beolvasás módja (fájlból olvasás esetén 2, billentyűzetről olvasás esetén 1)
1.	*torzs*	törzsek száma
2.	*torzsek*	törzsek neve
3.	*haboruk*	háborúk száma
4.	*ki*	ki háborúzott
5.	*kivel*	kivel háborúzott
6.	*mettol*	háború kezdete
7.	*meddig*	háború vége

A program kimenete

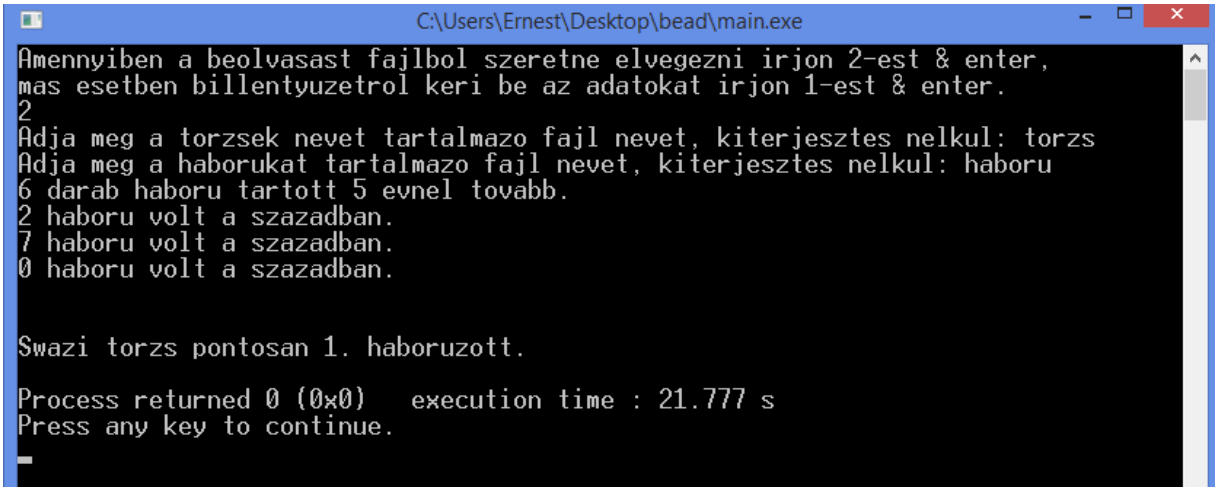
A program először kiírja az öt évnél tovább tartó háborúk számát, majd századonként a háborúk számát, valamint egy törzs nevét, amelyik csak egyszer háborúzott.

Minta bemenet és kimenet

Mintafutás billentyűzetről való beolvasás esetén:



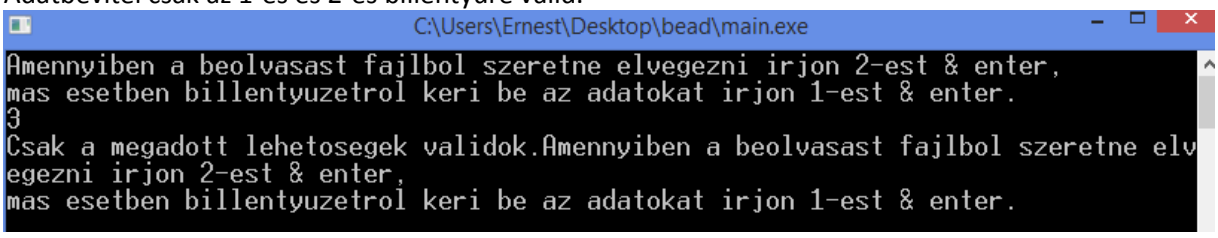
Mintafutás fájlból való beolvasás esetén:



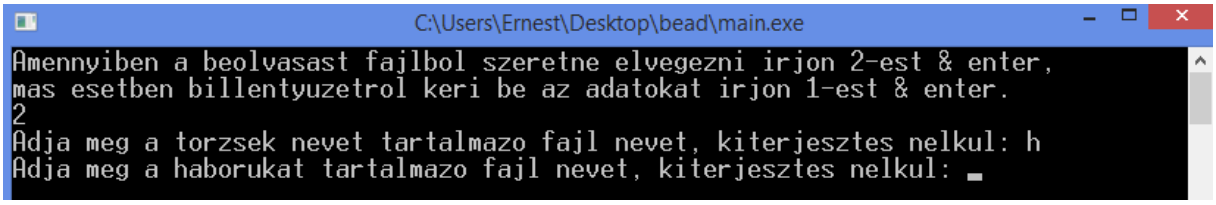
Hibalehetőségek:

Mintafutás hibás bemeneti adatok esetén:

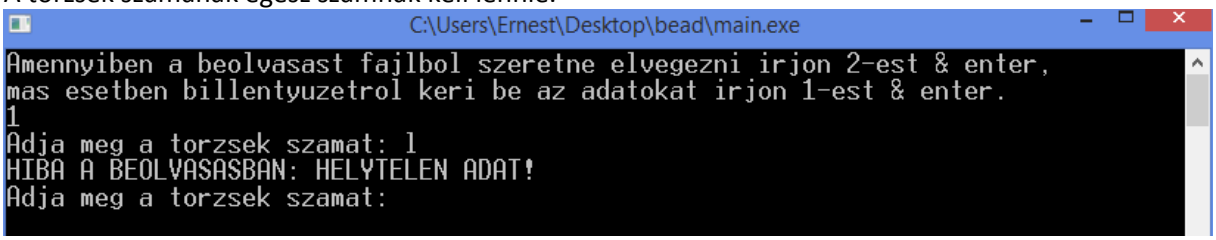
Adatbevitel csak az 1-es és 2-es billentyűre valid.



Nem létező fájl beolvasásánál hibát ír ki.



A törzsek számának egész számnak kell lennie.



Fejlesztői dokumentáció

Feladat

A jelenlegi Dél-Afrikai Köztársaság területén számos törzs élt, amikor a fehér gyarmatosítók megjelentek. Közülük egyesek békében éltek, mások időről időre ellenséges viszonyba keveredtek vagy éppen harcban álltak szomszédjaikkal. A háborúkat kezdőidő szerinti sorrendben ismerjük a XVII., XVIII., XIX. századból.

Sorszám	Feladat szövege
H	Add meg, hány 5 évnél tovább tartó háború volt!
P	Add meg mindhárom vizsgált évszázadra a háborúzó törzsek számát!
TT	Adj meg egy törzset, amely csak egyszer háborúzott!

Specifikáció

Bemenet: $\mathrm{N} * \mathrm{torzs} * \mathrm{haboruk} \in \mathbb{Z}, \mathrm{War} \in \mathrm{war}^{\mathrm{N}}, \mathrm{ki} * \mathit{kivel} * \mathit{mettol} * \mathit{meddig} \in \mathrm{S}$
Előfeltétel: $\mathrm{N} > 2, \forall \mathrm{i}(1 \leq \mathrm{i} \leq \mathrm{N}) : \mathrm{torzs_i}.\mathrm{torzsek} \neq '', \mathrm{haboruk_i}.\mathrm{ki} \geq 0'', \mathrm{haboruk_i}.\mathrm{kivel} \geq 0$
H feladat
Kimenet: darab, haboruMax$\in \mathbb{Z}$
Utófeltétel: war[i].meddig-war[i].mettol
P feladat
Kimenet: warSzazad, fordulo$\in \mathbb{Z}$
Utófeltétel: fordulo-100 & war[k].mettol < fordulo && war[k].mettol > _fordulo
TT feladat
Kimenet: torzsek$\in \mathbb{Z}$
Utófeltétel: torzsek[i] == war[j].ki || torzsek[i] == war[j].kivel

Környezet

IBM PC, exe futtatására alkalmas operációs rendszer (pl. Windows XP). C++ fordítóprogram (gcc v4.4.1), Code::Blocks fejlesztői környezet.

Forráskód

A teljes fejlesztői anyag a bead nevű könyvtárban található meg. A fejlesztés során használt könyvtár-struktúra:

Állomány	Magyarázat
bead\bin\Debug\bead.exe	nyomkövethető állapotú futtatható kód
bead\obj\Debug\main.o	nyomkövethető állapotú, félig lefordított (object-) kód
bead\bead.cbp	projektfájl,
bead\main.cpp	C++ forrás
bead\bead.layout	layout file
bead\src\torzs.txt	tesztfájl#1
bead\src\haboru.txt	tesztfájl#2
bead\bead.depend	depend file
bead\main.o	nyomkövethető állapotú, félig lefordított (object-) kód
bead\main.exe	nyomkövethető állapotú futtatható kód
bead/dokumentacio.pdf	a dokumentáció

Megoldás

Programértékek

Konstans

n:egész [a törzsek max száma]

Típus

torzs=rekord(

ki, kivel: szöveg,

mettol: egész,

meddig: egész)

Változó

line tömb[torzsek]

eldontes tömb[war]

szam egész

H egész

P tömb[War]

TT tomb[torzsek]

P tömb[megallo]

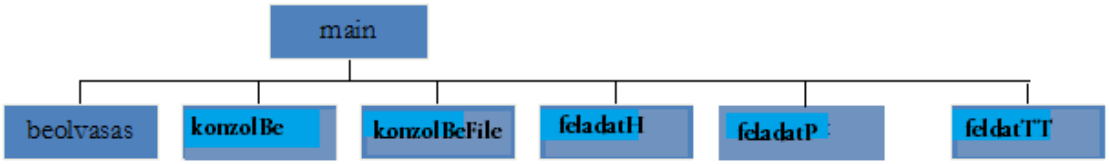
Programfelépítés

A program által használt modulok (és helyük):

bead – main.cpp

iostream, fstream, stdlib.h, string, stdio.h– a C++ rendszer része.

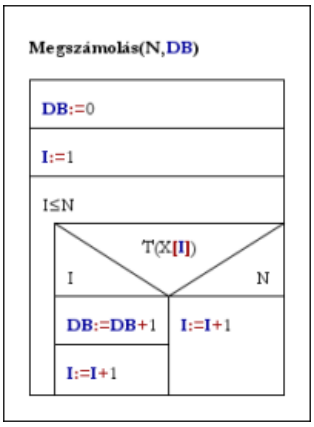
Függvénystruktúra



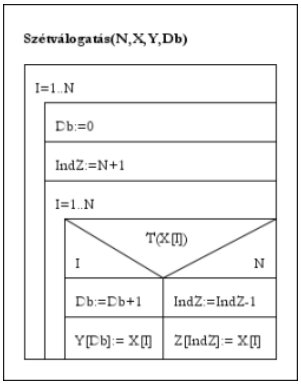
Algoritmus

Az algoritmizálás szempontjából a részfeladatokat megoldó alprogramok érdekesek: Ezek algoritmusa a következő:

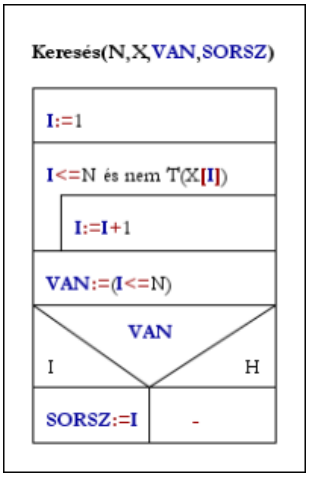
H feladat



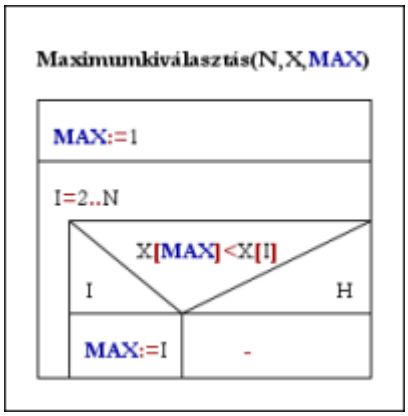
P feladat



TT feladat



és



A kód

```
//Prorok Ernest, FILSPA, torzsek: H,P,TT

#include <iostream>

#include <fstream>  // fajl

#include <string>   // string, fajl beolvas, fajlnev

#include <stdlib.h> // ki

#include <stdio.h>  // ki


using namespace std;


const int MAXN = 100;   // max elem


// haboruk deklarasa

struct War {

   string ki, kivel;

   int mettol, meddig;

};

// void

void konzolBe(int& torzs, int& haboruk, string torzsek[MAXN], War war[MAXN]);

void konzolBeFile(int& torzs, int& haboruk, string torzsek[MAXN], War war[MAXN]);


void feladatH(const int haboruk, const War war[MAXN]);

void feladatP(const int haboruk, const int torzs, const string torzsek[MAXN], const War war[MAXN]);

void feladatTT(const int haboruk, const int torzs,  const string torzsek[MAXN], const War
war[MAXN]);


int main() {

// 1. dekl

   int torzs;

   // hany torzs
```

```
    int haboruk;
    // hany haboru
    string torzsek[MAXN];
    // torzs
    War war[MAXN];
    // haboru

    int choice;
    // bekeres
    bool hiba;
// 2. beolv
    do {
        cout << "Amennyiben a beolvasast fajlbol szeretne elvegezni irjon 2-est & enter,"<<endl<<"mas esetben billentyuzetrol keri be az adatokat irjon 1-est & enter." << endl;
        cin >> choice;

        hiba = cin.fail() || choice > 2 || choice < 1;

        if (hiba) {
            cout << "Csak a megadott lehetosegek validok.";

            cin.clear();
            cin.ignore(1000, '\n');
        }
    } while (hiba);

    if (choice == 2) konzolBeFile(torzs, haboruk, torzsek, war);
    else konzolBe(torzs, haboruk, torzsek, war);

// 3. feladat
 // H
```

```
 feladatH(haboruk, war);
 // P
 feladatP(haboruk, torzs, torzsek, war);
 // TT
 feladatTT(haboruk, torzs, torzsek, war);
   return 0;
}

void konzolBe(int& torzs, int& haboruk, string torzsek[MAXN], War war[MAXN]) {
   bool hiba;

   do {
      cout << "Adja meg a torzsek szamat: ";
      cin >> torzs;

      hiba = torzs < 0 || cin.fail();

      if (hiba) {
         cout << "HIBA A BEOLVASASBAN: HELYTELEN ADAT!" << endl;

         cin.clear();
         cin.ignore(1000, '\n');
      }
   } while(hiba);

   for (int i = 0; i < torzs; i++) {
      cout << "Adja meg a(z) " << i+1 << ". torzs nevet: ";
      cin >> torzsek[i];
   }

   if (torzs > 1) {
```

```
        cout << endl << "Adja meg a haboruk szamat: ";
        cin >> haboruk;

        for (int i = 0; i < haboruk; i++) {
            cout << i+1 << ". haboru" << endl;

            cout << "Ki harcolt: ";
            cin >> war[i].ki;
            cout << "Kivel harcolt: ";
            cin >> war[i].kivel;

            do {
                cout << "Mikor kezdodott a haboru: ";
                cin >> war[i].mettol;

                hiba = war[i].mettol < 1600 || cin.fail();
                if (hiba) {
                    cout << "HIBA A BEOLVASASBAN: HELYTELEN ADAT!";

                    cin.clear();
                    cin.ignore(1000, '\n');
                }
            } while(hiba);

            cout << "Mikor ert veget a havoru: ";
            cin >> war[i].meddig;

            cout << endl;
        }
    }
}
```

```
void konzolBeFile(int& torzs, int& haboruk, string torzsek[MAXN], War war[MAXN]) {
    string pathTorzs, pathHaboru, fileTorzs, fileHaboru;

    cout << "Adja meg a torzsek nevet tartalmazo fajl nevet, kiterjesztes nelkul: ";
    cin >> fileTorzs;
    fileTorzs.append(".txt");

    cout << "Adja meg a haborukat tartalmazo fajl nevet, kiterjesztes nelkul: ";
    cin >> fileHaboru;
    fileHaboru.append(".txt");

    pathTorzs = pathHaboru = "src/";
    pathTorzs.append(fileTorzs);
    pathHaboru.append(fileHaboru);

    // torzs tomb feltoltese
    ifstream torzsIn(pathTorzs.c_str(), ios::in);
    if (torzsIn.is_open()) {
        torzs = 0; // elemek szama

        while(!(torzsIn.eof())) {
            torzsIn >> torzsek[torzs];
            torzs++;
        }

    torzsIn.close();
    } else {
        cout << "HIBA, NEM SIKERULT MEGNYITNI A FAJLT." << endl;
        exit(1);    // kilepes (ha a file nem valid)
    }
```

```
    // a haboruk feltoltese
    ifstream haboruIn(pathHaboru.c_str(), ios::in);
    if (haboruIn.is_open()) {
        haboruk = 0;    // elemek szama

        while(!(haboruIn.eof())) {
            haboruIn >> war[haboruk].ki;
            haboruIn >> war[haboruk].kivel;
            haboruIn >> war[haboruk].mettol;
            haboruIn >> war[haboruk].meddig;

            haboruk++;
        }

    haboruIn.close();
    } else {
        cout << "HIBA, NEM SIKERULT MEGNYITNI A FAJLT." << endl;
        exit(1);    // kilep (ha a file nem valid)
    }
}

void feladatH(const int haboruk, const War war[MAXN]) {
// 5 evnel hosszabb haboruk szama
    int warIndex = 0;
    int haboruMax = war[0].meddig - war[0].mettol;

    int darab = 0;

   for (int i = 0; i < haboruk; i++) {
        if (war[i].meddig - war[i].mettol > 5) {
            darab++;
```

```
        }
    }

    cout << darab <<" darab haboru tartott 5 evnel tovabb."<< endl;
}

void feladatP(const int haboruk, const int torzs, const string torzsek[MAXN], const War war[MAXN]) {
    int warSzazad[3] = {0};
    int fordulo = 1700;
    int _fordulo = fordulo - 100;

    for (int i = 0; i < 3; i++) {

        for (int j = 0; j < torzs; j++) {
            for (int k = 0; k < haboruk; k++) {
                if (war[k].mettol < fordulo && war[k].mettol > _fordulo) {
                    if (war[k].ki == torzsek[j] || war[k].kivel == torzsek[j]) {
                        warSzazad[i]++;
                        break;
                    }
                }

                else if (war[k].mettol < fordulo && war[k].mettol >= (_fordulo + 100)) {
                    if (war[k].ki == torzsek[j] || war[k].kivel == torzsek[j]) {
                        warSzazad[i+1]++;
                        break;
                    }
                }
            }
        }
```

```
        fordulo += 100;
        _fordulo += 100;
    }

    for (int i = 0; i < 3; i++) {
        cout << warSzazad[i] <<" haboru volt a szazadban."<< endl;
    }

    cout << endl << endl;
}

void feladatTT(const int haboruk, const int torzs,  const string torzsek[MAXN], const War war[MAXN])
{

    // kereses es maxkival tetel
    int mennyivel[MAXN] = {0};

    // kereses tetel
    for (int i = 0; i < torzs; i++) {
        for (int j = 0; j < haboruk; j++) {
            if (torzsek[i] == war[j].ki || torzsek[i] == war[j].kivel) {
                mennyivel[i]++;

                if (j <= haboruk-1) {
                    for (int k = j+1; k < haboruk; k++) {
                        if ((war[j].ki == war[k].ki && war[j].kivel == war[k].kivel) || (war[j].kivel == war[k].ki &&
war[j].ki == war[k].kivel)) {
                            mennyivel[i]--;
                        }
                    }
                }
            }
```

```
    }
  }

  int melyik;    // legnagyobb elem

  // maxkival tetel
  for (int i = 0; i < torzs; i++) {
    if (mennyivel[i] == 1) {
      melyik = i;
    }
  }

  cout << torzsek[melyik] << " torzs pontosan 1. haboruzott." << endl;
}
```

Tesztelés

Érvényes teszteset

Bemenet	
N = 2	
1	2
2	elso masodik
3	1848-1849 1914-1918
Kimenet	
H	0
P	1 1
TT	x.y.

Érvénytelen teszteset

Bemenet	
N = 0	
Kimenet	
H	-
P	-
TT	-

Fejlesztési lehetőségek

Amennyiben a felhasználó szeretné saját maga is létrehozhatna közvetlenül a programon keresztül állományokat, melynek adatait saját magunk adjuk meg a programon keresztül, illetve lehetőség lenne generált (random) adatokkal való munkára. Valamint a P feladat bontásában a századokon átnyúló háborúk validálása.