EX PRA TORN 1 X94998_ca

Aquest és un problema de Jutge per fer lliuraments del examen practica.
Descarregueu els fitxers públics per obtenir el material:

- enunciat
- fitxer llegeixme.txt
- codi ja implementat
- document ús de diccionaris
- fitxer Makefile
- joc de proves públic
- fixer.zip amb la documentació

La classe `Poblacion` té els següents atributs:

```
struct info_indiv {
Individuo whoIam;

/** @brief Nom del primer progenitor
 *
 * $ si no te ascendents */
string nom_pare;

/** @brief Nom del segon progenitor
 *
 * $ si no te ascendents */
string nom_mare;

/** @brief Iterador a l'element primer progenitor
 *
 * end() si no te ascendents */
 map<string,info_indiv>::const_iterator it_pare;

/** @brief Iterador a l'element segon progenitor
 *
 * end() si no te ascendents */
map<string,info_indiv>::const_iterator it_mare;
};

/** @brief Conjunt dels individus */
        map<string,info_indiv> m;
```

Podeu treballar tant amb els noms dels individus com amb els iteradors als elements del diccionari. Als fitxers públics hi ha un document sobre l'ús de diccionaris.
Cal implementar una de les dues seguents operacions de la classe `Poblacion`:

```
/** @brief Consultora de si un individu es descendent de l'altre
\pre it1 i it2 referencien dos individus existents al p.i.
\post el resultat indica si l'individu referenciat per it1 es descendent de
      l'individu referenciat per it2
```

```
*/
bool descendent_iterador(map<string,info_indiv>::const_iterator it1,
        map<string,info_indiv>::const_iterator it2) const;

/** @brief Consultora de si un individu es descendent de l'altre
\pre nom1 i nom2 són noms de dos individus al p.i.
\post el resultat indica si l'individu nom1 és descendent de
      l'individu nom2
*/
bool descendent_nom(const string & nom1, const string & nom2) const;
```

i també cridar a l'operació implementada al codi de:

```
/** @brief Consultora de si dos individus poden formar parella
\pre s1 i s2 son individus del sistema
\post el resultat és cert si s1 i s2 no són germans i cap d'ells
és ascendent de l'altre */
bool poden_ser_parella(const string & s1, const string & s2) const;
```

Per últim cal implementar l'operació de la classe `Par_Crom`:

```
/** @brief Creua un parell de cromosomes donada una serie de punts de tall
 *
\pre Al canal estandar d'entrada hi ha els punts de tall seguits de -1
\post Els dos cromosomes del parell del p.i. s'han creuat segons els punts de tall
*/
void creuar();
```

Fixeu-vos a l'enunciat com es fa el creuament. És diferent que el de la pràctica. Podeu fer servir l'operació de la classe `Par_Crom`:

```
/** @brief Intercanvia un mateix segment entre cr1 i cr2
 *
\pre 1 <= ini_tram <= fi_tram <= cr1.size()
\post S'ha intercanviat el contigut dels segments cr1[ini_tram-1..fi_tram-1] i cr2[ini_tram-1..fi_
*/
        void intercanviar(int ini_tram, int fi_tram);
```

Copieu aquesta plantilla en un fitxer anomenat `solution.cc` i completeu-la. El vostre `solution.cc` no pot contenir la implementació d'altres operacions de la classe. És l'unic fitxer que cal lliurar.

```
// Poseu aqui el vostre nom d'usuari

#include "Poblacion.hh"
#include "Par_crom.hh"

bool Poblacion::poden_ser_parella(const string & s1, const string & s2) const
{
map<string,info_indiv>::const_iterator it1 = m.find(s1);
map<string,info_indiv>::const_iterator it2 = m.find(s2);
bool b = germans(it1->second, it2->second);

// Podeu treballar amb noms o amb iteradors

// Si feu servir descendent_iterador descomenteu la seguent linia
// if (not b) b = descendent_iterador(it1, it2) or descendent_iterador(it2, it1);

// Si feu servir descendent_nom descomenteu la seguent linia
// if (not b) b = descendent_nom(s1, s2) or descendent_nom(s2, s1);
```

```
return not b;
}


// Si treballeu amb iteradors, implementeu aquest
/*
bool Poblacion::descendent_iterador(map<string,info_indiv>::const_iterator it1,
           map<string,info_indiv>::const_iterator it2) const
{
//Descomenteu aquest i implementeu-lo si el voleu fer servir.
}*/

// Si treballeu amb noms, implementeu aquest
/*
bool Poblacion::descendent_nom(const string & nom1, const string & nom2) const
{
//Descomenteu aquest i implementeu-lo si el voleu fer servir
}
*/


// S'ha d'implementar aquest
// Es recomana fer servir el mètode intercanviar de la classe Par_crom
void Par_crom::creuar() {

}
```

Entrada

Una seqüència d'instruccions seguint el format de l'enunciat de l'examen i del joc de proves públic.

Sortida

El seu resultat seguint el format de l'enunciat de l'examen i del joc de proves públic.

Observació

El Jutge prova el vostre lliurament mitjançant 4 jocs de proves.
Heu de lliurar un fitxer `solucio.cc` amb una implementació eficient de les operacions que es demanen.

Exemple d'entrada 1

```
2

Carles

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Marcela

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

escribir_poblacion

reproduccion_sexual Carles Joana FillImpossible
0 0 5 10 15 -1
1 1 5 10 15 -1
0 0 5 10 15 -1

escribir_genotipo Carles

escribir_genotipo Marcela

reproduccion_sexual Carles Marcela MiniCarles
0 0 5 10 15 -1
```

```
1 1 5 10 15 -1
0 0 5 10 15 -1

escribir_genotipo MiniCarles

reproduccion_sexual Marcela Carles MiniMarcela
0 1 5 10 15 -1
1 0 5 10 15 -1
0 1 5 10 15 -1

escribir_genotipo MiniMarcela

reproduccion_sexual MiniMarcela MiniCarles FillImpossible
0 0 5 10 15 -1
1 1 5 10 15 -1
0 0 5 10 15 -1

reproduccion_sexual Marcela MiniCarles FillImpossible
0 0 5 10 15 -1
1 1 5 10 15 -1
0 0 5 10 15 -1

anadir_individuo Ada

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

reproduccion_sexual Ada Carles SuperFill
1 1 3 13 15 -1
0 0 3 13 15 -1
1 1 3 13 15 -1

escribir_genotipo SuperFill

escribir_poblacion

acabar
```

Exemple de sortida 1

```
escribir_poblacion
  Carles ($,$)
  Marcela ($,$)
reproduccion_sexual Carles Joana FillImpossible
  error
escribir_genotipo Carles
  1.1: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1.2: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  2.1: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  2.2: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  3.1: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  3.2: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
escribir_genotipo Marcela
  1.1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1.2: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  2.1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  2.2: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  3.1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  3.2: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
reproduccion_sexual Carles Marcela MiniCarles
escribir_genotipo MiniCarles
  1.1: 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0
  1.2: 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1
  2.1: 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1
  2.2: 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0
  3.1: 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0
  3.2: 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1
reproduccion_sexual Marcela Carles MiniMarcela
escribir_genotipo MiniMarcela
  1.1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1.2: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  2.1: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  2.2: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  3.1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  3.2: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
reproduccion_sexual MiniMarcela MiniCarles FillImpossib
  estan emparentados
reproduccion_sexual Marcela MiniCarles FillImpossible
  estan emparentados
anadir_individuo Ada
reproduccion_sexual Ada Carles SuperFill
escribir_genotipo SuperFill
  1.1: 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
  1.2: 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1
  2.1: 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1
  2.2: 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
  3.1: 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
  3.2: 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1
escribir_poblacion
  Ada ($,$)
  Carles ($,$)
  Marcela ($,$)
  MiniCarles (Carles,Marcela)
  MiniMarcela (Carles,Marcela)
  SuperFill (Ada,Carles)
acabar
```

Informació del problema

Autoria: PR02

Generació: 2026-01-25T17:26:49.196Z