Grupo 13 - Laboratorio 3

Estudiantes:

- FERNANDO HERMOSO CARA (C40)
- IGNACIO PALLÁS GOZÁLVEZ (C62)

10

Fichero `2020_03_25/C40/7-C06/8014_AC/Prac6.cpp`

```
/*
* ----------------------------------------------------
*                ESTRUCTURAS DE DATOS
* ----------------------------------------------------
* Indica el nombre y apellidos de los componentes del
* grupo:
*
* Nombre 1: Fernando Hermoso Cara
* Nombre 2: Ignacio Palls Gozlvez
* ----------------------------------------------------
*/



#include <iostream>
#include <vector>
#include <fstream>
#include <cassert>
#include <string>



class Date {
public:
  Date(int day, int month, int year) : day(day), month(month), year(year) {  }
  Date() : Date(1, 1, 1900) {  }

  bool operator<(const Date& other) const {
    if (this->year < other.year) {
      return true;
    }
    else if (this->year == other.year) {
      if (this->month < other.month) {
        return true;
      }
      else if (this->month == other.month) {
        if (this->day < other.day) {
          return true;
        }
        else {
          return false;
```

```
        }
      }
      else {
        return false;
      }
    }
    else {
      return false;
    }
  }
  bool operator<=(const Date& other) const {
    if (this->year < other.year) {
      return true;
    }
    else if (this->year == other.year) {
      if (this->month < other.month) {
        return true;
      }
      else if (this->month == other.month) {
        if (this->day <= other.day) {
          return true;
        }
        else {
          return false;
        }
      }
      else {
        return false;
      }
    }
    else {
      return false;
    }
  }

  void read(std::istream& in) {
    char slash1, slash2;
    in >> this->day >> slash1 >> this->month >> slash2 >> this->year;
  }

private:
  int day;
  int month;
  int year;
};

std::istream & operator>>(std::istream &in, Date &f) {
  f.read(in);
  return in;
}
```

```
class FamilyTree {
public:
  FamilyTree() {
    this->root = nullptr;
  }


  ~FamilyTree() {
    while (this->root != nullptr) {
      borrar(this->root);
      this->root = nullptr;
    }
  }



  void read(std::istream& in) {
    Date fecha;
    int n;
    in >> fecha >> n;
    root = new FamilyTreeNode;
    root->date = fecha;
    if (n > 0) {
      this->root->children.resize(n);
      for (int i = 0; i < n; ++i)
        hijos(in, this->root->children[i]);
    }

  }

  // O( nlogn ) la funcion sentido
  bool nonsense()const {
    return !sentido(this->root);
  }



private:
  struct FamilyTreeNode {
    Date date;
    std::vector<FamilyTreeNode*> children;
  };
  FamilyTreeNode * root;

  void borrar(FamilyTreeNode* nodo) {

    for (int i = 0; i < nodo->children.size(); ++i) {
```

O(n) n = nº nodos

```
      borrar(nodo->children[i]);
    }
    delete nodo;

  }

  void hijos(std::istream& in, FamilyTreeNode*& nodo) {
    Date fecha;
    int n;
    in >> fecha >> n;
    nodo = new FamilyTreeNode;
    nodo->date = fecha;
    if (n > 0) {
      nodo->children.resize(n);
      for (int i = 0; i < n; ++i)
        hijos(in, nodo->children.at(i));
    }
  }

  bool sentido(FamilyTreeNode* nodo)const {
    int i = 0;
    bool ok = true;
    // comprobamos si los hijos son mayores que el padre
    while (i < nodo->children.size() && ok) {
      if (nodo->children[i]->date <= nodo->date) {
        ok = false;
      }
      ++i;
    }
    i = 0;
    // recursion
    while (i < nodo->children.size() && ok) {
      ok = sentido(nodo->children[i]);
      ++i;
    }
    return ok;
  }
};

std::istream & operator>>(std::istream &in, FamilyTree &f) {
  f.read(in);
  return in;
}


using namespace std;

void tratar_caso() {
  FamilyTree arbol;
  arbol.read(cin);
  if (arbol.nonsense()) {
```

```
    cout << "SI" << endl;
  }
  else
    cout << "NO" << endl;
}


int main() {/*
  #ifndef DOMJUDGE
  std::ifstream in("sample.in");
  auto cinbuf = std::cin.rdbuf(in.rdbuf());
  #endif*/

  int num_casos;
  cin >> num_casos;
  for (int i = 0; i < num_casos; i++) {
    tratar_caso();
  }
  /*
  #ifndef DOMJUDGE
  std::cin.rdbuf(cinbuf);
  // Descomentar si se trabaja en Windows
   system("PAUSE");
  #endif*/

  return 0;
}
```