```
//  File:  tuples.h
//
//Massachusetts Institute of Technology
//16.412J/6.834J Cognitive Robotics
//
//Russian Doll Search
//
//Problem Set #2
//Due: in class Wed, 3/9/05
//
//Lawrence Bush, Brian Bairstow
//{ bush12, bairstow }@mit.edu
// ________________________________________
//
//  tuples.h -  Contains a set of tuples and algorithm pertaining to them.
//              Stores a vector of tuples.
//

#ifndef _tuples_h_
#define _tuples_h_
#include <string>
#include <vector>
#include <algorithm>
#include "variable.h"
using namespace std;

// tuples class
//
//
class tuples {


public:
    tuples() {}  // default constructor
    // Constructor - assigns all variable attributes
    tuples(int constraint_arity_in, int default_cost_in, int number_of_tuples_in, vector
    <int> cX_indecies_in)
        : constraint_arity(constraint_arity_in), default_cost(default_cost_in),
    number_of_tuples(number_of_tuples_in), cX_indecies(cX_indecies_in)
    {
    }

    void insert(tuple t_in){ // insert a variable object into the container
        tuple_vector.push_back(t_in);
    }

    //remove a tuple from the container
    void remove(){
        tuple_vector.pop_back();
    }

    int get_constraint_arity()  const {return constraint_arity;}
    double get_default_cost()   const {return default_cost;}
    int get_number_of_tuples()  const {return number_of_tuples;}
    vector<int> get_cX_indecies()   const {return cX_indecies;}

    // assessor operator, returns player k
    tuple operator[](int k) const
    {
        return tuple_vector[k];
    }


    int size() { return tuple_vector.size(); } // returns the number of tuple in the
    container

    // print tuple container
```

```
    void print(std::ostream & out) {
        out << "-------------------------------------------------------"<<endl;
        out << "-----                Next Tuple Set              -----"<<endl;
        out << "-------------------------------------------------------"<<endl;
        out << "Tuple Set Arity:          "<<constraint_arity<<endl;
        out << "Tuple Set Variables:      ";
        for(vector<int>::iterator iter = cX_indecies.begin();iter != cX_indecies.end();
    iter++){ out << *iter << "  ";}
        out << endl;
        out << "Tuple Set Default Value: "<<default_cost<<endl;
        out << "Tuple Set Size:           "<<number_of_tuples<<endl;

        out << "-------------------------------------------------------"<<endl;

        out << "Print each Tuple in this Tuple Set: \n";
        out << "-------------------------------------------------------"<<endl;
        for( int i = 0 ; i < size() ; i++ ) {
            tuple_vector[i].print(out);
            out << "-------------------------------------------------------"<<endl;
        }
    }

    bool constraintdefined(variables & ca_in)
        //This returns true if all the variables necessary for the constraint have been
    defined.
    {
        for(int i = 0; i < constraint_arity; i++)
        {
            if(!ca_in.isIn(cX_indecies[i]))
                return false;
        }
        return true;
    }


    double evaluate(variables & ca_in, vector<unsigned long long int> & operations)
    {
        double temp;

        if(constraintdefined(ca_in)){


            for( int i = 0 ; i < size() ; i++ )//Size is the number of tuple objects.
            {

                // 0 is tuple    count
                // 1 is variable count
                // 3 is nodes visited
                operations[0] += 1;// for each tuple evaluated
                temp = tuple_vector[i].evaluate(ca_in, operations);
                //operations[1] += tuple_vector[i].size();// count each variable in each
    tuple set evaluated
                if(temp != -1)
                {
                    return temp;
                }
            }
            return default_cost;
        }
        return 0;   //This must be the neutral value
    }

private:
    int constraint_arity;
    double default_cost;
    int number_of_tuples;
    vector<tuple> tuple_vector;
```

```
    vector<int> cX_indecies;
};

#endif
```