```
//  File:  variables.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
//{ bushl2, bairstow }@mit.edu
// __________________________________________
//

//  variables.h -   Contains a variables class.
//          Stores a vector of variables read in from file.
//          Performs functions on the vector.
//          This is a wrapper for a vector.
//          The vector stores the list of all of the variables to pick from.
//

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


class variables {


public:
    variables() {}  // default constructor

    void insert(variable v){ // insert a variable object into the container
        variable_list.push_back(v);
    }

    // remove a variable object from the container
    void remove(){
        variable_list.pop_back();
    }

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


    int size() { return variable_list.size(); } // returns the number of variables in the ↙
    container

    vector<variable> get_variable_list() { return variable_list; } // returns the list of ↙
    variables

    // print container
    void print(std::ostream & out) {

        out << "------------------------------------------------------"<< endl;
        out << "Print all Variables: \n";
        for( int i = 0 ; i < size() ; i++ ) {
            out  <<"Variable Index: " << variable_list[i].get_var_index() << ", Domain 
    Size: " << variable_list[i].get_domain_size() << ", Domain Value: " << variable_list 
    [i].get_domain_value() <<  endl;
        }
    }


    bool isIn(int & index)
    {
        for(int i = 0; i < size(); i++)
        {
            if(index == variable_list[i].get_var_index())
                return true;
        }
        return false;
    }

    bool isMatched(variable & a)
    {
        for(vector<variable>::iterator i = variable_list.begin(); i != variable_list.end()
    ; i++)
        {
            if(a.get_var_index() == i->get_var_index() && a.get_domain_value() == i-> 
    get_domain_value())
                return true;
        }

        return false;
    }


    bool matches(variables & ca_in, vector<unsigned long long int> & operations)
        //This returns true if the variable assignments (ca_in) contains the tuple 
    assignments.(c_vars)
    {
        for(vector<variable>::iterator i = variable_list.begin(); i != variable_list.end()
    ; i++)
        {
            // 0 is tuple    count
            // 1 is variable count
            // 2 is nodes visited
            operations[1]+=1;
            if(!ca_in.isMatched(*i)) {

                return false;
            }
        }

        return true;
    }


    bool empty() { return variable_list.empty(); }

    variable back() {
        return variable_list.back();
    }


private:
    vector<variable> variable_list; // private vector data member

};
```

```
#endif
```