CSCE 876 **FALL 2017**

Venkata Krishna Mohan Sunkara

2 Missionaries & Cannibals puzzle :

The program is written in Java Programming language and has been coded using the Eclipse IDE. To compile and run the program open the file in Eclipse IDE and click on the Run button in the menu of Eclipse. The inputs to the program are hardcoded as initial state as [3,3,0,0,0] and goal state as [0,0,3,3,3]. The general representation of a state can be understood as [Missionaries on left, Cannibals on left, Missionaries on right bank, Cannibals on right bank, Position of boat]

In the main function, we call the getpath function with the initial state and goal state as arguments to the function.

We define a static function called safe() which takes a state array as argument and check whether the number of missionaries are less than the number of cannibals on either side of the river and also checks the condition where number of missionaries may be zero. If the state is safe then it returns true, otherwise it returns false.

The function contains() takes an arraylist consisting of already visited states and a state as arguments. It checks whether the state is present in the arraylist and returns true if it is already present in the arraylist. Otherwise, it returns false.

The function copy() makes a copy of the state passed as argument as returns the newly created copied state.

Now we define several functions with each possible action as a function, which take an arraylist consisting of already visited states and the current state. These functions copy the current state(cs) as modified state (ms) and then performs the action on the modified state. Now it calls the safe() function to check whether the modified state is safe or not. It also checks whether this modified state is already present in the list of visited states by calling the contains() function and also checks that the modified state is in acceptable ranges, if not it assigns all the values in modified state to zero.

If all the above conditions in an action function are true then it assigns the values of modifies state(ms) to that of current state (cs) and returns true. If not, the current state is not changed and it returns false.

The getpath function takes the initial and goal states as arguments. It creates a copy of initial state as gs[] and inserts it into the arraylist consisting of all visited states. It now iterates over the possible set of actions until the current state gs[] is equal to the goal state or until the loop breaks as no solution exists. If an action is viable and contributes to the goal state then that state is added to the visited states arraylist and the action taken is displayed. After exiting the iteration structure, this function displays all the visited states in arraylist.