National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	Artificial Intelligence	Course Code:	CS 401
Program:	BS(CS)	Semester:	Fall 2017
Duration:	1 hr	Total Points:	20
Paper Date:	Monday, 19 Sep 2017	Weight	10%
Section:	ALL	Page(s):	6
Exam Type:	Mid-I		

Student : Name:_______________________________ Roll No.________________ Section:_______

Instruction/Notes: Please Solve all questions on the question paper and also attach all rough sheets at the end.

Problem 1. Solving by Search

Classically the 15-Puzzle has been used to measure the performance of intelligent algorithms that solve a problem by searching for the optimal solution. This problem can be described as follows

The 15-puzzle is a sliding puzzle that consists of a 4 x 4 frame containing fifteen numbered square tiles, numbered from 1 to 15 and a missing tile as shown

The object of the puzzle is to place the tiles in order by making sliding moves that use the empty space.

Part a) Search Algorithms **[1 + 1 + 1 + 1 Points]**

Assuming that the **graph search version** of the blind search algorithms have been implemented and that on average 24 steps are needed to solve a puzzle.

What is the maximum and Minimum number of nodes/states explored by each of the following blind search algorithms on average? Give a brief description of your answer as well

Algorithm	Maximum	Minimum
DFS		
BFS		

Uniform Cost Search		
Iterative Deepening		

Part b) Search Algorithms **[3 + 1 Points]**

For the following **initial** and **goal** states, what is the maximum number of states/nodes expanded by A* and Greedy Best First Search if the **number of tiles out of place** is used as our heuristic value?

Initial State

1	2	3	4
5		7	8
10	6	11	12
9	13	14	15

Goal State

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Part c) Search Algorithms **[2 Points]**

Also compute the maximum number of nodes expanded by BFS and DFS for this search problem.

Problem 2. [Short Questions] [1 + 1 + 1 + 2 Points]

Provide short answers (1-3 sentences) for each of the following questions.

a) In what way is iterative deepening is better than depth-first search? Use the four criteria typically used to compare the search algorithms.

b) Under what minimal conditions on the heuristic function the A* algorithm with graph-search is guaranteed to return an optimal solution?

c) State two major difference between Hill-climbing search and Best-first search.

d) What algorithm would result as a special case if

    i. Local beam search is applied with k = 1.

    ii. Genetic algorithm with population size N = 1 and a mutation rate of 1. (i.e. always apply mutation)

Problem 3: Game Playing: Four In A Row

Four in a Row is a two-player connection game in which the players first choose a color and then take turns dropping colored discs from the top into a seven-column, six-row vertically suspended grid. (As shown in figure 1a) The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs before your opponent.

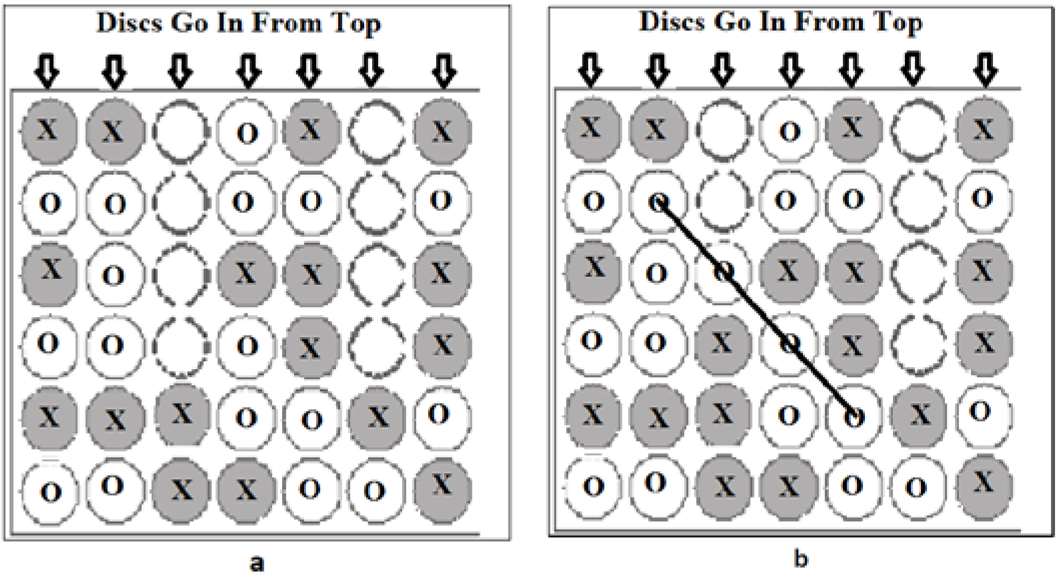

Figure: (a) Four in a Row Game grid. Discs can only be added from top. For sake of paper we represent two different colors by X and O (b) Grey Win state for, as four grey discs are on diagonal

The rules for Four in a Row are simple.

- The field (board) has seven columns and six rows.
- Two players play by alternately dropping a chip down one of the columns (from top).
- The chip drops to the lowest unoccupied spot in that column.
- The first player to get four of his own chips in a row, either vertical, horizontal, or diagonal, wins.
- The game ends in a draw if it fills before someone wins.

An AI student has decided to build an automatic player of FOUR IN A ROW using MINIMAX algorithm. Initially he decide to calculate a move at any given point in the game by building a complete game tree.

a) How many nodes will the game tree have when making the first move? (Give an approximate Answer) Note that at each level a player has about seven possible moves **[2 Points]**

The student figured out that the number of nodes in the game tree is large enough to prohibit building a complete game tree therefore he decided to choose a move by looking only **D** level deep in the tree. For this purpose he comes up with the following evaluation/expert function E.

```
int[][] evaluationTable = {{3, 4, 5,  7,  5,  4, 3},
                           {4, 6, 8,  10, 8,  6, 4},
                           {5, 8, 11, 13, 11, 8, 5},
                           {5, 8, 11, 13, 11, 8, 5},
                           {4, 6, 8,  10, 8,  6, 4},
                           {3, 4, 5,  7,  5,  4, 3}};
//This evaluation table is used as follows

int evaluateContent() {
        int utility = 128;
        int sum = 0;
        for (int i = 0; i < rows; i++)
            for (int j = 0; j <columns; j++)
                if (board[i][j] == 'O')
                    sum -= evaluationTable[i][j];
                else if (board[i][j] == 'X')
                    sum += evaluationTable[i][j];
        return utility + sum;
    }
```

The main idea behind this evaluation function is that the numbers in the table indicate the number of four connected positions which include that space. This gives a measurement of how useful each square is for winning the game and hence it helps decide the strategy. The student implemented the MINIMAX (with alpha-beta pruning) algorithm using his evaluation function. For the state of game given in figure 2**.**

b) It is X's turn to make a move show which move will be selected by the MINIMAX if D = 1

[3 Points]

BONUS) It is X's turn to make a move show which move will be selected by the MINIMAX if D = 2 [3 Points]