CIS 32 **Spring 2007** **Jansen**
MIDTERM

Print Your Name: **KEY**

Chipp Honor Code
"I have neither given nor received aid on this exam." _____ your signature

Problems	**Points**
1 – 12	/ 12
13	/ 7
14	/ 3
15	/ 3
16	/ 3
17	/ 10
18	/ 2
19	/ 8
20	/ 18
21	/ 16
22	/ 12
23	/ 12
TOTAL:	/ 106

You may use one (1) 8.5" by 11" sheet of notes on this exam. Good Luck!

Multiple Choice 1-12 (1 point each)

CLEARLY CIRCLE ONLY ONE CHOICE FOR EACH PROBLEM.

1. What is the **Turing test**?
(a) a test for NP-completeness
(b) a test where two agents try to optimize their goals
(c) a test where an agent and a human answer questions
(d) a test where weekly randomly chosen numbers have a chance to win millions of dollars

2. A perfectly rational backgammon agent never loses.
(a) true
(b) false

3. Weak AI is described as?
(a) the computer is not merely a tool in the study of the mind; rather, the programmed computer really is a mind
(b) the computer accomplishes specific problem solving or reasoning tasks that do not encompass the full range of human cognitive abilities
(c) the computer becomes sapient, (or self-aware), but may or may not exhibit human-like thought processes
(d) all of the above

4. What is a *teleo-reactive* (**TR**) program?
(a) a hierarchical production system
(b) a perceptron
(c) a subsumption system
(d) a deliberative control mechanism

5. What is a *heuristic* function?
(a) in A* search, it is the **g** component of the f = g + h function
(b) an estimate of the path cost from the start to the current node in a search tree
(c) an estimate of the distance from the current node to the goal
(d) the function that gives the output of a TLU based on the sum of the products of the inputs and their weights

6. Which technique constrains the number of levels of nodes expanded in a search tree?
(a) bi-directional search
(b) A* search
(c) depth-limited search
(d) min-max search

7. Depth-first search always expands at least as many nodes as A* search with an admissible heuristic.

(a) true
(b) false

8. What is *supervised learning*?

(a) when the right answer is already in the search tree
(b) when the learner has a training set with which to learn from
(c) when the function being learned has an acceptable state
(d) when the knowledge representation is real-valued

9. Which type of search expands to the cheapest node next (i.e., the one that will cost the least to expand to, from the current node)?

(a) iterative deepening search
(b) best-first search
(c) uniform cost search
(d) greedy searchs

10. What is a fitness function?

(a) in genetic algorithms, the function that computes how good a solution is
(b) in genetic programming, the function that computes how good a solution is
(c) both of the above
(d) none of the above

11. Which of the following supervised training techniques of a TLU (a Threshold Logic Unit used in Neural Networks) does **NOT** use *gradient descent* to minimize the error generated by the unit?

(a) Error Correction Method
(b) Widroff-Hoff Method
(c) Generalized-Delta
(d) none of the above.

12. In Value-Iteration (i.e. a Reinforcement Learning technique), the *discounting factor* in the Value function of a particular state, given a certain policy:

(a) controls how fast the agent learns the new heuristic function for this state
(b) controls how large to update the weights of the TLU given the size of error to a training set example.
(c) controls how much of long-term expected reward is added to the Value of this state
(d) controls how much of an immediate reward given the policy is added to the Value of this state

13. (7 points) Match these people with the contributions to AI that we discussed in class:

Herb Simon	• **A**	• **E.** invents LISP; proponent of logic and representation
Claude Shannon	• **B**	• **G.** anti-logic, just make programs work; microworlds
Ed Feigenbaum	• **C**	• **F.** Subsumption system, cognitive robotics
Warren McCulloch	• **D**	• **D.** Worked with Pitts on the first artificial Neural Network
John McCarthy	• **E.**	• **A.** Worked with Newell on the general problem solver (GP
Rodney Brooks	• **F.**	• **C.** Knowledge principle, rise of expert systems
Marvin Minsky	• **G.**	• **B.** Wrote first chess playing program

14. (1 point) Which (there is only one) of these kinds of functions are not possible for a TLU to learn?

(a) XOR
(b) AND
(c) NOT
(d) OR

14a. (2 points) Why is it not possible for a TLU to learn this function?

XOR is not linearly seperable

15. (3 points) When is a search *heuristic function* ***admissible***? (In other words, define ***admissible*** in terms of heuristic functions)

admissible heiristic functions are optimistic – they never over-estimate the "distance" to the goal.

16. (3 points) Describe how one would account for the use of **dice** (i.e. an element of chance) in a min-max game (i.e. in an adversarial search problem)?
Draw how a min-max tree accounting for the use of a 6-sided die would look like.

One would add a 3rd player (called DICE) who would play before every move of the MIN or MAX player.

A 6-sided die would always have 6 children to take into account (i.e. the 6 results of a roll).

DICE would go first with 6 children.
Then MAX would go with it’s move.
DICE would go again with 6 children
Then MIN would go.

17. (2 points each) Explain the difference between each of the following agent environment characteristics:
(**Examples help!**)

a. Fully Observable vs Not Fully Observable (Accessible vs Inaccessible)

An fully observable environment is one in which the agent can obtain complete, accurate, up-to-date (relevant) information about the environment's state.

b. Deterministic vs Stochiastic (i.e. Non-Deterministic)

A deterministic environment is one in which any action has a single guaranteed effect — there is no uncertainty about the state that will result from performing an action.

c. Episodic vs Sequential (Non-episodic)

In an episodic environment, the performance of an agent is dependent on a number of discrete episodes, with no link between the performance of an agent in different scenarios.

d. Static vs. Dynamic

A static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent.
A dynamic environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control. The physical world is a highly dynamic environment.

e. Discrete vs. Continuous

An environment is discrete if there are a fixed, finite number of actions and percepts in it.

18. (2 points) The following pseudo-code represents the algorithm for what?

A* search

```
agenda = initial state;
```

```
while agenda not empty do {
    take node from agenda such that
        f(node) = min { f(n) | n in agenda }
        where f(n) = g(n) + h(n);
    new nodes = apply operations to node;
    if goal state in new nodes then {
        return solution;
    }
    else add new nodes to agenda
}
```

19. Given an agent (@) whose sensors can detect lines in each of the following two directions:

… and whose motors can move it in one of the four directions: *north*, *south*, *east*, or *west*.

a. (2 points) Describe a mapping for the set *S* of the possible sensor readings.

S: { right-line, left-line }

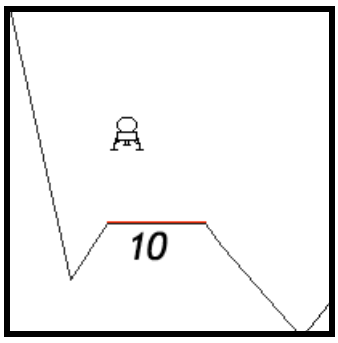
b. (2 points) Describe a mapping for the set *A* of possible actions.

A: { go-north, go-south, go-east, go-west }

c. (4 points) Define a production system for the robot to perform *line following,* describing the function *f : S -> AA* from the set of sensor readings (which you have described above) to a recommended pair of actions for each sensor reading.

1: right-line AND NOT left-line -> go-north, go-east
2: NOT right-line AND left-line -> go-north, go-west
3: right-line AND left-line -> go-north, go-east
4: nil -> north, north

Lines 1 and 2 follow the lines.
3 resolves the conflict of two lines, and it prefers going North East
By default the agent moves north.



20. Consider our Lunar Lander Agent, which we have been working on in Project 1.

Remember, its set of Actions are *thrust, rotate-left*, *rotate-right*, and *do-nothing.*

a. (4 points) Describe how one would **represent** the Lunar Lander for use as a genetic algorithm or a genetic program.

From the paper for the Lunar Lander Game.

Action : Duration Pairs.

Actions:

Turn-left
Turn-right
Do-nothing
Thrust

b. (4 points) Describe the fitness function that would use to rate different Genetic Lunar Landers in Tournament Selection. (Remember a fitness function is not necessary one discrete function, but rather a process to establish a concrete discrete fitness as a number rating)

A fitness function can be a combination of the following factors:
1. Time that the Lunar Lander is in the air
2. Velocity that the Lunar Lander is at when it lands
3. Number of moves that the Lunar Lander does
4. Being in an upright position once landing.

c. (3 points) Describe how your Lunar Landers will reproduce.

Bisection of the lunar lander gene – and swap between parents

d. (3 points) Describe how one would mutate your Lunar Lander.

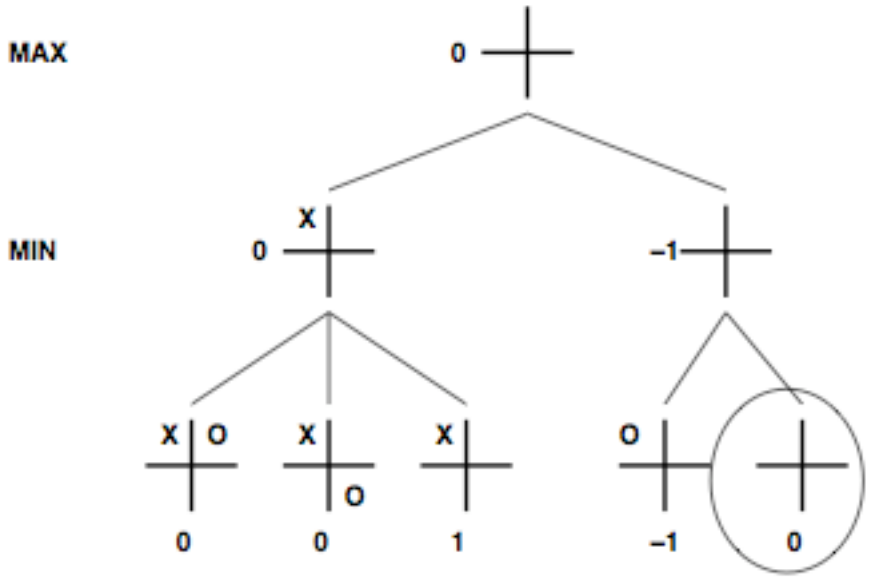
Flipping Actions or Randomly changing Duration.

e. (4 points) Describe how a Tournament Selection process would go to evolve a successful Lunar Lander Agent. (i.e. give the percentages of different populations of Lunar Landers that survive, are replaced through reproduction, are mutated).

Select top 10% of the Batch.
Re-create the 90% of the Batch
Pick less than 1% to replace.
Mutate ~1% people

21. Consider the game of 2 × 2 tictactoe (also known as crosses) where each player has the additional option of *passing* (i.e., marking no square). Assume X goes first.

a. (6 points) Draw the full game tree down to depth 2 (i.e. one move for Player Max, and one move for Player Min). You need not show nodes that are rotations or reflections of siblings already shown. Assume that our agent is unable to recognize repeated states. (*Your tree should have five leaves*.)



b. (3 points) Suppose the evaluation function is the number of Xs minus the number of Os. Mark the values of all leaves and internal nodes of the game tree.

c. (3 points) Circle any node that would not be evaluated by alpha–beta during a left-to-right exploration of your tree.

d. (2 points) Suppose we wanted to *solve* the game to find the optimal move (i.e., no depth limit). Explain why alpha–beta with an appropriate move ordering would be *much* better than min-max search.

Minimax will loop forever. Because alpha-beta, with the right move ordering, prunes the no-move node as soon as it finds a sure win for X, it avoids the loop.

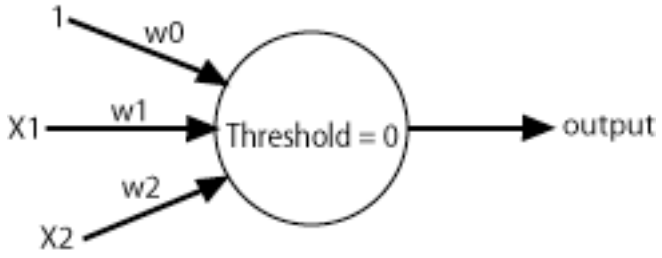
e. (2 points) Briefly discuss how one might modify min-max so that it can solve the really exciting game of 2 × 2 tictactoe (with the ability to pass on one's turn), in which the first player to complete 2-in-a-row **loses**. Describe optimal play for this game. [*Hint*: which is better —a move that definitely loses or a move whose value is unknown?]

T*he no-move solution is optimal for both players. Minimax cannot return this is a solution because it requires going into the infinite loop! We can avoid the infinite loop in minimax by recognizing that the current node is identical to an earlier node. But we need a way to give it a "value" so we can choose a move. We could assign 0 for "draw", but this is not right in cases where the game is winnable by one player or another from the repeated position. Instead, we can assign "?" and use the fact that a win is better than or equal to "?" which is better than or equal to a loss. This can be encoded directly into the inequality tests in the Min-Value and Max-Value functions.*

22. TLUs. Given the following truth table:

X1		X2		d (Output)
0	AND NOT	0	=	0
1	AND NOT	0	=	1
0	AND NOT	1	=	0
1	AND NOT	1	=	0

And the following perceptron:



a. (6 points) What are acceptable values for `w0`, `w1`, and `w2`?

W0 = -0.5
W1 = 1
W2 = -0.5

X1	X2	S	D
0	0	(-0.5)(1) + (1)(0) + (-0.5)(0) = -0.5	0
1	0	(-0.5)(1) + (1)(1) + (-0.5)(0) = 0.5	1
0	1	(-0.5)(1) + (1)(0) + (-0.5)(0) = -0.5	0
1	1	(-0.5)(1) + (1)(1) + (-0.5)(1) = 0	0

b. (6 points) Below we are in the middle of training a TLU for the given function above using the **Error Correction Method**. Fill in the values for `w0`, `w1`, and `w2` for the next 4 training set examples (already given in the table as columns `X1`, `X2`, and `d`).

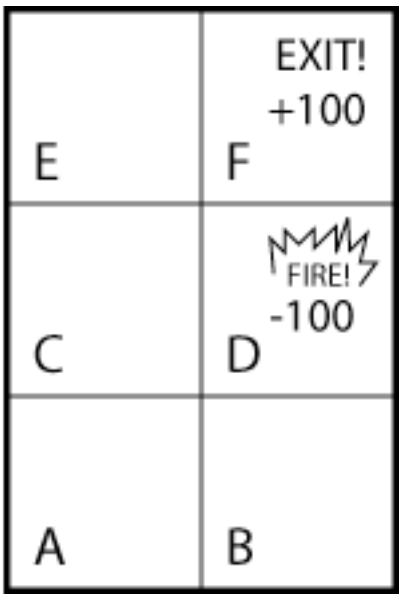
The *learning rate* for this training method is set to 0.1

The other columns are to aid you in your training (if you went over the Neural Network Training Example Excell Spreadsheet).

w0	w1	w2	X1	X2	d	f	d-f	dw0	dw1	dw2
0.1	0.3	0.2	1	1	0	1	-1	-0.1	-0.1	-0.1
0	**0.2**	**0.1**	0	0	0	**0**	**0**	**0**	**0**	**0**

0	**0.2**	**0.1**	0	1	0	**1**	**-1**	**0**	**0**	**-0.1**
-0.1	**0.2**	**0**	1	0	1	**1**	**0**	**0**	**0**	**0**
-0.1	**0.2**	**0**	1	1	0	**1**	**-1**	**-0.1**	**-0.1**	**-0.1**

23. Q-Learning. Consider the room to the right made up of 6 grid squares. Square F is an EXIT! And Square D is ON FIRE!



We are going to train an Agent with **Q-Learning** to navigate around the FIRE and go to the EXIT.

Assume that the States for the Q-Learning are each represented as the 6 Squares in the map of the room on the right.

The Agent can only move *north*, *south*, *east*, and *west* (**no diagonal moves**). Assume that the actions the Agent will consider are: go(X), where X is a Sqaure in the room.

Finally, Square D has a *reward* of -100 because it is on fire, and Square F has a *reward* of +100 because it is the EXIT!

a. (4 points) Fill out the following *Reward Matrix*, for the State-Action pairs. Put a dash '-' for Actions not available in a given State (i.e. Action go(A) in State A is a dash).

	go(A)	**go(B)**	**go(C)**	**go(D)**	**go(E)**	**go(F)**
State A	**-**	**0**	**0**	**-**	**-**	**-**
State B	**0**	**-**		**-100**	**-**	**-**
State C	**0**	**-**	**-**	**-100**	**0**	**-**
State D	**-**	**0**	**0**	**-**	**-**	**100**
State E	**-**	**-**	**0**	**-**	**-**	**100**
State F	**-**	**-**	**-**	**-100**	**0**	**-**

After some training (and some burnt wheels!), the Agent has learned the following Q-matrix.

	go(A)	**go(B)**	**go(C)**	**go(D)**	**go(E)**	**go(F)**
State A	**-**	**41**	**105**	**-**	**-**	**-**
State B	**84**	**-**	**-**	**64**	**-**	**-**
State C	**84**	**-**	**-**	**64**	**164**	**-**
State D	**-**	**41**	**105**	**-**	**-**	**205**
State E	**-**	**-**	**105**	**-**	**-**	**205**
State F	**-**	**-**	**-**	**64**	**164**	**-**

b. (4 points) Given the learned Q-matrix, and the Agent starts in **Square B**. Show the next 4-actions and states if the Agent uses the Q-matrix in choosing the next action.

Action	***State***
Start	*B*
1. Go-A	*A*
2. Go-C	*C*
3. Go-E	*E*
4. Go-F	*F*

c. (2 points) Again, given the learned Q-matrix, and the Agent start in **Square F**. Using the Q-matrix, what is the next action and resulting state for the Agent? How can you chance the Reward Matrix, such that the Agent stays in **Square F** once it reaches **Square F**?

go-E

Change (State F, go(F)) to a positive reward.

d. (2 points) Finally, assume that the Agent will continue to update the values of the learned Q-matrix given on the previous page. If the Agent starts in **Square C**, and randomly chooses to take action **go(E),** what will the resulting Q-matrix values be (assume a *learning rate* of 0.8)?

	go(A)	**go(B)**	**go(C)**	**go(D)**	**go(E)**	**go(F)**
State A	**-**	**41**	**105**	**-**	**-**	**-**
State B	**84**	**-**	**-**	**64**	**-**	**-**
State C	**84**	**-**	**-**	**64**	**~160**	**-**
State D	**-**	**41**	**105**	**-**	**-**	**205**
State E	**-**	**-**	**105**	**-**	**-**	**205**
State F	**-**	**-**	**-**	**64**	**164**	**-**

Updating the square (State C, go(E))

Q(State C, go(E)) = Reward(State C, go(E)) + 0.8 * [Max-Q(State E)]
= 0 + 0.8 * Max{ 105, 205 } ~= 160