Deferred Final Examination — January 17, 2019

Duration: 180 minutes
No aids allowed.
Total marks: 135.

Name:

Student Number:

1)	/15
2)	/10
3)	/8
4)	/4
5)	/8
6)	/10
7)	/20
8)	/20
9)	/10
10)	/10
11)	/10
12)	/10
Total	/135

1. [15 points] For each of the following statements, indicate whether it is *true* or *false*:

   (a) A chess playing agent is operating in a stochastic environment.

   (b) A checkers playing agent is operating in a fully observable environment.

   (c) A Horn clause has at most one negative literal.

   (d) We can make the closed-world assumption about a 2-place predicate/relation r if we have complete information about all the pairs (a, b) where $r(a, b)$ is true.

   (e) Resolution is refutation complete, i.e., if the empty clause is entailed by a set of formulas, it can be derived by resolution.

   (f) Iterative deepening depth-first search has a lower space complexity than breadth-first search.

   (g) A constraint satisfaction problem that has been made arc-consistent can always be solved without backtracking.

   (h) If a heuristic is monotone/consistent, it must be admissible.

   (i) If q_1 is conditionally independent from q_2, then q_2 is conditionally independent from q_1.

   (j) When A^* uses a heuristic that is only admissible, it will nonetheless always find an optimal path to a goal state (assuming a goal state is reachable).

   (k) A solution to a Markov decision problem is a policy, i.e., a function from state to action.

   (l) The policy iteration algorithm can be used to solve a Markov decision problem.

   (m) In Q-learning, one learns the function $Q(a, s)$, whose value is the expected utility to taking an action a in a state s.

   (n) A single-layer feedforward neural network (perceptron) can learn any boolean function that decision-tree learning can learn.

   (o) A multi-layer feedforward neural network can learn any continuous function.

2. [10 points] Suppose that we have a search algorithm that keeps unexpanded nodes in a sorted list, the frontier (or fringe), and that always selects the first node from the frontier. What kind of search do we have if...

   (a) we always place the successors of the current node at the back of the frontier?

   (b) we always place the successors of the current node at the front of the frontier?

   (c) we insert the successors of the current node so as to keep the frontier sorted by the value of the heuristic function $h(n)$?

   (d) we insert the successors of the current node so as to keep the frontier sorted by path cost $g(n)$ (i.e. the cost of the path from the start node to node n)?

   (e) we insert the successors of the current node so as to keep the frontier sorted by the value of the sum of the path cost and the heuristic function $g(n) + h(n)$?

3. [8 points] Implement the following description of an *ancestor* relation using one or more Prolog rules:

   if X is a parent of Y, then X is an ancestor of Y;

   if X is a parent of Y and Y is an ancestor of Z, then X is an ancestor of Z;

   nothing else is in the ancestor relation.

   You can assume that the `parentOf(X,Y)` predicate has already been defined.

4. [4 points] What is the output of the following Prolog program, when we invoke the query `?- p.`

```
p:- writeln(p1), q.
p:- writeln(p2), !, q.
p:- writeln(p3), fail.

q:- writeln(q1), !, fail.
q:- writeln(q2), fail.
```

5. [8 points] The predicate `bagof(T, Q, S)` constructs a list `S` of all instances of `T` for which `Q` is true. A textbook describes the computation as follows: the query `Q` is satisfied in all possible ways, and for each instantiation of the variables in the arguments of `Q`, `T` is instantiated and recorded as such in the list `S`.

   (a) Give the value computed for `L` by the query

   ```
   ?- bagof(p(X), member(X, [a, t(0), b, a]), L).
   ```

   (b) The following query will not work:

   ```
   ?- bagof(X, member(0, X), L).
   ```

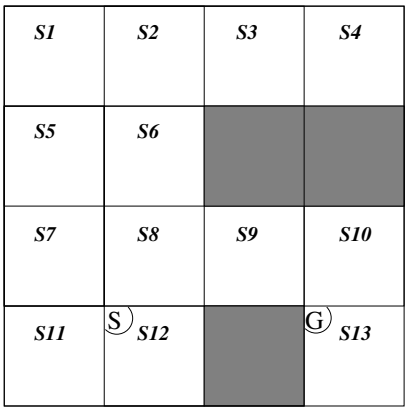
   Explain briefly why.

6. [10 points] Convert the following formulas into clausal form (recall that the 8 steps are: eliminate implications, move negations inward, standardize variables, skolemize, convert to prenix, distribute $\vee$ over $\wedge$, flatten conjunctions and disjunctions, and convert to clauses):

   (a) $(\neg\exists X.\forall Y.p(X,Y)) \rightarrow (\forall Y.\exists X.p(X,Y))$

   (b) $((\exists X.p(X)) \vee (\exists X.q(X))) \rightarrow (\exists X.(p(X) \vee q(X)))$

7. [20 points] Consider the problem of finding a path in a grid using *heuristic search*. The problem is to find a path from square $S = s12$ to square $G = s13$, given that you can move only horizontally and vertically, one square at a time, and no step may be made into a shaded square. Each step *has cost one*.



Suppose that we want to use A* *search with cycle-checking* and the *Manhattan distance* as the heuristic function h to solve this problem. The Manhattan distance between two cells is the sum of the horizontal and vertical distances (counting shaded squares as well). Note that this heuristic is *monotonic*.

(a) Write down the h values of squares $s1$ through $s13$ in the grid below (the values for the start node $s12$ and the goal node $s13$ have already been given):

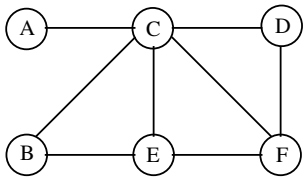
S1 h=	S2 h=	S3 h=	S4 h=
S5 h=	S6 h=		
S7 h=	S8 h=	S9 h=	S10 h=
S11 h=	S S12 h=2		G S13 h=0

(b) Now, apply the A* search: continue writing down the frontier for each step of the algorithm in the format shown below (it must be sorted on f values, and f written as $f = g + h$). In the case of equal f values, break ties as follows: the square with the *larger number* comes first (e.g. $s11$ before $s10$ and $s5$). Underline the node selected for expansion at each step. *Remember to enforce cycle-checking.*

1. $\left\{\underline{\langle s12, 2 = 0 + 2\rangle}\right\}$
2. $\left\{\underline{\langle s11, 4 = 1 + 3\rangle}, \qquad\qquad\qquad\qquad\right\}$
3.

$\vdots$

8. [20 points] Consider the following *constraint satisfaction problem*. We have the graph shown below with 6 nodes A, B, C, D, E, and F. We want to color each node such that no two nodes connected by an edge get the same color. There are only three colors b, g and r (i.e. blue, green, red) available for nodes C, D, E, F. Moreover, node A must be colored red and node B must be colored blue.

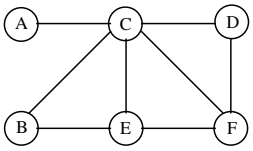


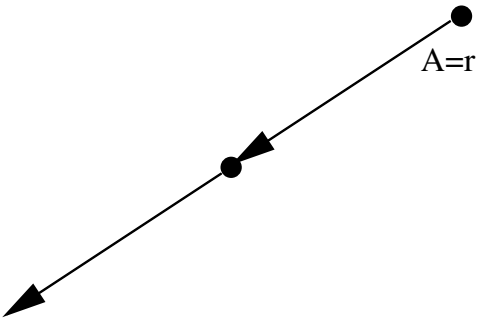
(a) Write down the domain of each of the 6 variables (do not apply forward checking yet).

$Dom(A) = \{\quad\}$, $Dom(B) = \{\quad\}$, $Dom(C) = \{\quad\}$,
$Dom(D) = \{\quad\}$, $Dom(E) = \{\quad\}$, $Dom(F) = \{\quad\}$.

(b) Complete the search tree on the next page to show the tree that would be explored by the *backtracking search with forward checking* algorithm. Remember that in this algorithm, constraints are checked when all but one of their variables have been assigned and values that violate the constraint are removed from the domain of the unassigned variable. You are to use a static variable ordering whereby each branch considers assignments to the variables in the sequence A, B, C, D, E, F. The values are also tried in the sequence b, g, and r when applicable. Annotate each edge with what variable is assigned what value and the updated domains of the remaining variables. Use the symbol DWO to mark whenever a domain wipe-out happens.

Q2b: Complete the search tree. Remember to try values in the order b, g, and then r. The graph on the left is shown for your convenience.





9. [10 points] Consider the following dynamic domain example concerning a room with two windows, $window1$ and $window2$, which is specified in the situation calculus. Suppose that the following are all of the effect axioms about the fluents $isOpen$ and $roomHot$ (describing all conditions under which these fluents change value):

$$\forall S.\forall Window.isOpen(Window, do(open(Window), S))$$
$$\forall S.\forall Window.\neg isOpen(Window, do(close(Window), S))$$
$$\forall S.(\neg isOpen(window2, S) \rightarrow roomHot(do(close(window1), S)))$$
$$\forall S.(\neg isOpen(window1, S) \rightarrow roomHot(do(close(window2), S)))$$
$$\forall S.\neg roomHot(do(open(window1), S))$$
$$\forall S.\neg roomHot(do(open(window2), S))$$

a) Give a frame axiom for the action *open* and the fluent $isOpen$.

b) Give the successor state axiom for the fluent $isOpen$.

c) Give the successor state axiom for the fluent $roomHot$.

10. [10 points] Consider the following STRIPS actions:

Action Name	Preconditions	Add Effects	Delete Effects
A	p	q	r
B	p	r	-
C	-	p	-
D	o	-	o

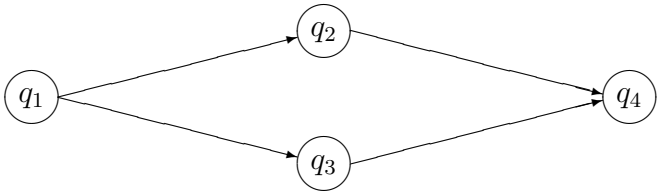
Suppose that the start state is $S_0 = \{o\}$ and the goal is $G = \{p, q, r, o\}$.

(a) Which actions are applicable in the start state S_0? What is the new state for each of the applicable actions?

(b) Suppose we want to do backward/regression planning. Which actions are consistent with the goal G? What is the regressed goal for all the consistent actions?

(c) Give a sequence of actions that achieves the goal G when executed in the starting state S_0.

11. [10 points] Suppose that we have the following belief network (or Bayes net):



(a) True or false, given this belief network, is it the case that $Pr(q_4|q_1, q_2, q_3) = Pr(q_4|q_2, q_3)$?

(b) What are the conditional probabilities that need to be specified to fully determine the joint probability distribution?

(c) Express $Pr(q_2|q_3)$ in terms of the conditional probabilities given in your answer to the previous question.

12. [10 points] Suppose that there are N men and N women who want to get married. Each man has a list of all the women in his preferred order and similarly each woman has a list of all the men in her preferred order. We want to find a set of marriages that is stable.

    A set of marriages is unstable if two people who are not married to each other both prefer each other to their spouses. For example, suppose that we have two men `m1` and `m2` and two women `w1` and `w2`, such that `m1` prefers `w1` to `w2`, and `w1` prefers `m1` to `m2`. Then the set of marriages `m1-w2` and `m2-w1` is unstable because `m1` and `w1` both prefer each other to their spouses.

    (a) Assume that the preferences of a person are represented by asserting a predicate `preferences(Person,[P1, P2, ...., Pn])`, meaning that `[P1, P2, ...., Pn]` is the list of `Person`'s preferred spouses from most preferred `P1` to least preferred `Pn`. Building on this, give a definition of a Prolog predicate `prefers(Pa,Pb,Pc)` that holds if and only if `Pa` prefers `Pb` to `Pc`. If you use any other non-built-in predicates, provide their definitions.

(b) Give a definition of a Prolog predicate `stableMarriages(MenList, WomenList, MarriagesList)` that holds if `MarriagesList` is a set of marriages that is stable for the given sets of men and women. Assume that sets are represented as list of distinct items and that there is an equal number of men and women. A marriage between a man `m` and a woman `w` should be represented by the term `m-w`. You may use the predicate `prefers` defined in (a) in your definition. If you use any other non-built-in predicates, provide their definitions. (It is a theorem of graph theory that it is always possible to find a stable marriage set.)