NAMIBIA UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Faculty of Computing and Informatics

Department of Computer Science

QUALIFICATION: Bachelor of Computer Science	
QUALIFICATION CODE: 07BACS	**LEVEL:** 7
COURSE: Artificial Intelligence	**COURSE CODE:** ARI711S
DATE: JULY 2023	**SESSION:** 2
DURATION: 3 HOURS	**MARKS:** 100

SUPPLEMENTARY/ SECOND OPPORTUNITY EXAMINATION QUESTION PAPER	
EXAMINER	MR STANTIN SIEBRITZ
MODERATOR:	MS PAULINA SHIFUGULA

THIS EXAM PAPER CONSISTS OF 8 PAGES
(Including this front page)

INSTRUCTIONS

1. This paper contains 6 questions.
2. Answer ALL questions on the examination booklet provided.
3. Marks/scores are indicated at the right end of each question.
4. Calculators are permitted.
5. NUST examination rules and regulations apply.

QUESTION 1: SHORT ANSWERS **[20]**

State whether the following statements are either TRUE or FALSE.

1.1 Uniform-cost search will never expand more nodes than A*-search. (2)

1.2 Depth-first search will always expand more nodes than breadth-first search. (2)

1.3 The most-constrained variable heuristic provides a way to select the next variable to assign in a backtracking search for solving a CSP. (2)

1.4 By using the most-constrained variable heuristic and the least-constraining value heuristic we can solve every CSP in time linear in the number of variables. (2)

1.5 When using alpha-beta pruning, it is possible to get an incorrect value at the root node by choosing a bad ordering when expanding children. (2)

1.6 When using alpha-beta pruning, the computational savings are independent of the order in which children are expanded. (2)

1.7 For an MDP (S; A; T; ϒ; R) if we only change the reward function R the optimal policy is guaranteed to remain the same. (2)

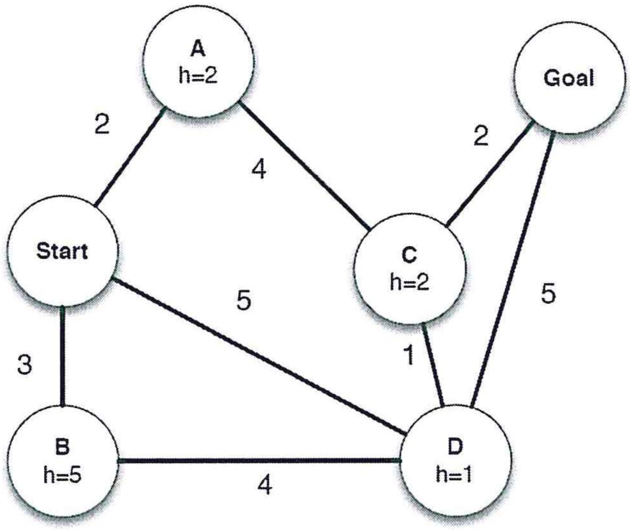
1.8 Policies found by value iteration are superior to policies found by policy iteration. (2)

1.9 Q-learning can learn the optimal Q-function Q* without ever executing the optimal policy. (2)

1.10 If an MDP has a transition model T that assigns non-zero probability for all T(s; a; s0) then Q-learning will fail. (2)

QUESTION 2: SEARCH **[20]**

For each of the following graph search strategies, work out the order in which states are expanded, as well as the path returned by graph search. In all cases, assume ties resolve in such a way that states with earlier alphabetical order are expanded first. The start and goal state are S and G, respectively.



2.1	Depth-First Search	(4)
2.2	Breadth-First Search	(4)
2.3	Uniform Cost Search	(4)
2.4	Greedy search with the heuristic h shown on the graph.	(4)
2.5	A* search with the same heuristic.	(4)

QUESTION 3: CSP **[6]**

You are in charge of scheduling for computer science classes that meet Mondays, Wednesdays and Fridays. There are 5 classes that meet on these days and 3 professors who will be teaching these classes. You are constrained by the fact that each professor can only teach one class at a time.

The classes are:

- Class 1 - Intro to Programming: meets from 8:00-9:00am
- Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:30am
- Class 3 - Natural Language Processing: meets from 9:00-10:00am
- Class 4 - Computer Vision: meets from 9:00-10:00am
- Class 5 - Machine Learning: meets from 10:30-11:30am

The professors are:

- Professor A, who is qualified to teach Classes 1, 2, and 5.
- Professor B, who is qualified to teach Classes 3, 4, and 5.
- Professor C, who is qualified to teach Classes 1, 3, and 4.

3.1 Formulate this problem as a CSP problem in which there is one variable per class, stating the domains, and constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit. (2)

3.2 Draw the constraint graph associated with your CSP. (2)

3.3 Provide a possible solution to the CSP scheduling problem. All variables should be assigned and no constraints broken. (2)

QUESTION 4: CSP **[14]**

After years of struggling through mazes, Pacman has finally made peace with the ghosts, Blinky, Pinky, Inky, and Clyde, and invited them to live with him and Ms. Pacman. The move has forced Pacman to change the rooming assignments in his house, which has 6 rooms. He has decided to figure out the new assignments with a CSP in which the variables are Pacman (**P**), Ms. Pacman (**M**), Blinky (**B**), Pinky (**K**), Inky (**I**), and Clyde (**C**), the values are which room they will stay in, from 1-6, and the constraints are:

i) No two agents can stay in the same room
ii) **P** > 3
iii) **K** is less than **P**
iv) **M** is either 5 or 6
v) **P** > **M**
vi) **B** is even
vii) **I** is not 1 or 6
viii) (**I-C**) = 1
ix) (**P-B**) = 2

4.1 **Unary constraints:** On the grid below cross out the values from each domain that are eliminated by enforcing unary constraints. (4)

P	1	2	3	4	5	6
B	1	2	3	4	5	6
C	1	2	3	4	5	6
K	1	2	3	4	5	6
I	1	2	3	4	5	6
M	1	2	3	4	5	6

4.2 **MRV:** According to the Minimum Remaining Value (MRV) heuristic, which variable should be assigned to first? (2)

4.3 **Forward Checking**: For the purposes of decoupling this problem from your solution to the previous problem, assume we choose to assign P first, and assign it the value 6. What are the resulting domains after enforcing unary constraints (from part 4.1) and running forward checking for this assignment? (4)

P						6
B	1	2	3	4	5	6
C	1	2	3	4	5	6
K	1	2	3	4	5	6
I	1	2	3	4	5	6
M	1	2	3	4	5	6

4.4 **Iterative Improvement:** Instead of running backtracking search, you decide to start over and run iterative improvement with the min-conflicts heuristic for value selection. Starting with the following assignment: P:6, B:4, C:3, K:2, I:1, M:5 (4)

First, for each variable write down how many constraints it violates in the table below. Then, in the table on the right, for all variables that could be selected for assignment, put an [X] in any box that corresponds to a possible value that could be assigned to that variable according to min-conflicts. When marking next values a variable could take on, only mark values different from the current one.

Variable	# violated
P	
B	
C	
K	
I	
M	

	1	2	3	4	5	6
P						
B						
C						
K						
I						
M						

QUESTION 5: MDP **[20]**

The following problems take place in various scenarios of the gridworld MDP (as in Assignment P3). In all cases, A is the start state and double-rectangle states are exit states.

From an exit state, the only action available is *Exit*, which results in the listed reward and ends the game (by moving into a terminal state X, not shown).

From non-exit states, the agent can choose either *Left* or *Right* actions, which move the agent in the corresponding direction. There are no living rewards; the only non-zero rewards come from exiting the grid.

Throughout this problem, assume that value iteration begins with initial values $V_0(s) = 0$ for all states s.

First, consider the following mini-grid. For now, the discount is $\Upsilon = 1$ and legal movement actions will always succeed (and so the state transition function is deterministic).

+1	A			+10

5.1 What is the optimal value $V^*(A)$? (1)

5.2 When running value iteration, remember that we start with $V_0(s) = 0$ for all *s*. What is the first iteration *k* for which $V_k(A)$ will be non-zero? (2)

5.3 What will $V_k(A)$ be when it is first non-zero? (1)

5.4 After how many iterations *k* will we have $V_k(A) = V^*(A)$? If they will never become equal, write *never*. (2)

Now the situation is as before, but the discount is less than 1.

5.5 If $\Upsilon = 0.5$, what is the optimal value $V^*(A)$? (3)

5.6 For what range of values Υ of the discount will it be optimal to go Right from A? Remember that $0 \leq \Upsilon \leq 1$. Write *all* or *none* if all or no legal values of have this property. (4)

Let's kick it up a notch! The *Left* and *Right* movement actions are now *stochastic* and *fail* with probability *f*. When an action fails, the agent moves up or down with probability f/2 each. When there is no square to move *up* or *down* into (as in the one-dimensional case), the agent stays in place. The *Exit* action does not fail.

For the following mini-grid, the failure probability is $f = 0.5$ and the discount is $\Upsilon = 1$

A			+10

5.7 What is the optimal value $V^*(A)$? (1)

5.8 When running value iteration, what is the smallest value of k for which $V_k(A)$ will be non-zero? (1)

5.9 What will $V_k(A)$ be when it is first non-zero? (2)

5.10 After how many iterations *k* will we have $V_k(A) = V^*(A)$? If they will never become equal, write *never*. Motivate your answer. (3)

QUESTION 6: REINFORCEMENT LEARNING **[10]**

Assume we have an *MDP* with state space *S*, action space *A*, reward function *R(s; a; s')*, and discount *ϒ*.

Our eventual goal is to learn a policy that can be used by a robot in the real world. However, we only have access to simulation software, not the robot directly. We know that the simulation software is built using the transition model $T_{sim}(s; a; s')$ which is unfortunately different than the transition model that governs our real robot, $T_{real}(s; a; s')$.

Without changing the simulation software, we want to use the samples drawn from the simulator to learn Q-values for our real robot.

Recall the Q-learning update rule. Given a sample *(s; a; s'; r)*, it performs the following update:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a')\right]$$

6.1 Assuming the samples are drawn from the simulator, which new update rule will learn the correct Q-value functions for the real-world robot? Provide the correct update rule and explain. (6)

6.2 Now consider the case where we have *n* real robots with transition models $T^1_{real}(s; a; s'), \ldots, T^n_{real}(s; a; s')$ and still only one simulator. Is there a way to learn policies for all *n* robots simultaneously by using the same samples from the simulator? If yes, explain how. If no, explain why not. (1-2 sentences) (4)

END OF EXAMINATION PAPER