KOÇ UNIVERSITY
COLLEGE OF ENGINEERING

COMP 341: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Midterm Examination I

FALL 2016, 27/10/2016
DURATION: 100 MINUTES

Name: Solutions

ID: 00110001 00110000 00110000

- This exam contains 7 pages including this cover page and 6 questions. Check to see if any pages are missing. Put your initials on the top of every page, in case the pages become separated.
- By submitting this exam, you **agree** to fully comply with Koç University Student Code of Conduct, and accept any punishment in case of failure to comply. If you do not submit this exam on time, you will receive 0 credits.
- The exam is **closed book** and **closed notes**. You are **not** allowed to use any additional material including any electronic equipment such as computers and mobile phones.
- You are expected to be able provide clear and concise answers. Gibberish will not receive any credit.
- Your answers need to be readable by a human, illegible writing is not gradable hence there is a strong chance that such answers will not get any credit.
- Do not write in the table below

Question:	1	2	3	4	5	6	Total
Points:	18	28	20	15	10	9	100
Score:							

1. (18 points) Answer the following questions:

   (a) (6 points) In 10 words or less:
   What is the main purpose behind a rational agent's decision making (e.g. picking an action) process?
   Maximizing its utility function. I will accept anything that reflects something similar.

   A reflex agent maps its percepts (or its current state) directly to actions. Write 1 advantage and 1 disadvantage of this approach

   - Advantage: Fast computation, easy to code, good enough for certain problems
   - Disadvantage: No look ahead, implicit goals, getting stuck

   I will accept other legitimate answers that are not listed here

   Give one reason that the solutions returned by our algorithms may not be reliable in real life.
   Modelling errors, excessive uncertainty.

   In general, the errors tend to be due to modelling. There are multiple modelling related issues and I will accept anything that makes sense.

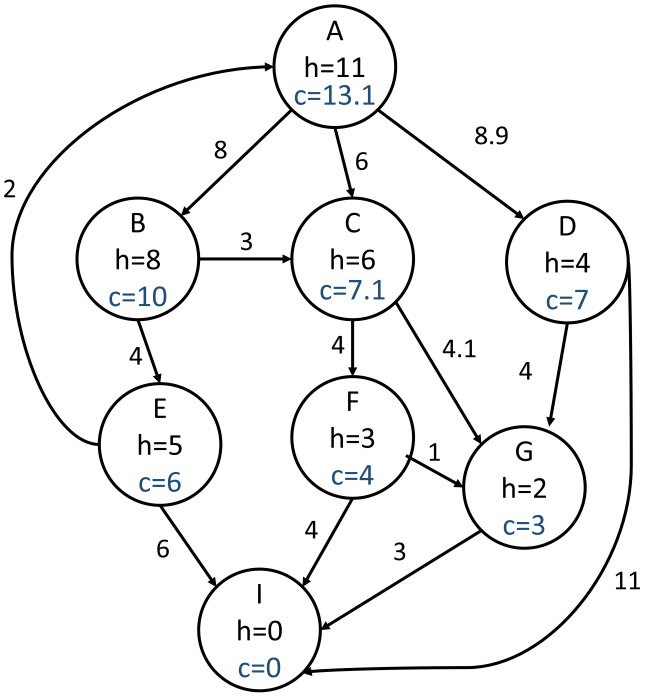
   (b) (12 points) True or False (Depth First Search: DFS, Constraint Satisfaction Problem: CSP):

False	Iterative deepening DFS has higher time complexity than regular DFS
True/False	Uniform cost search is always optimal for non-negative costs (meant to ask positive)
False	A* search is always optimal for non-negative costs
False	Cut-set conditioning can reduce the complexity of an arbitrary CSP
True	Simulated annealing can make a move that decreases the objective function
False	Alpha-Beta pruning runs faster because it sacrifices optimality

   A large cutset would still have a high complexity

2. (28 points) Consider the graph below. **A** is the initial state and **I** is the goal state. The arcs represent the state transitions, which are directional. The cost of state transitions are written on the arcs. The heuristic values of the states are written in their respective circles. For the given algorithms, write the expansion order of the nodes, breaking ties **alphabetically**, and the resulting solution path. Use the **graph search** versions of the algorithms.

   I will give partial credit to consistent orderings



   (a) (5 points) Breadth First Search
   Expanded Nodes:
   A-B-C-D-E-F-G-I

   Resulting Path:
   A-D-I

   (b) (5 points) Depth First Search
   Expanded Nodes:
   A-B-C-F-G-I

   Resulting Path:
   A-B-C-F-G-I

   (c) (5 points) Uniform Cost Search
   Expanded Nodes:
   A-C-B-D-F-G-E-I

   Resulting Path:
   A-C-G-I

   (d) (5 points) Greedy Search
   Expanded Nodes:
   A-D-I

   Resulting Path:
   A-D-I

   (e) (5 points) A* Search
   Expanded Nodes:
   A-C-G-D-F-I

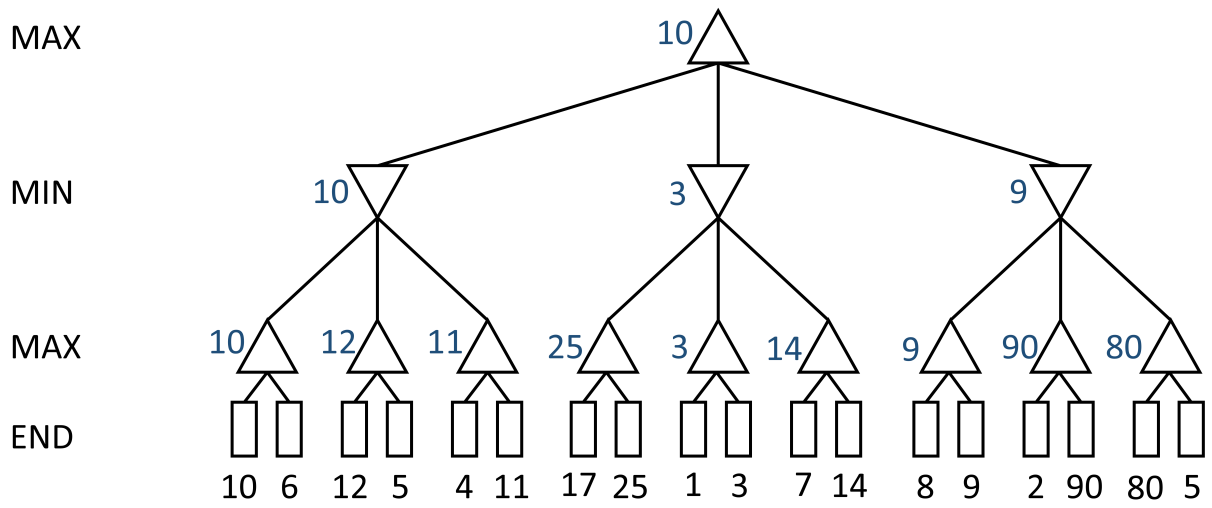
   Resulting Path:
   A-C-G-I

   (f) (3 points) Is this heuristic admissible? Justify your answer by giving the optimal cost for each state.

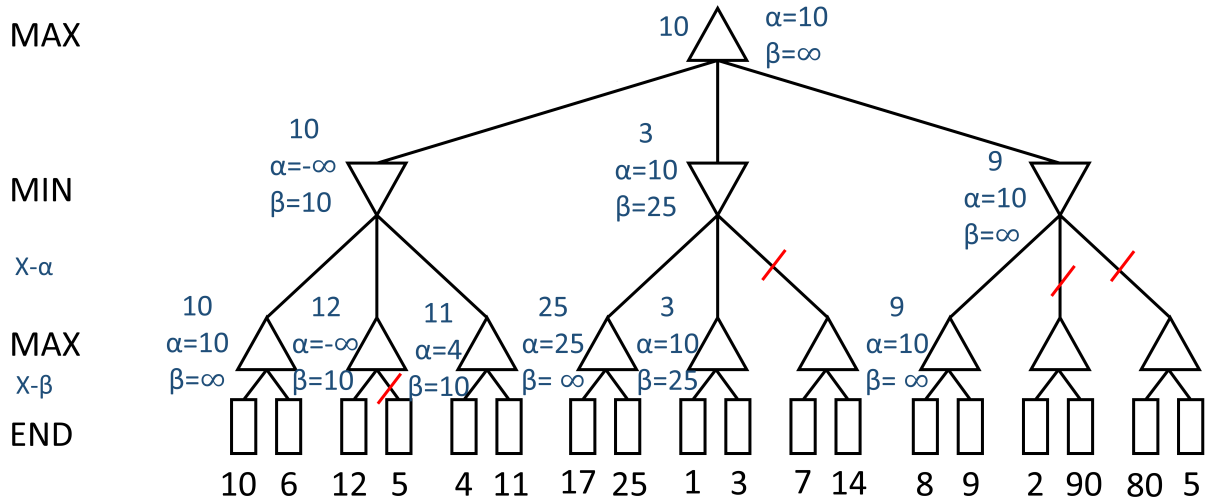
   For all the nodes, $c^*(s) \geq h(s)$, look at the graph

3. (20 points) Adversarial Search:

   (a) (5 points) Fill in the minimax values for the nodes



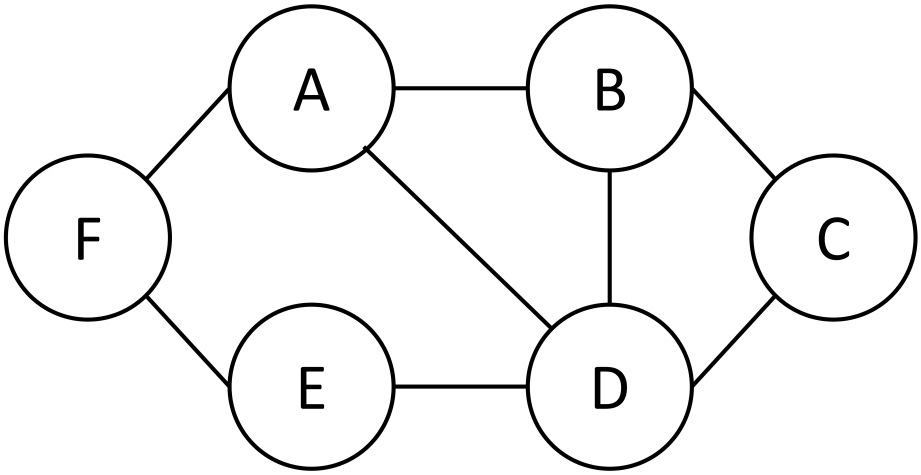
   (b) (15 points) Apply alpha-beta pruning to the tree, assuming a depth-first left-to-right expansion. Write the final $\alpha - \beta$ values next to the nodes, wherever applicable. Mark the pruned paths with an **X**-α or an **X**-β, depending on which one resulted in the pruning.



The root node's β values does not really matter as it will always have β = inf if treated as a max-node. Another alternative is to let it be the best potential value for the root node, given the expanded children, which does not fit the definition but this was used in the slides.
Cuts: 6 (1.5 each), $\alpha - \beta$ values: 7 (around 0.75 each, but will be rounded), Cut-types: 1 point, Misc (root-node, correct action etc.): 1 point

4. (15 points) Consider the following constraint graph where the nodes represent variables and arcs represents the *difference* constraint, i.e, if two variables have an arc between them, they cannot be the same value. The domains of all the variables are initially the same: $\{1, 2, 3\}$. Solve this CSP by using the minimum value heuristic, degree heuristic and forward checking.

   Show all your work in the table provided. Under each variable column, you are going to write the remaining domains of the variables. Circle the value you chose under the corresponding variable, leading up to the next step. Cross out the values you eliminate with forward checking before moving to the next step. Under the MRV and DH, fill in the heuristic values, if you need to. Break all ties **alphabetically** for variables and by **ascending** order for values. Put a dash in the remaining cells of the variable you have assigned a value to. Do not backtrack but stop if you get an empty domain. With this condition, you can at most do 6 assignments, hence the table will be enough. Write the final assignments in the last row, leave any cell blank if its domain was empty before assignment.

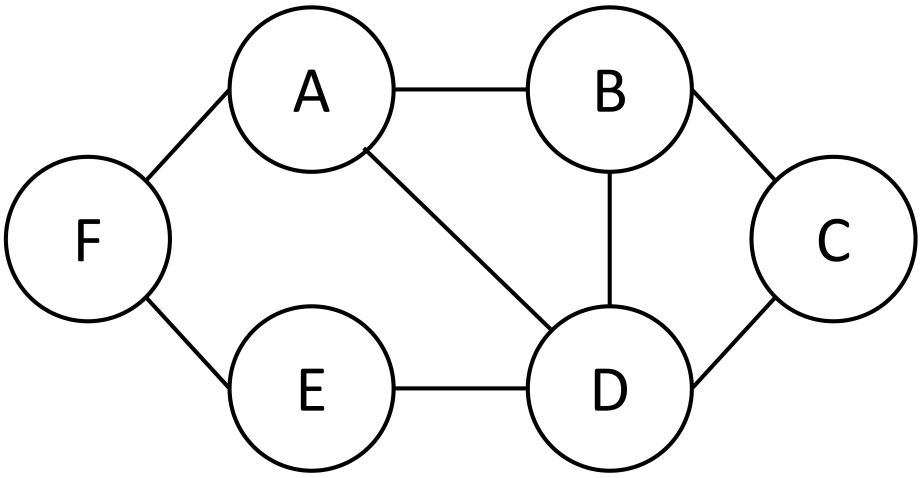


	A	B	C	D	E	F	MRV	DH
Init	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	3	4
Step 1	2,3	2,3	2,3	-	2,3	1,2,3	2	2
Step 2	-	3	2,3	-	2,3	1,3	1	(1)
Step 3	-	-	2	-	2,3	1,3	1	(0)
Step 4	-	-	-	-	2,3	1,3	2	1
Step 5	-	-	-	-	-	1,3	2	(0)
Final	2	3	2	1	2	1	-	-

I will give partial points to other orderings as long as they make sense (e.g. no DH). Note that I did not ask the Least Constraining Value (LCV) heuristic, which is about value selection and not variable selection. Exercise: Would the assignments have changed if we were to use LCV?
It was difficult to formalize the entire grading scheme for this problem. I tried to be as fair as possible in my partial credits. (Meaning if you received partial credit, it is wise not to ask for re-evaluation unless you are absolutely positively definitely sure.)

5. (10 points) Consider the following constraint graph again. This time you are going to solve it using hill-climbing. The objective will be to minimize the number of conflicts and the allowable actions at each step is to change only one variable's assignment at a time. The domains of the variables are again $\{1, 2, 3\}$.

   You are going to fill the table below, given the initial assignment. At each step, write the value of the objective function and circle the assignment you are going to change. Stop if you get to a local minimum. The size of the table is not an indication of the solution length. You can extend it if you need to or finish before filling it up. As before, break all ties **alphabetically** for variables and by **ascending** order for values.



	A	B	C	D	E	F	Obj. Func.
Init	1	1	2	1	2	2	4
Step 1	3	1	2	1	2	2	2
Step 2	3	1	2	1	3	2	1
Step 3	3	2	2	1	3	2	1
Step 4							
Step 5							
Step 6							

The last row is optional. Note that the HC with the given tie breakers would alternate between the last two rows. I will give partial points to other orderings as long as they make sense. Wrong alphabetical starts (B and D) will get -2, others will get -3. Wrong moves will start from -3 and increase. Wrong objective function will get -1 and no objectives will get -2. In other words Correct start:2-3, correct moves 5-6, correct objective: 2

6. (9 points) Your embedded systems project requires you to measure the pressure under a tank to deduce how much water is left in the tank. You are given a pressure sensor and told that its voltage output changes with pressure according to a known function $p = f(v, w)$, where p is the pressure, v is the voltage and w is the function's parameter vector. On the design sheet, the model parameters are given as w_0. As you implement your project, you realize that something is off, the values do not make too much sense. Assuming that the initial values are not too far off, you decide to update the model parameters, w, by using gradient descent. As a first step, you take n measurements of v using the sensor and the corresponding p with known quantities of water.

   What would be the objective function to be minimized? What is the gradient of this function? Write the resulting update equation. When would you stop your algorithm?

   This was a somewhat open ended question. I am providing my expected answer. I will accept any other reasonable answer as well.

   The aim here is to do sensor calibration. We want to have accurate readings of p. Given the measurements we made, we want to update w such that the difference $f(v_i, w) - p_i$ is small. In fact we want to make minimize that difference.

$$e = \frac{1}{2}\sum_{i=1}^{n}(f(v_i, w) - p_i)^2$$

   One reason we have square is to make sure that both the positive and negative errors are penalized. There are multiple reasons why we do not have the absolute value. The interested student is referred to the wikipedia pages on Least Squares and Least Absolute Deviations.

$$\nabla e(w) = \sum_{i=1}^{n}((f(v_i, w) - p_i)\nabla f(v_i, w))$$

   Note that the gradients are vector valued! The resulting update equation, with w_0 as the initial point:

$$w_{t+1} = w_t - \alpha \nabla e(w_t)$$

   where $\alpha > 0$ is the step size parameter. Then you would stop when $|w_{t+1} - w_t| < \epsilon$.