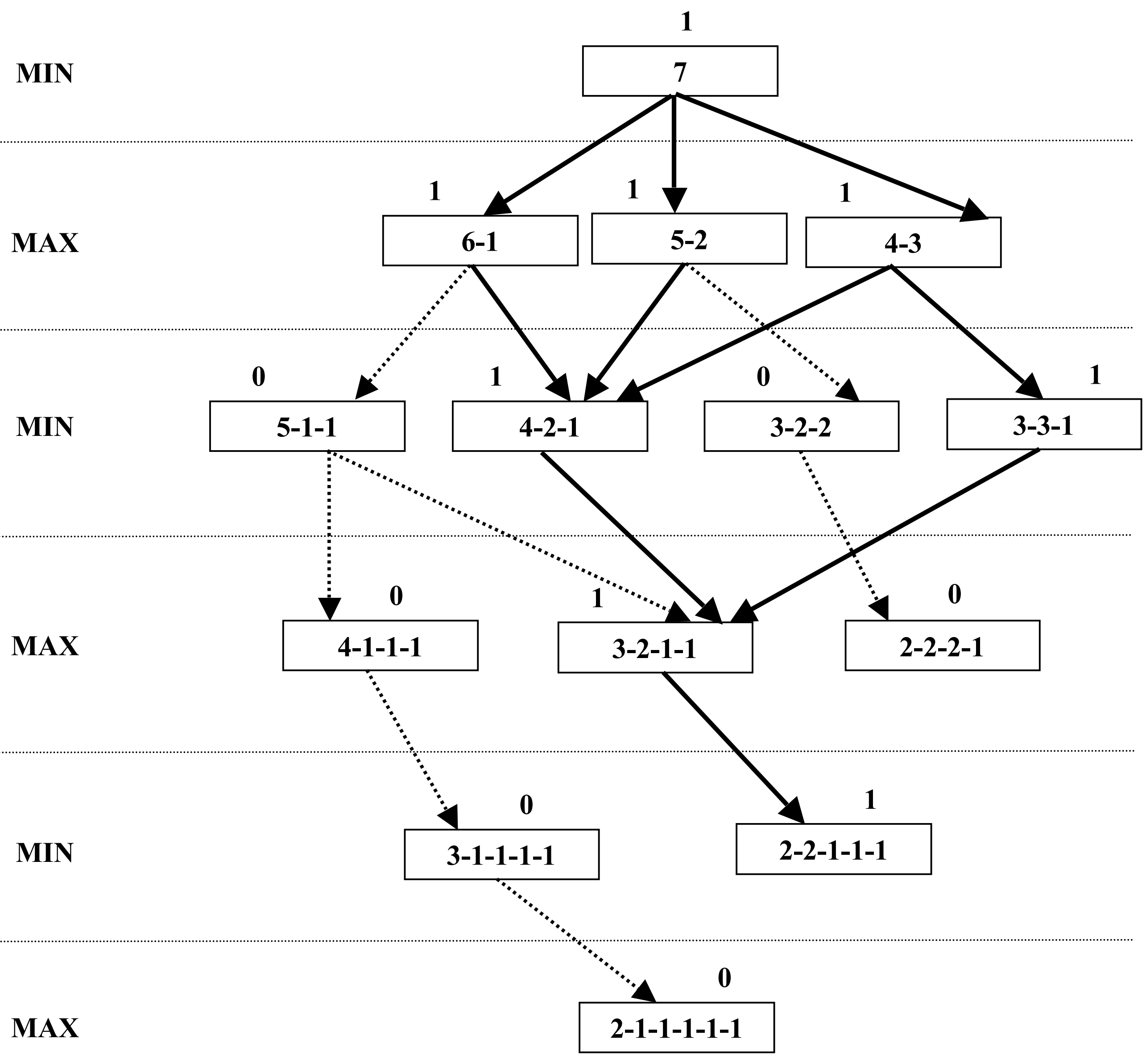
Question 1 – Possible solutions

(a) Draw the complete search tree for nim.

The search tree is shown below. The student might draw a search tree which duplicates some of the nodes. This is acceptable.
Note : At this stage, only the search tree is required. The "Min", "Max" and the Utility Functions are not required.



(b) Assume two players, min and max, play nim (as described above). Min plays first.

If a terminal state in the search tree developed above is a win for min, a utility function of zero is assigned to that state. A utility function of 1 is assigned to a state if max wins the game.

Apply the minimax algorithm to the search tree to assign utility functions to all states in the search tree.

The utility functions are shown in the search tree above. The student should initially assign values to the terminal states and then propagate these up the tree, depending on whether it is min or max's turn to move (min minimises the utility function of its child nodes, max maximises).

The student should mark the tree with the players turn – deduct one mark if this is not done. The other two marks are for correctly assigning the correct utility vales to each node. Give one mark if the student appears to be doing it correctly but has made a mistake.

(c) If both min and max play a perfect game, who will win? Explain your answer.

As the value at the top of the search tree is 1, it shows that maximise is guaranteed to win (if it plays a perfect game). This is because minimise will never be able to follow a path that leads to a utility function of 0. One mark for pointing this out.

Two marks for showing the path that would be taken, if both players played a perfect game. These are shown in bold in the search tree above.

(d) Given the following search tree, apply the alpha-beta pruning algorithm to it and show the search tree that would be built by this algorithm. Make sure that you show where the alpha and beta cuts are applied and which parts of the search tree are pruned as a result.

The search tree that the student should re-produce is shown below.

If the student produces this search tree they should receive 6 marks (note any extra nodes they produce should be penalised as the idea behind the alpha-beta algorithm is that it restricts the number of nodes that are expanded).

The other three marks are to be allocated based on the way the student explains how the search tree was built. One mark for stating that alpha-beta pruning ***must*** use Depth First Search.
One mark for explaining why the alpha cut off can be made. One mark why the beta cut off can be made.

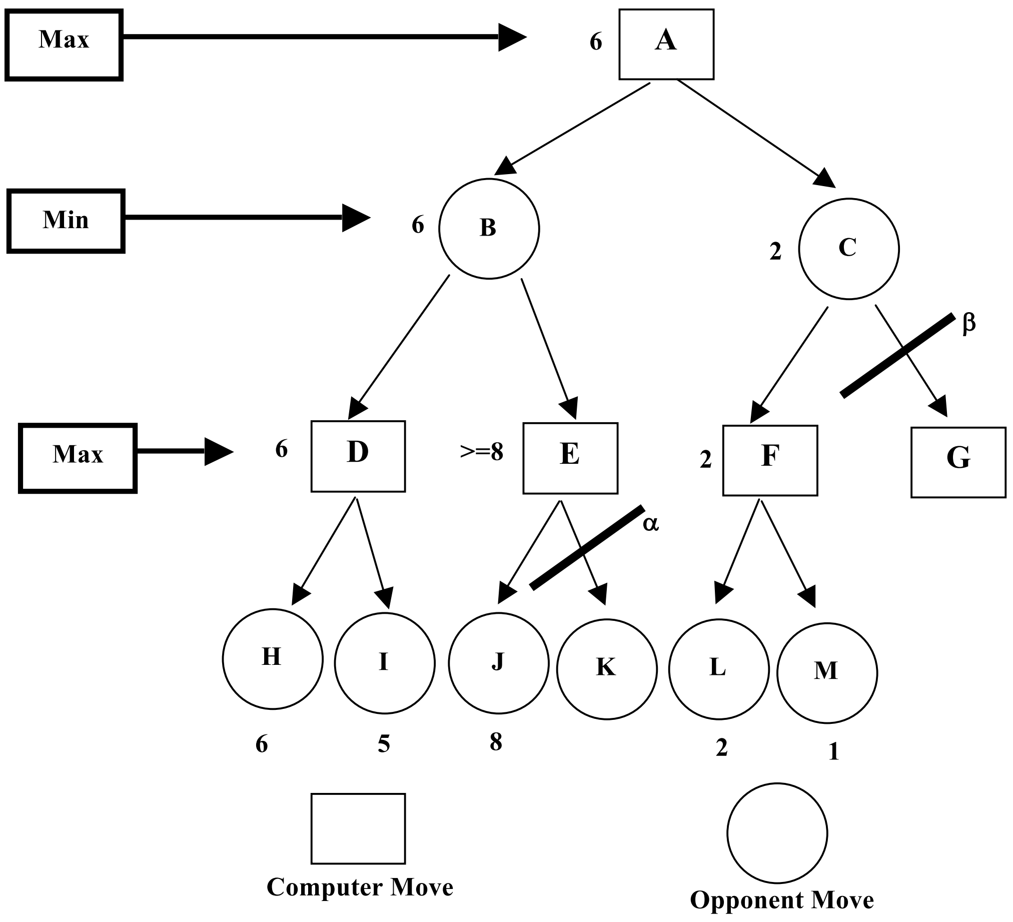
This is the explanation given in the course notes

Assume we have evaluated, using depth first search down to node I. This means we can maximise node D to a value of 6. If we now continue the search we will eventually evaluate node J and assign it a value of 8. At this point we do not need to evaluate K (or any of its siblings); for the following reasons.

- Node E already has a value of at least 8 (as MAX is trying to maximise this value so it cannot take on a smaller value).
- Node E is already greater than node D and, as MIN will be trying to minimise these two values, it is always going to choose D over E.
- Therefore, we can cut off the search at this point (shown an the diagram).

Similarly, we can cut of the search from node G downwards.

- Having evaluated F to 2, C cannot be more than this value (as C is trying to minimise).
- B is already greater than 2.
- Therefore, MAX is always going to prefer B over C, so we cannot cut off the rest of the search below C.



Question 2 – Model Answer

a) Given the following parents, P_1 and P_2, and the template T

Uniform

P_1	A	B	C	D	E	F	G	H	I	J
P_2	E	F	J	H	B	C	I	A	D	G
T	1	0	1	1	0	0	0	1	0	1

C_1	A	F	C	D	B	C	I	H	D	J
C_2	E	B	J	H	E	F	G	A	I	G

Order-Based

P_1	A	B	C	D	E	F	G	H	I	J
P_2	E	F	J	H	B	C	I	A	D	G
T	1	0	1	1	0	0	0	1	0	1

C_1	A	E	C	D	F	B	I	H	G	J
C_2	E	B	J	H	C	D	F	A	I	G

4 marks each

b) Given the following four chromosomes give the values for x and f(x).

Chromosome	Binary String	x	f(x)
P_1	11100	28	212
P_2	01111	15	3475
P_3	10111	23	1227
P_4	00100	4	2804

2 marks (half mark each)

c) If P_3 and P_2 are chosen as parents and we apply one point crossover show the resulting children, C_1 and C_2. Use a crossover point of 1 (where 0 is on the very left of the chromosome)
Do the same using P_4 and P_2 with a crossover point of 2 and create C_3 and C_4

Chromosome	Binary String	x	f(x)
C_1	11111	31	131
C_2	00111	7	3803
C_3	00111	7	3803
C_4	01100	12	3998

x and f(x) can be ignored for this question (see (d)).

4 marks (1.5 marks for each child they get correct)

d) Calculate the value of x and f(x) for $C_1..C_4$.

These are shown in the above table

2 marks (half mark each)

e) Assume the initial population was x={17, 21, 4 and 28}. Using one-point crossover, what is the probability of finding the optimal solution? Explain your reasons.

The probability is zero

1 mark

If we look at the values in binary we get

x	Binary
17	10001
21	10101
4	00100
28	11100

We know (although for any realistic problem we would not) that the global optimum is x = 10 which, in binary is 01010.
You can see that we need a 1 in positions 2 and 4 (counting from the right – although in this case it does not matter). In the initial population there is no individual with a 1 in position 2. This means that no matter how many times we apply single point crossover we will never be able to find the optimum.

3 marks

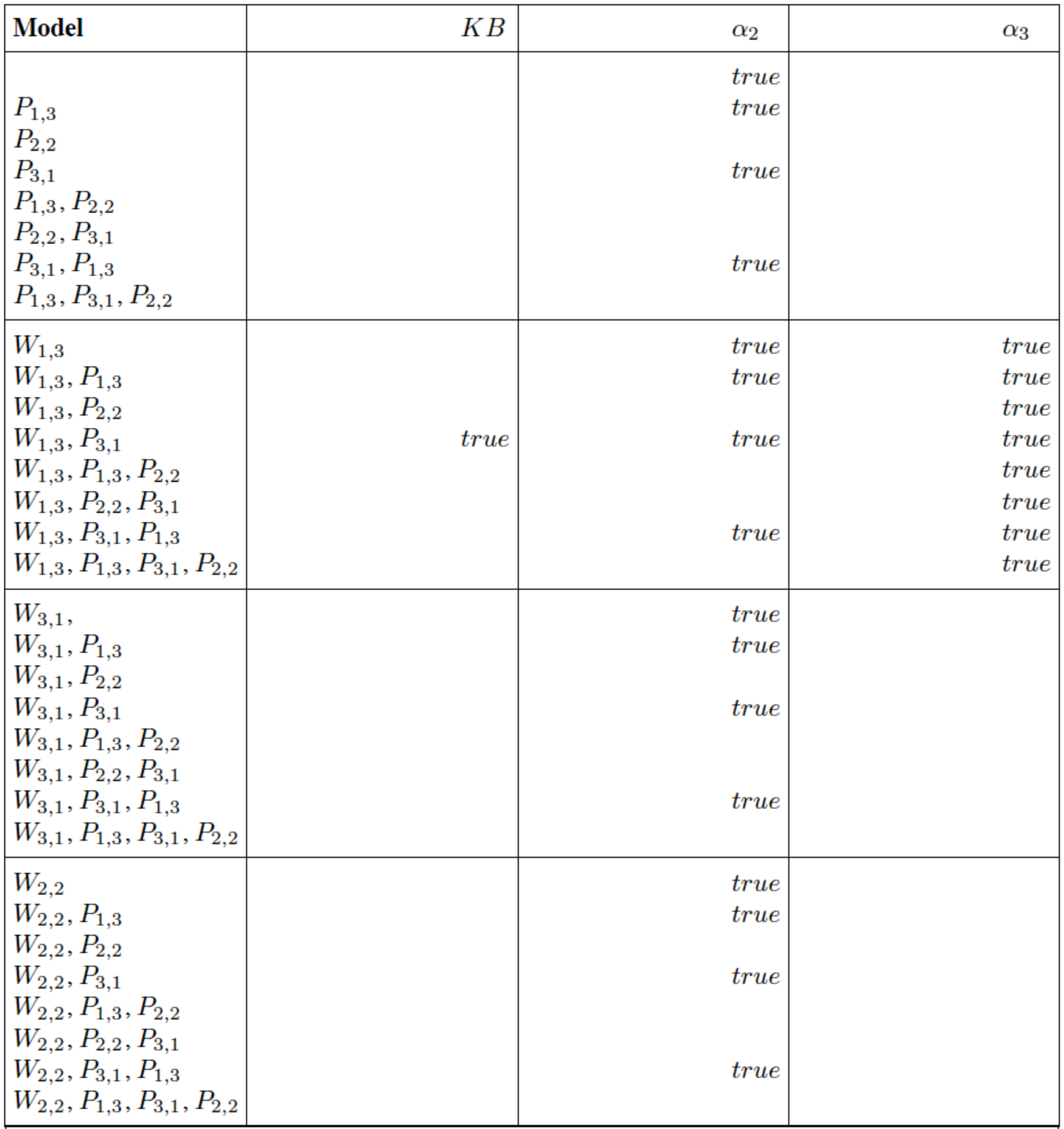
To overcome this we would need to introduce a mutation operator that flips bit with some low probability.

3 marks

Q3.

There are eight possible combinations of pits in the three squares, and four possibilities for the wumpus location (including nowhere).

We can see that KB |= α2 because every line where KB is true also has α2 true. Similarly, for α3.



Model	KB	α_2	α_3
		true	
$P_{1,3}$		*true*	
$P_{2,2}$			
$P_{3,1}$		*true*	
$P_{1,3}, P_{2,2}$			
$P_{2,2}, P_{3,1}$			
$P_{3,1}, P_{1,3}$		*true*	
$P_{1,3}, P_{3,1}, P_{2,2}$			
$W_{1,3}$		*true*	*true*
$W_{1,3}, P_{1,3}$		*true*	*true*
$W_{1,3}, P_{2,2}$			*true*
$W_{1,3}, P_{3,1}$	*true*	*true*	*true*
$W_{1,3}, P_{1,3}, P_{2,2}$			*true*
$W_{1,3}, P_{2,2}, P_{3,1}$			*true*
$W_{1,3}, P_{3,1}, P_{1,3}$		*true*	*true*
$W_{1,3}, P_{1,3}, P_{3,1}, P_{2,2}$			*true*
$W_{3,1},$		*true*	
$W_{3,1}, P_{1,3}$		*true*	
$W_{3,1}, P_{2,2}$			
$W_{3,1}, P_{3,1}$		*true*	
$W_{3,1}, P_{1,3}, P_{2,2}$			
$W_{3,1}, P_{2,2}, P_{3,1}$			
$W_{3,1}, P_{3,1}, P_{1,3}$		*true*	
$W_{3,1}, P_{1,3}, P_{3,1}, P_{2,2}$			
$W_{2,2}$		*true*	
$W_{2,2}, P_{1,3}$		*true*	
$W_{2,2}, P_{2,2}$			
$W_{2,2}, P_{3,1}$		*true*	
$W_{2,2}, P_{1,3}, P_{2,2}$			
$W_{2,2}, P_{2,2}, P_{3,1}$			
$W_{2,2}, P_{3,1}, P_{1,3}$		*true*	
$W_{2,2}, P_{1,3}, P_{3,1}, P_{2,2}$			

Figure S7.1 A truth table constructed for Ex. 7.2. Propositions not listed as true on a given line are assumed false, and only *true* entries are shown in the table.

NB—Students may have drawn this instead of table. So be careful when grading it.

Q4.
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 – Computing 1: meets from 8:00-9:00am
- Class 2 - Artificial Intelligence: meets from 8:30-9:30am
- Class 3 - Natural Language: meets from 9:00-10:00am
- Class 4 - Computer Vision: meets from 9:00-10:00am
- Class 5 - Machine Learning: meets from 9:30-10:30am

The professors are:

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

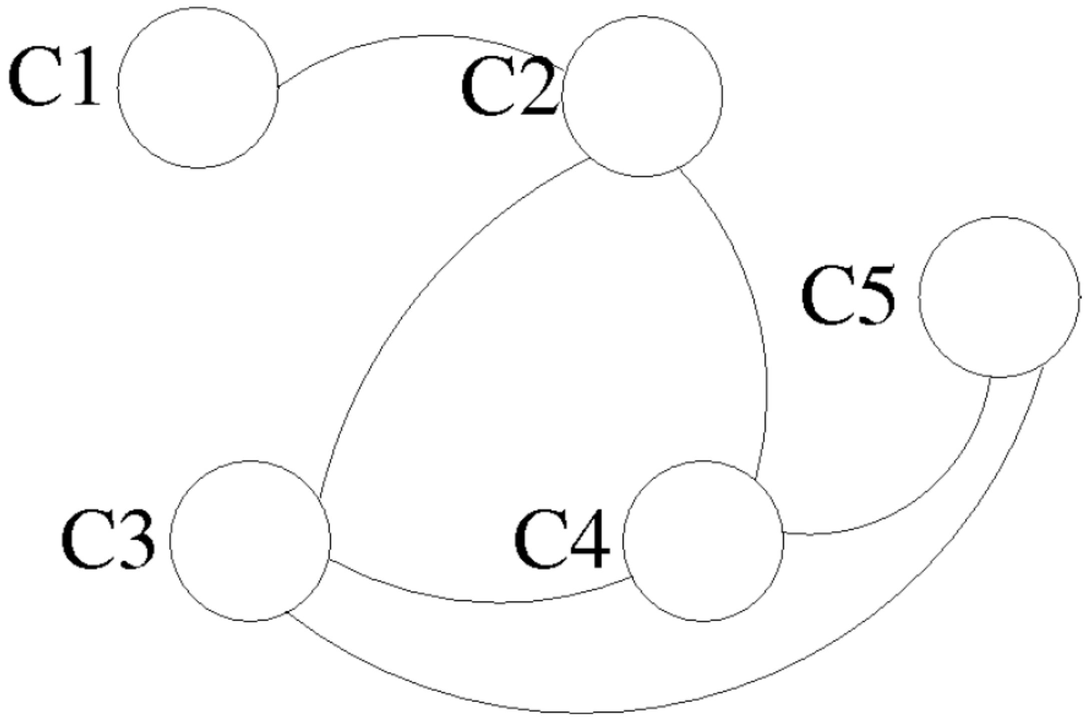
a) Formulate this problem as a CSP problem in which there is one variable per class, stating the domains, and constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit. (4)

Variables	Domains (or unary constraints)
C1	C
C2	B,C
C3	A,B,C
C4	A,B,C
C5	B,C

Constraints:

C1 != C2	C3 != C4	C4 != C5
C2 != C3	C2 != C4	C3 != C5

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



c) Show the domains of the variables after running arc-consistency on this initial graph (after having already enforced any unary constraints). (4)

Variable	Domain
C1	C
C2	B
C3	A,C
C4	A,C
C5	B,C

Note that C5 cannot possibly be C, but arc consistency does not rule it out.

d) Give one solution to this CSP. (1)
C1 = C, C2 = B, C3 = C, C4 = A, C5 = B.
One other solution is possible (where C3 and C4 are switched).

e) Your CSP should look nearly tree-structured. Briefly explain (one sentence or less) why we might prefer to solve tree-structures CSPs. (2)

Minimal answer: Can solve them in polynomial time. If a graph is tree structured (i.e. has no loops), then the CSP can be solved in O(nd2) time as compared to general CSPs, where worstcase time is O(dn). For tree-structured CSPs you can choose an ordering such that every node's parent precedes it in the ordering. Then you can greedily assign the nodes in order and will find a consistent assignment without backtracking.

f) Name (or briefly describe) a standard technique for turning these kinds of nearly tree structured problems into tree-structured ones. (2)

Minimal answer: cutset conditioning. One standard technique is to instantiate cutset, a variable (or set of variables) whose removal turns the problem into a tree structured CSP. To instantiate the cutset you set its values in each possible way, prune neighbors, then solve the reduced tree structured problem (which is fast).

--Some students might mention tree decomposition (please evaluate the answer depending on the explanation)