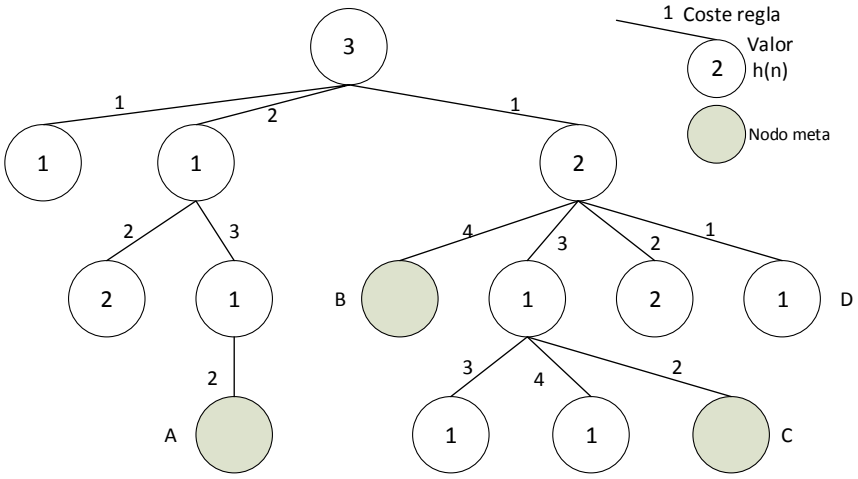
Intelligent Systems – Final exam (Block 1), January 17 2018
Test (2 points) score: max (0, (#correct_answers – #wrong_answers/3)/3)

Family name:	**First name**:
Group: A B C D E F FLIP	

1) If we apply a *Greedy* search on the search space of the figure, which goal node (shadowy node) will be selected first as a solution and how many nodes are generated to find such a solution?



A. Goal node A generating 7 nodes
B. Goal node B generating 8 nodes
C. Goal node B generating 11 nodes
D. Goal node C generating 14 nodes

---

2) Given the search space of the above figure, mark the **INCORRECT** statement:

A. The heuristic function h(n) is admissible
B. The heuristic function h(n) is consistent
C. A Breadth-first algorithm will find the same solution as an algorithm of type A
D. A Depth-first algorithm will find the same solution as a greedy search algorithm

---

3) Let d1, d2 and d3 be three depth levels of a search tree where d1 < d2 < d3, such that there exists one solution at level d1, another solution at level d2 and another solution at level d3 (there is only one solution at each level). Mark the **CORRECT** statement:

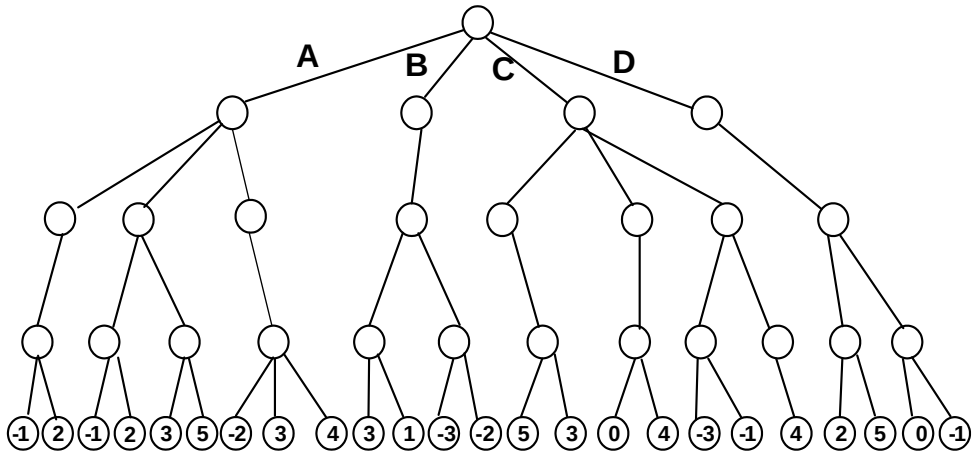
A. The time complexity of a Breadth-first algorithm is $O(b^{d2})$ and the time complexity of an Iterative Deepening algorithm is $O(b^{d1})$.
B. The time complexity of a Depth-first tree search with maximum depth limit m=d1, is $O(b^{d1+1})$.
C. Assuming the user selects m=d3 as maximum depth limit, a Depth-first tree search will always find first the solution at level d1 or at level d2.
D. Assuming the user selects m=d1 as maximum depth limit, the time complexity of a Depth-first tree search and an Iterative Deepening algorithm is $O(b^{d1})$ for both.

4) Let's assume an algorithm A* is applied to solve a problem and that *G* is the solution node found. Show the **FALSE** statement:

A. If h(n) is consistent then ∀n1, n2 such that n2 is a child node of n1, it always holds h(n2)>=h(n1)
B. ∀n1, n2, such that n1 and n2 are two nodes on the optimal path to *G*, it always holds g(n1)+h*(n1)=g(n2)+h*(n2)
C. ∀n, such that n is a node on the optimal path to *G*, it always holds f(n) <= g(*G*)
D. It always holds f(*G*)=g(*G*).

---

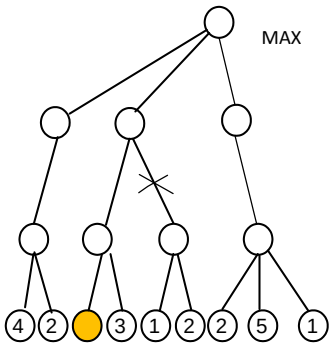
5) Which branch of the game tree of the figure below will be chosen if we apply the MINIMAX algorithm?



A. A
B. B
C. C
D. D

---

6) Which values should the shadowy node have so that the cutoff of the figure is always produced?



A. Any value in [-∞ 4].
B. Any value.
C. Any value in [4 +∞].
D. The cutoff can never happen.

Intelligent Systems – Final exam (Block 1), January 17 2018
Problem: 3 points

Sokoban game

The figure below shows a board of the Sokoban game. Each cell contains an obstacle (O), represented with light colour squares; a box (B), shown with dark squares that contain a cross; a store (S), shown with a circle; or nothing (N). Likewise, there is a player (P) located in one of the cells. The purpose of the game is for the player to push the boxes to the stores. The player can move in one of the four possible directions: down, up, right and left, and a box must also be pushed in one of these 4 directions.

The above figure represents the initial state of a particular problem. In this state, the player (P) is in the same position as the store of the upper row. In order to push a box, the player must be located in a cell adjacent to the box and push it to a cell that contains a store (S) or nothing (N). Given the above example, if the player wants to push the box of the upper row , the player can:

- go to the cell on the right of the box and push the box towards left; the effect of this operations is that both the box and player are moved one cell to the left.
- go to the cell on the left of the box and push the box towards right; the effect of this operation is that both the box and the player are moved one cell to the right.
- go to the cell above the box is not allowed because this cell contains an obstacle (O)
- go to the cell below the box but then the player will not be able to push the box upwards because there is an obstacle (O) in the cell above the box

We want to design a RBS in CLIPS to solve this problem. To do this, we will use the following pattern representation:

$$(\text{sokoban P } rp^s\ cp^s\ [\text{pos } rc^s\ cc^s\ v^s]^m) \text{ where}$$

rp, cp, rc, cc $\in$ INTEGER ;; rp and cp represent the row and columnn of the player (P); rc and cc represent the row and column of each of the cells
v $\in$ {O,B,S,N} ;; represents the contents of the cell

The position of the board cells must appear ordered by rows (from top to bottom) and by columns (from left to right). It is not necessary to represent the obstacles around the board in the example of the figure; it suffices to use a 4-row x 5-column representation. For instance, the position (1,1) indicates the cell in

the upper row, left column; and the position (3,2) represents the cell located in the third row starting at the top, second column starting at the left, which contains a box (B).

For the ease of the design, we will assume:

- it is not necessay to explictly represent in the board whenever a box (B) is at a store (S); that is, when the player pushes a box to a cell that contains a store, we will eliminate the box from the representation
- stores can keep any number of boxes

Answer the following questions:

1) (0.3 points) Represent the initial state of the figure
2) (0.8 points) Write a rule to move the player (P) to the right
3) (1.3 points) Write a rule that allows the player to push a box **upwards** to a cell that IS NOT a store
4) (0.6 points) Assuming we have rules that check out whenever a box is pushed to a store, and that the effect of such rules is to eliminate the box from the board representation, write a rule that is triggered when the problem is solved

1)
```
(deffacts data
    (sokoban P 1 5 pos 1 1 N pos 1 2 B pos 1 3 N pos 1 4 N pos 1 5 S pos 2 1 N pos 2 2 N
                pos 2 3 N pos 2 4 B pos 2 5 S pos 3 1 N pos 3 2 B pos 3 3 O pos 3 4 N
                pos 3 5 S pos 4 1 N pos 4 2 N pos 4 3 O pos 4 4 N pos 4 5 N))
```

2)
```
(defrule move_right
        (sokoban P ?xj ?yj $?res1 pos ?xj ?yc ?v $?res2)
        (test (= ?yj (- ?yc 1)))
        (test (and (neq ?v O)(neq ?v B)))
    =>
        (assert (sokoban P ?xj ?yc $?res1 pos ?xj ?yc ?v $?res2)))
```

3)
```
(defrule push_upwards
    (sokoban P ?xj ?yj $?res1 pos ?xn ?yj ?v $?res2 pos ?xc ?yj B $?res3)
    (test (= ?xj (+ ?xc 1)))
    (test (= ?xn (- ?xc 1)))
    (test (and (neq ?v O)(neq ?v B)(neq ?V S)))  ;; (test (eq ?v N))
 =>
    (assert (sokoban P ?xc ?yj $?res1 pos ?xn ?yj B $?res2 pos ?xc ?yj N $?res3)))
```

4)
```
(defrule final
    (declare (salience 100))
    (sokoban $?res1)
    (test (not (member B $?res1)))
 =>
    (printout t "Solution found " crlf)
    (halt))
```