1. Let $S$ be a state space in which every edge has the same cost, and $h(n)$ be a heuristic function that always returns 0. True or false:

(a) (5 points.) If we run A* with a breadth-first tie-breaking rule, it will expand the nodes in the same order that a breadth-first search would expand them.

**Answer.** True.

(b) (5 points.) If we run A* with a depth-first tie-breaking rule, it will expand the nodes in the same order that a depth-first search would expand them.

**Answer.** False. It will still do a breadth-first search, but for each level of the search tree, the nodes at level will be expanded in a different order than in part (a).

2. Let $S$ be a state space in which all edge-costs are integers $> 0$. Let $n$ be the optimal solution cost. Let $h$ be a monotone, admissible monotone heuristic function that always returns integers. For every integer $i > 0$, suppose there are $\Theta(i^2)$ nodes whose $f$-value is $i$.

(a) (5 points.) If we run A* on $S$ using $h$, how many node generations will it do? Your answer should be a $\Theta$-value.

**Answer.** $\Theta(1^2 + \ldots + (n - 1)^2) + O(n^2) = \Theta(n^3)$.

(b) (5 points.) If we run IDA* on $S$ using $h$, how many node generations will it do? Your answer should be a $\Theta$-value.

**Answer.** $\sum_{i=1}^{n}[\Theta(1^2 + \ldots + (n - 1)^2) + O(n^2)] = \sum_{i=1}^{n} \Theta(n^3) = \Theta(n^4)$.

(c) (10 points.) In the problem definition, why did I specify that $h$ is monotone?

**Answer.** If $h$ weren’t monotone, then the above answers wouldn’t always be correct. For example, we could construct a situation in which there are $\Theta(i^2)$ nodes whose $f$-value is $i$, but all but one of those nodes (namely the one on the optimal path) has a pathmax value that is $> n$. In that case, A* would do $\Theta(n)$ node generations, and IDA* would do $\Theta(n^2)$ node generations.
3. In the game tree shown at right, suppose every node has exactly $k$ children (where $k$ is a constant). Suppose we are doing an alpha-beta search without using any node-ordering heuristics (i.e., the algorithm visits the nodes in left-to-right order). Suppose that the search algorithm has just returned a value of 1 for the node $b$.

(a) (4 points.) From the information above (i.e., without running alpha-beta any farther), what can we say about the minimax value of $a$?

**Answer.** It is $\geq 1$.

(b) (4 points.) If we continue to run alpha-beta, how many of $c$’s children will it need to examine in the worst case? In the best case?

**Answer.** The worst case occurs if all children of $c$ have minimax values $> 1$, in which case alphabeta will need to examine all $k$ of them. The best case occurs if the first child of $c$ has a minimax value $\leq 1$, in which case alphabeta will only need to examine that one child.

(c) (4 points.) Which minimax values for $d$ would produce the best case for part (b)?

**Answer.** Any value $\leq 1$, as explained above.

(d) (4 points.) How many of $d$’s children will alpha-beta need to examine in the worst case? In the best case?

**Answer.** The answer is $k$ in both cases. We can’t prune any children of $d$, because it is Max’s move at $d$ and $\beta = \infty$.

4. Let $T$ be a game tree, $t$ be a nonterminal node of $T$, and $m$ be $t$’s minimax value. Let $\alpha$ and $\beta$ be numbers, and let $v$ be the value returned by alphabeta($t, d, \alpha, \beta$). What can you say about $v$ in each of the following cases?

(a) (5 points.) The case where $m < \alpha < \beta$.

**Answer.** $m \leq v \leq \alpha$.

(b) (5 points.) The case where $\alpha < \beta < m$.

**Answer.** $m \geq v \geq \beta$.

(c) (5 points.) The case where $\alpha < m < \beta$.

**Answer.** $v = m$. 

2
5. Here are three definitions of the Lisp function `copy-list` (which makes a copy of a list).

```lisp
(defun copy-list-1 (x)
  (let ((result nil))
    (dolist (item x result)
      (setf result (append result (list item)))))

(defun copy-list-2 (x)
  (let ((result nil))
    (dolist (item x (reverse result))
      (setq result (cons item result)))))

(defun copy-list-3 (x)
  ;; below, (IDENTITY X) is a built-in function that simply returns X
  (mapcar #'identity x))
```

(a) (5 points.) Which implementation is least preferable? Explain why.

**Answer.** The least preferable one is `copy-list-1`, because it has a quadratic running time and the other two implementations have linear running time.

(b) (5 points.) Which implementation is most preferable? Explain why.

**Answer.** `copy-list-3` is most preferable. It has lower overhead than `copy-list-2`, because it doesn’t have to reverse the list after constructing it.

6. (4 points.) Let $T$ be a first-order theory, and let $B$ be a statement such that $\neg B$ is a theorem of $T$. Let $D$ be a domain, and $I$ be an interpretation of $T$ in $D$. Suppose that for every axiom $A$ of $T$, $I(A)$ is true in $D$. What can we say about $I(B)$?

**Answer.** From the Completeness Theorem it follows that $I(\neg B)$ is true in $D$, and thus $I(B)$ must be false in $D$.

7. (5 points.) Let $A$ be a statement, and suppose $T$ is a first-order theory such that both $A$ and $\neg A$ are axioms of $T$. What can we say about the models of $T$?

**Answer.** If $(I, D)$ were a model of $T$, then both $I(A)$ and $I(\neg A)$ would have to be true in $D$, which is impossible. Thus $T$ has no models.
8. For each pair of clauses, find all of the most-general resolvents (if there are any). For each one, give both the resolvent and the unifier. The symbols $x, y, v$ all are variable symbols.

(a) (5 points.) $P(y,f(x)) \lor Q(f(x),y)$ and $\neg P(v,f(v)) \lor \neg Q(v,f(v))$

**Answer.**

We can resolve the two “P” atoms using the unifier $\{x = v, y = v\}$, giving the resolvent $Q(f(v),v) \lor \neg Q(v,f(v))$.

We can resolve the two “Q” atoms using the unifier $\{v = f(x), y = f(f(x))\}$, giving the resolvent $P(f(f(x)),f(x)) \lor \neg P(f(x),f(f(x)))$.

(b) (5 points.) $P(y,f(x)) \lor Q(g(x),y)$ and $P(v,f(v)) \lor \neg Q(v,f(v))$

**Answer.** The two “P” clauses can’t be resolved because neither of them is negated.

We can resolve the two “Q” clauses using the unifier $\{v = g(x), y = f(g(x))\}$, giving the resolvent $P(f(g(x)),f(x)) \lor P(g(x),f(g(x)))$.

(c) (5 points.) $P(y,g(x)) \lor Q(y,f(x))$ and $\neg P(x,f(x)) \lor \neg Q(f(x),x)$

**Answer.** This time, we can’t resolve anything unless we first standardize the variables. To do that, let’s rename $x$ to $v$ in the second clause.

Even after standardizing the variables, we still can’t resolve the two “P” clauses since we can’t make $f = g$. However, we can resolve the two “Q” clauses using the unifier $\{v = f(x), y = f(f(x))\}$, giving $P(f(f(x)),g(x)) \lor \neg P(f(x),f(f(x)))$.

(d) (5 points.) $P(f(x),x) \lor Q(f(y),y)$ and $\neg P(v,f(v)) \lor \neg Q(v,f(v))$

**Answer.** There are no resolvents, because nothing unifies.