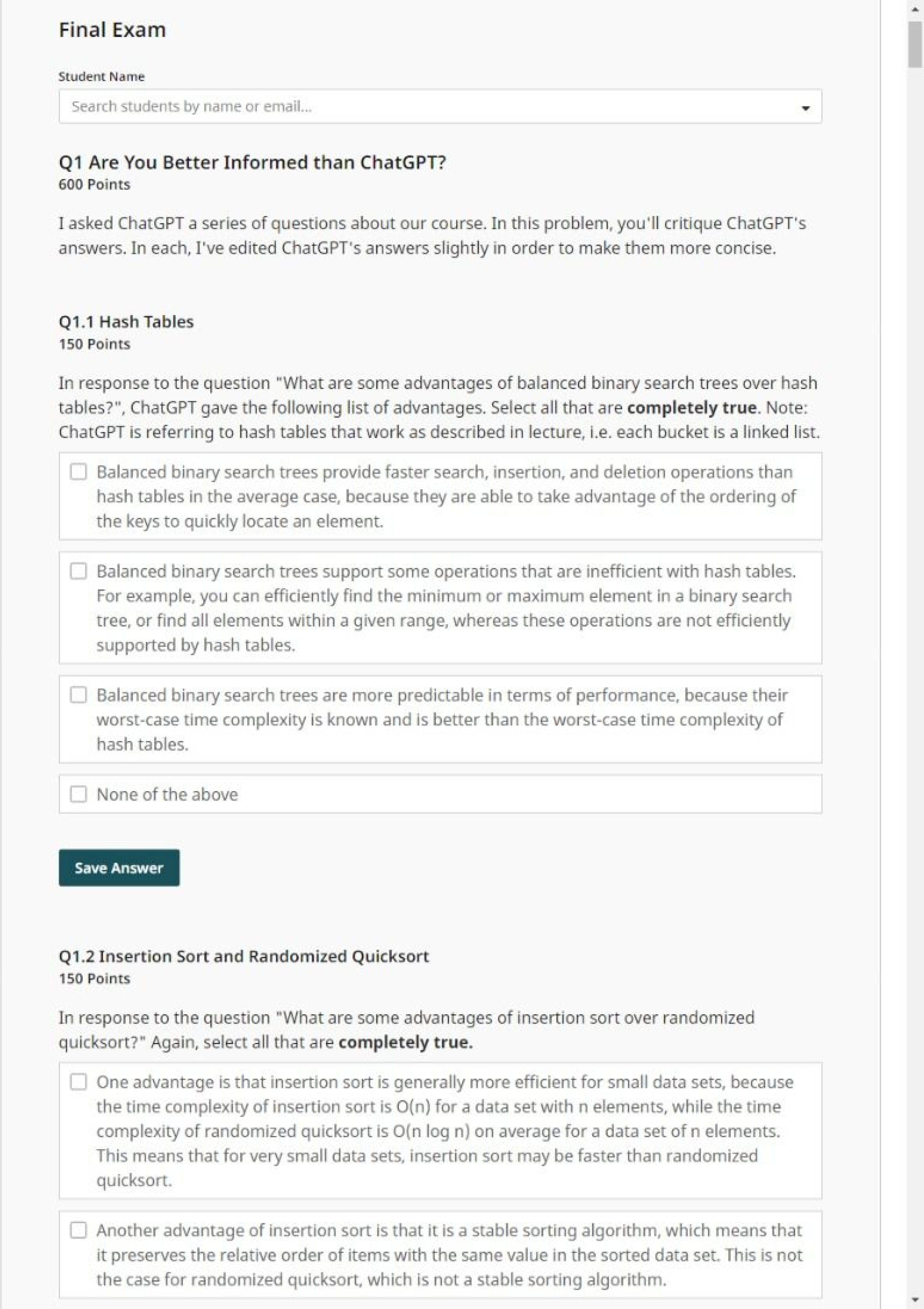


Final Exam

Student Name

Q1 Are You Better Informed than ChatGPT?

600 Points

I asked ChatGPT a series of questions about our course. In this problem, you'll critique ChatGPT's answers. In each, I've edited ChatGPT's answers slightly in order to make them more concise.

Q1.1 Hash Tables

150 Points

In response to the question "What are some advantages of balanced binary search trees over hash tables?", ChatGPT gave the following list of advantages. Select all that are **completely true**. Note: ChatGPT is referring to hash tables that work as described in lecture, i.e. each bucket is a linked list.

□ Balanced binary search trees provide faster search, insertion, and deletion operations than hash tables in the average case, because they are able to take advantage of the ordering of the keys to quickly locate an element.

□ Balanced binary search trees support some operations that are inefficient with hash tables. For example, you can efficiently find the minimum or maximum element in a binary search tree, or find all elements within a given range, whereas these operations are not efficiently supported by hash tables.

□ Balanced binary search trees are more predictable in terms of performance, because their worst-case time complexity is known and is better than the worst-case time complexity of hash tables.

□ None of the above

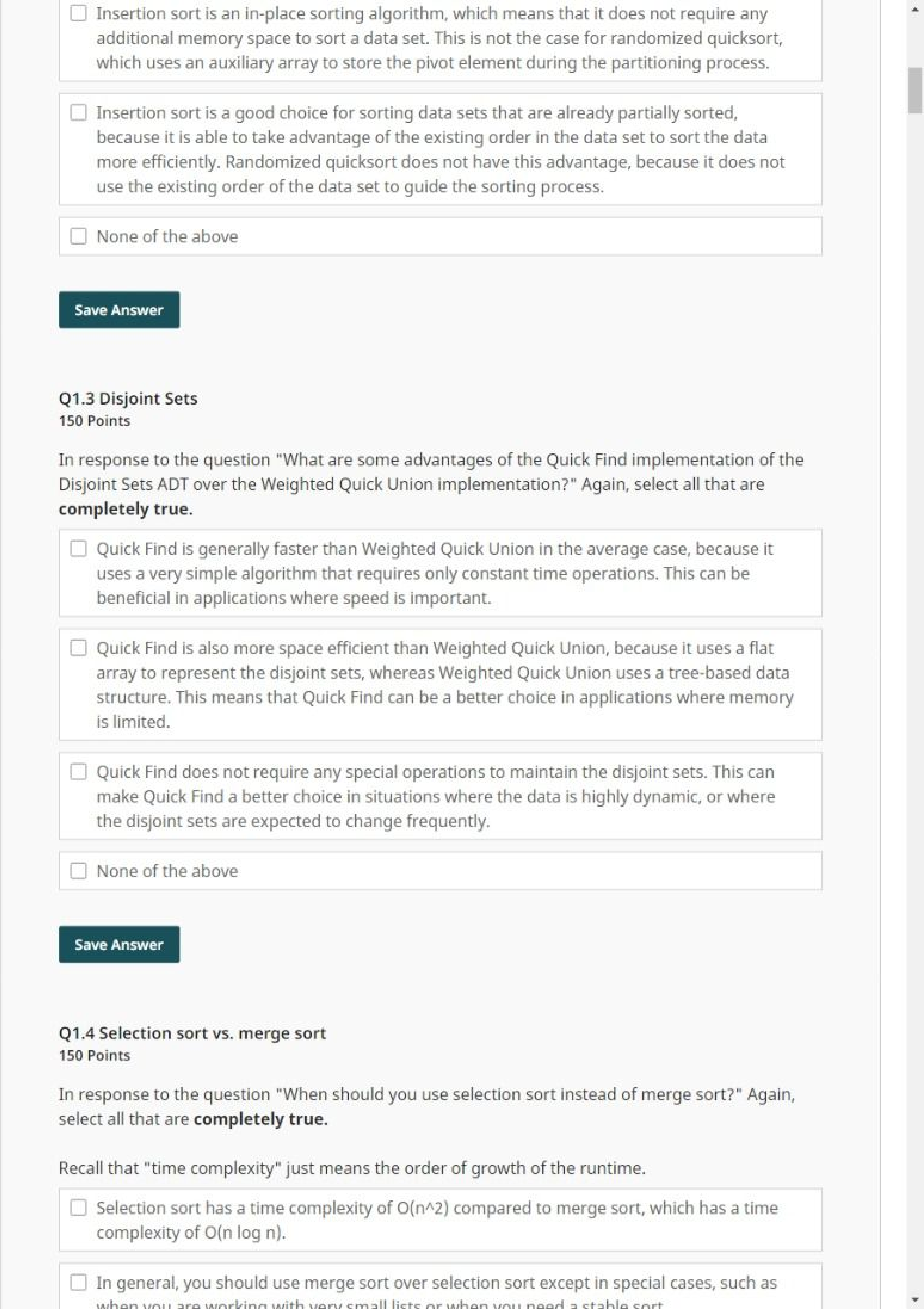
Q1.2 Insertion Sort and Randomized Quicksort

150 Points

In response to the question "What are some advantages of insertion sort over randomized quicksort?" Again, select all that are **completely true.**

□ One advantage is that insertion sort is generally more efficient for small data sets, because the time complexity of insertion sort is O(n) for a data set with n elements, while the time complexity of randomized quicksort is O(n log n) on average for a data set of n elements. This means that for very small data sets, insertion sort may be faster than randomized quicksort.

□ Another advantage of insertion sort is that it is a stable sorting algorithm, which means that it preserves the relative order of items with the same value in the sorted data set. This is not the case for randomized quicksort, which is not a stable sorting algorithm.



□ Insertion sort is an in-place sorting algorithm, which means that it does not require any additional memory space to sort a data set. This is not the case for randomized quicksort, which uses an auxiliary array to store the pivot element during the partitioning process.

□ Insertion sort is a good choice for sorting data sets that are already partially sorted, because it is able to take advantage of the existing order in the data set to sort the data more efficiently. Randomized quicksort does not have this advantage, because it does not use the existing order of the data set to guide the sorting process.

□ None of the above

Q1.3 Disjoint Sets

150 Points

In response to the question "What are some advantages of the Quick Find implementation of the Disjoint Sets ADT over the Weighted Quick Union implementation?" Again, select all that are **completely true.**

□ Quick Find is generally faster than Weighted Quick Union in the average case, because it uses a very simple algorithm that requires only constant time operations. This can be beneficial in applications where speed is important.

□ Quick Find is also more space efficient than Weighted Quick Union, because it uses a flat array to represent the disjoint sets, whereas Weighted Quick Union uses a tree-based data structure. This means that Quick Find can be a better choice in applications where memory is limited.

□ Quick Find does not require any special operations to maintain the disjoint sets. This can make Quick Find a better choice in situations where the data is highly dynamic, or where the disjoint sets are expected to change frequently.

□ None of the above

Q1.4 Selection sort vs. merge sort

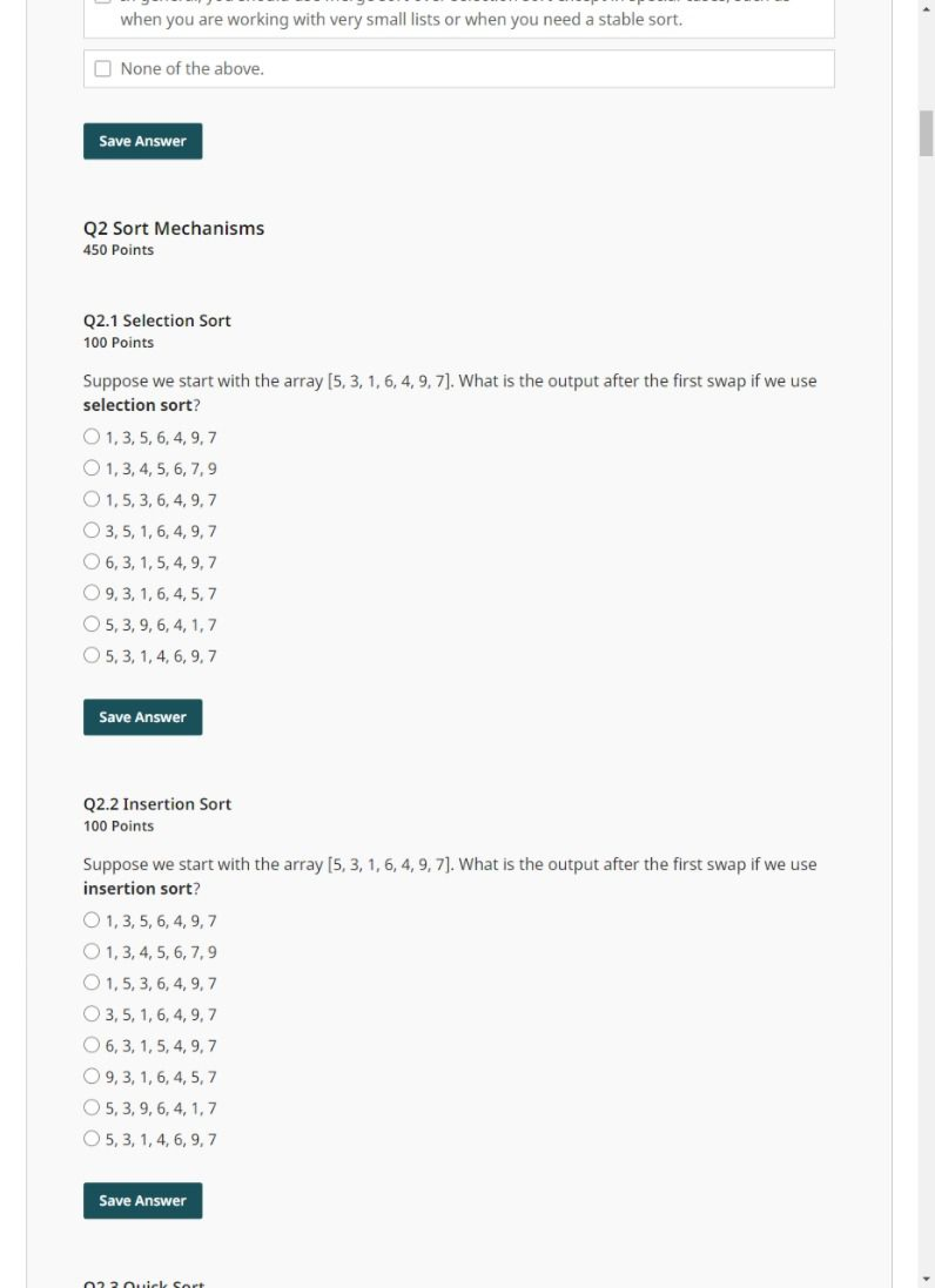
150 Points

In response to the question "When should you use selection sort instead of merge sort?" Again, select all that are **completely true.**

Recall that "time complexity" just means the order of growth of the runtime.

□ Selection sort has a time complexity of O(n^2) compared to merge sort, which has a time complexity of O(n log n).

□ In general, you should use merge sort over selection sort except in special cases, such as when you are working with very small lists or when you need a stable sort.



□ In general, you should use merge sort over selection sort except in special cases, such as when you are working with very small lists or when you need a stable sort.

□ None of the above.

Q2 Sort Mechanisms

450 Points

Q2.1 Selection Sort

100 Points

Suppose we start with the array [5, 3, 1, 6, 4, 9, 7]. What is the output after the first swap if we use **selection sort**?

○ 1, 3, 5, 6, 4, 9, 7

○ 1, 3, 4, 5, 6, 7, 9

○ 1, 5, 3, 6, 4, 9, 7

○ 3, 5, 1, 6, 4, 9, 7

○ 6, 3, 1, 5, 4, 9, 7

○ 9, 3, 1, 6, 4, 5, 7

○ 5, 3, 9, 6, 4, 1, 7

○ 5, 3, 1, 4, 6, 9, 7

Q2.2 Insertion Sort

100 Points

Suppose we start with the array [5, 3, 1, 6, 4, 9, 7]. What is the output after the first swap if we use **insertion sort**?

○ 1, 3, 5, 6, 4, 9, 7

○ 1, 3, 4, 5, 6, 7, 9

○ 1, 5, 3, 6, 4, 9, 7

○ 3, 5, 1, 6, 4, 9, 7

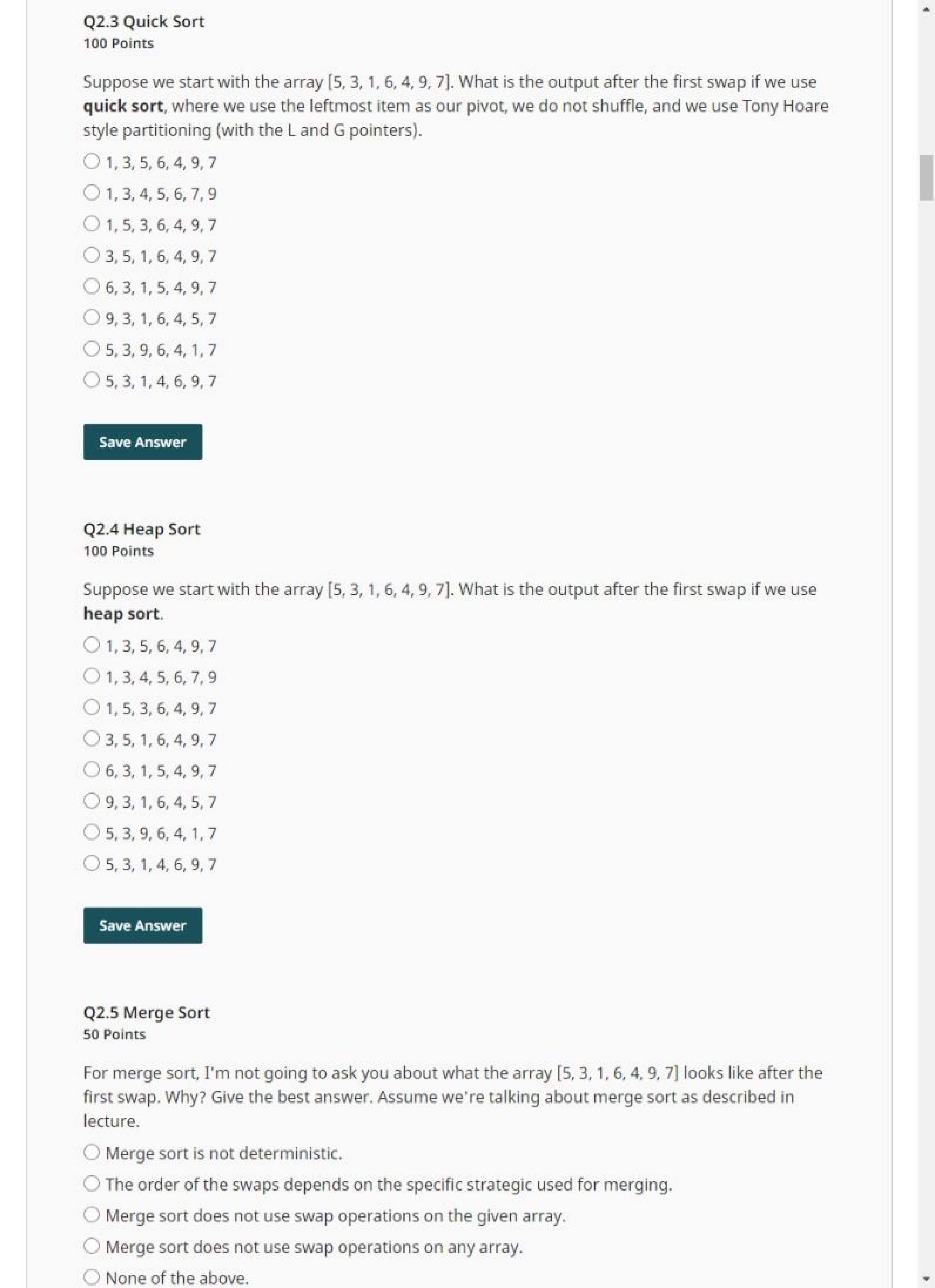
○ 6, 3, 1, 5, 4, 9, 7

○ 9, 3, 1, 6, 4, 5, 7

○ 5, 3, 9, 6, 4, 1, 7

○ 5, 3, 1, 4, 6, 9, 7

Q2.3 Quick Sort



Q2.3 Quick Sort

100 Points

Suppose we start with the array [5, 3, 1, 6, 4, 9, 7]. What is the output after the first swap if we use **quick sort**, where we use the leftmost item as our pivot, we do not shuffle, and we use Tony Hoare style partitioning (with the L and G pointers).

○ 1, 3, 5, 6, 4, 9, 7

○ 1, 3, 4, 5, 6, 7, 9

○ 1, 5, 3, 6, 4, 9, 7

○ 3, 5, 1, 6, 4, 9, 7

○ 6, 3, 1, 5, 4, 9, 7

○ 9, 3, 1, 6, 4, 5, 7

○ 5, 3, 9, 6, 4, 1, 7

○ 5, 3, 1, 4, 6, 9, 7

Q2.4 Heap Sort

100 Points

Suppose we start with the array [5, 3, 1, 6, 4, 9, 7]. What is the output after the first swap if we use **heap sort**.

○ 1, 3, 5, 6, 4, 9, 7

○ 1, 3, 4, 5, 6, 7, 9

○ 1, 5, 3, 6, 4, 9, 7

○ 3, 5, 1, 6, 4, 9, 7

○ 6, 3, 1, 5, 4, 9, 7

○ 9, 3, 1, 6, 4, 5, 7

○ 5, 3, 9, 6, 4, 1, 7

○ 5, 3, 1, 4, 6, 9, 7

Q2.5 Merge Sort

50 Points

For merge sort, I'm not going to ask you about what the array [5, 3, 1, 6, 4, 9, 7] looks like after the first swap. Why? Give the best answer. Assume we're talking about merge sort as described in lecture.

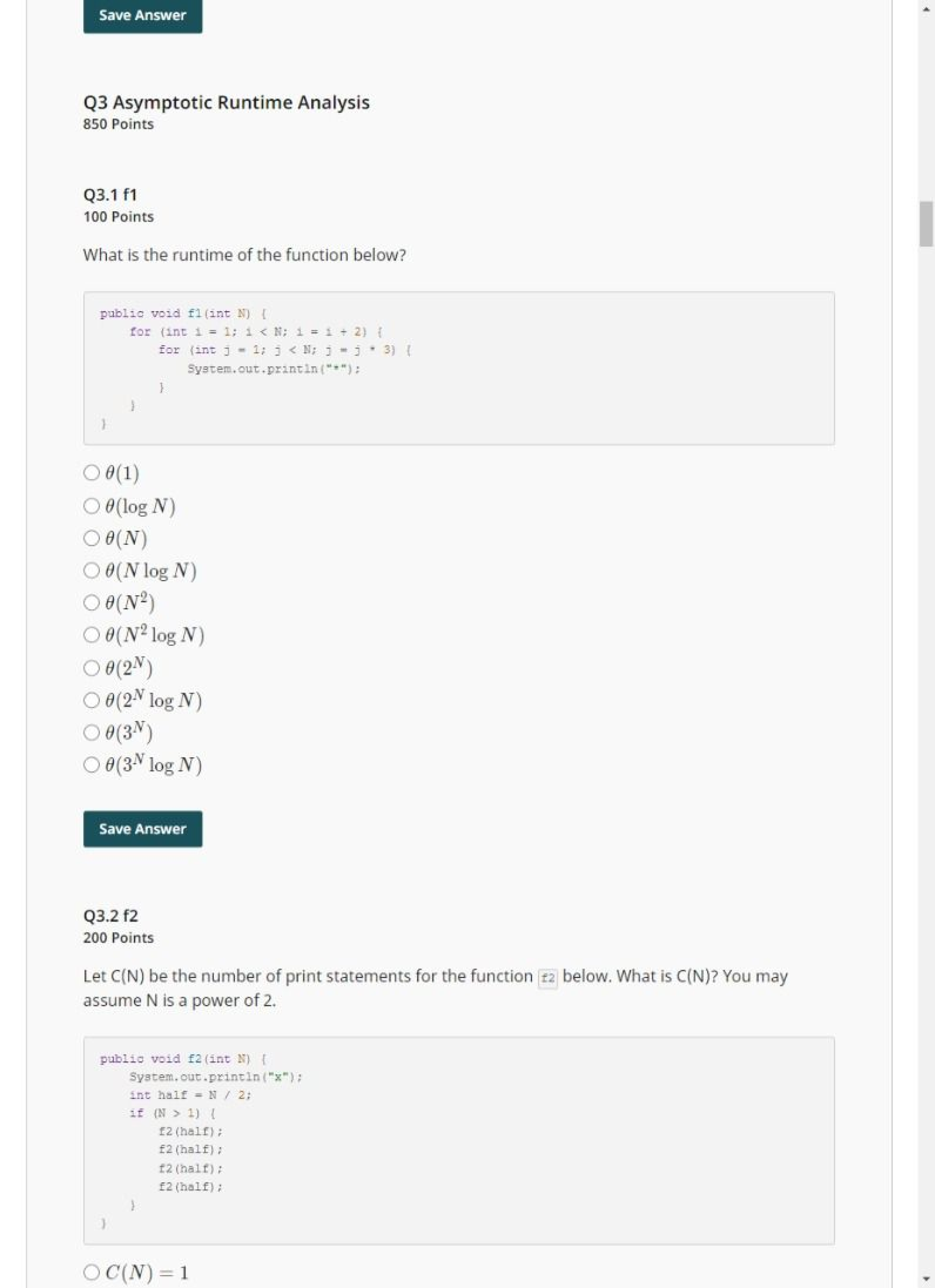
○ Merge sort is not deterministic.

○ The order of the swaps depends on the specific strategic used for merging.

○ Merge sort does not use swap operations on the given array.

○ Merge sort does not use swap operations on any array.

○ None of the above.



Save Answer

Q3 Asymptotic Runtime Analysis

850 Points

Q3.1 f1

100 Points

What is the runtime of the function below?

```
public void f1(int N) {
    for (int i = 1; i < N; i = i + 2) {
        for (int j = 1; j < N; j = j * 3) {
            System.out.println("*");
        }
    }
}
```

○ $\theta(1)$

○ $\theta(\log N)$

○ $\theta(N)$

○ $\theta(N \log N)$

○ $\theta(N^2)$

○ $\theta(N^2 \log N)$

○ $\theta(2^N)$

○ $\theta(2^N \log N)$

○ $\theta(3^N)$

○ $\theta(3^N \log N)$

Save Answer

Q3.2 f2

200 Points

Let C(N) be the number of print statements for the function `f2` below. What is C(N)? You may assume N is a power of 2.

```
public void f2(int N) {
    System.out.println("x");
    int half = N / 2;
    if (N > 1) {
        f2(half);
        f2(half);
        f2(half);
        f2(half);
    }
}
```

○ $C(N) = 1$

○ $C(N) = 1 + 2 + 3 + 4 + ... + N$

○ $C(N) = 1 + 4 + 8 + 16 + ... + N$

○ $C(N) = 1 + 4 + 8 + 16 + ... + N^2$

○ $C(N) = 1 + 4 + 8 + 16 + ... + 2^N$

○ $C(N) = 1 + 4 + 8 + 16 + ... + 4^N$

○ $C(N) = 1 + 4 + 16 + 64 + ... + N$

○ $C(N) = 1 + 4 + 16 + 64 + ... + N^2$

○ $C(N) = 1 + 4 + 16 + 64 + ... + 2^N$

○ $C(N) = 1 + 4 + 16 + 64 + ... + 4^N$

○ $C(N) = 1 + N + N^2 + N^3 + ... + N^N$

Save Answer

Q3.3 f3

200 Points

What is the runtime of f3(N)? You may assume N is a power of 2.

```
public void f3(int N) {
    for (int i = 0; i < N; i += 1) {
        System.out.print("*");
    }
    int half = N / 2;
    if (N > 1) {
        f3(half);
        f3(half);
        f3(half);
        f3(half);
    }
}
```

○ $\theta(1)$

○ $\theta(\log N)$

○ $\theta(N)$

○ $\theta(N \log N)$

○ $\theta(N^2)$

○ $\theta(N^2 \log N)$

○ $\theta(2^N)$

○ $\theta(2^N \log N)$

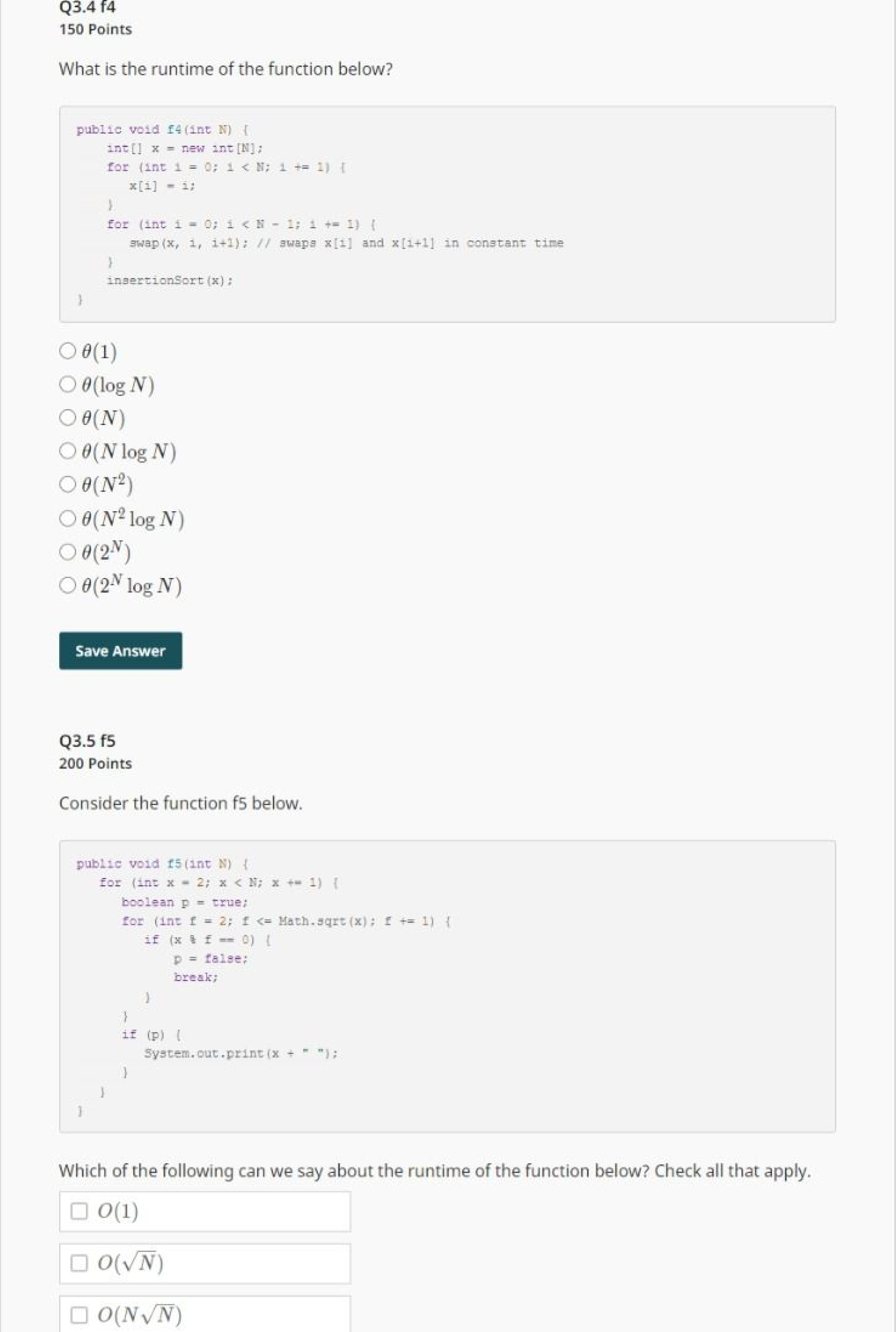
○ $\theta(3^N)$

○ $\theta(3^N \log N)$

○ $\theta(4^N)$

○ $\theta(4^N \log N)$

Save Answer



Q3.4 f4

150 Points

What is the runtime of the function below?

```
public void f4(int N) {
    int[] x = new int[N];
    for (int i = 0; i < N; i += 1) {
      x[i] = i;
    }
    for (int i = 0; i < N - 1; i += 1) {
      swap(x, i, i+1); // swaps x[i] and x[i+1] in constant time
    }
    insertionSort(x);
}
```

○ $\theta(1)$

○ $\theta(\log N)$

○ $\theta(N)$

○ $\theta(N \log N)$

○ $\theta(N^2)$

○ $\theta(N^2 \log N)$

○ $\theta(2^N)$

○ $\theta(2^N \log N)$

Q3.5 f5

200 Points

Consider the function f5 below.

```
public void f5(int N) {
   for (int x = 2; x < N; x += 1) {
      boolean p = true;
      for (int f = 2; f <= Math.sqrt(x); f += 1) {
         if (x % f == 0) {
            p = false;
            break;
         }
      }
      if (p) {
         System.out.print(x + " ");
      }
   }
}
```

Which of the following can we say about the runtime of the function below? Check all that apply.

□ $O(1)$

□ $O(\sqrt{N})$

□ $O(N\sqrt{N})$

□ $O(N^2\sqrt{N})$

□ $O(N!)$

Save Answer

Q3.6 What does f5 do?
0 Points

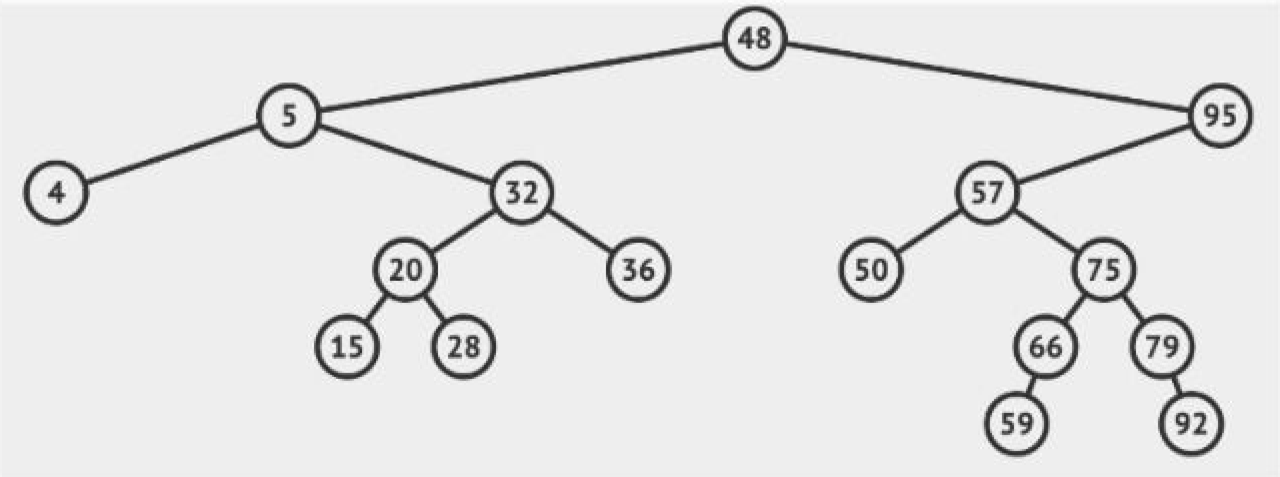
Not for credit, what does f5 do? This problem will not be graded.

Enter your answer here

Save Answer

Q4 BSTs
500 Points

Suppose we have the BST shown below. This is a standard BST, with no special balancing operations.



Q4.1 add(58)
100 Points

If we add 58 to the BST, where will it end up?

○ The insertion will fail.

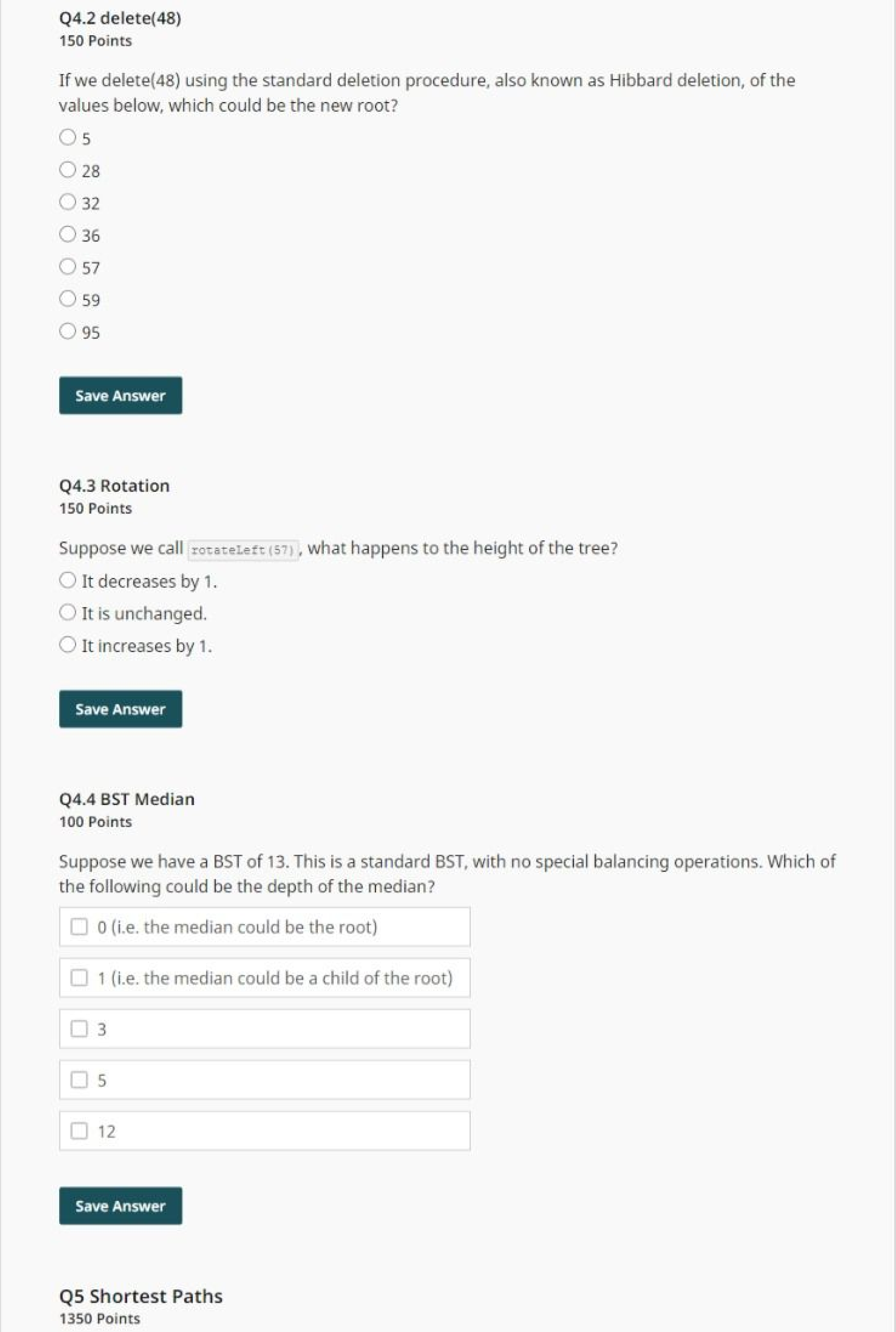
○ As the right child of 36.

○ As the right child of 50.

○ As the right child of 57.

○ As the left child of 59.

Save Answer



Q4.2 delete(48)

150 Points

If we delete(48) using the standard deletion procedure, also known as Hibbard deletion, of the values below, which could be the new root?

○ 5

○ 28

○ 32

○ 36

○ 57

○ 59

○ 95

Q4.3 Rotation

150 Points

Suppose we call `rotateLeft(57)`, what happens to the height of the tree?

○ It decreases by 1.

○ It is unchanged.

○ It increases by 1.

Q4.4 BST Median

100 Points

Suppose we have a BST of 13. This is a standard BST, with no special balancing operations. Which of the following could be the depth of the median?

□ 0 (i.e. the median could be the root)

□ 1 (i.e. the median could be a child of the root)

□ 3

□ 5

□ 12

Q5 Shortest Paths

1350 Points

Q5.1 Input Differences

100 Points

To run, Dijkstra's needs a source vertex and a graph. It returns a correct shortest paths tree.

What does A* need in order to run? Check all that apply.

□ A graph

□ A source vertex

□ A target vertex

□ A list of disallowed vertices

□ A shortest paths tree

□ A heuristic

Save Answer

Q5.2 A* Output

50 Points

Suppose we try to get driving directions from Chicago to New York City using A*. Which can A* return? Assume nothing about `h(v)`.

□ A correct shortest paths tree to every city from Chicago.

□ A incorrect shortest paths tree to every city from Chicago. That is, every such path is suboptimal.

□ A correct shortest path from Chicago to New York.

□ An incorrect shortest path from Chicago to New York.

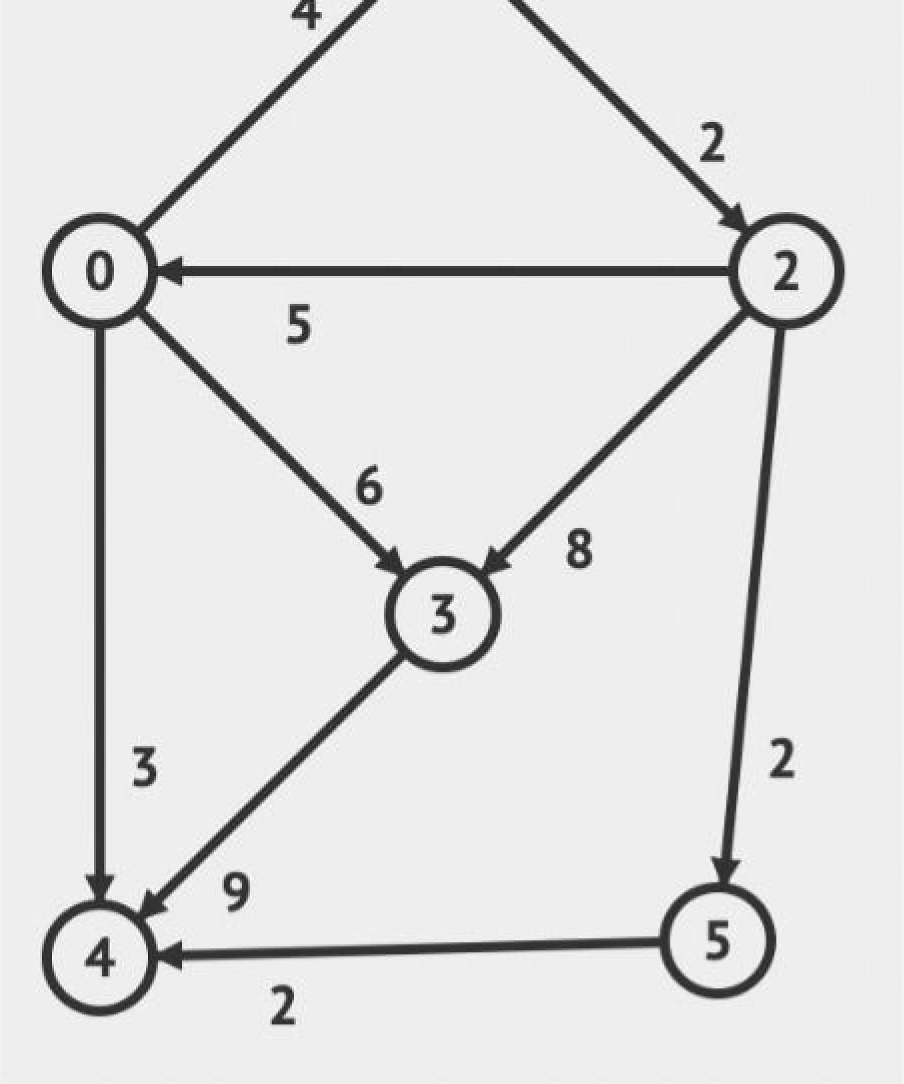
□ None of the above

Save Answer

Q5.3 Dijkstra's Order

200 Points

Suppose we run Dijkstra's on the graph below **starting from vertex 2**. In what order will the vertices be removed from the priority queue?



○ 0, 1, 2, 3, 4, 5

○ 0, 1, 2, 5, 4, 3

○ 0, 4, 1, 3, 5, 2

○ 2, 0, 4, 1, 3, 5

○ 2, 0, 3, 1, 5, 4

○ 2, 5, 4, 0, 3, 1

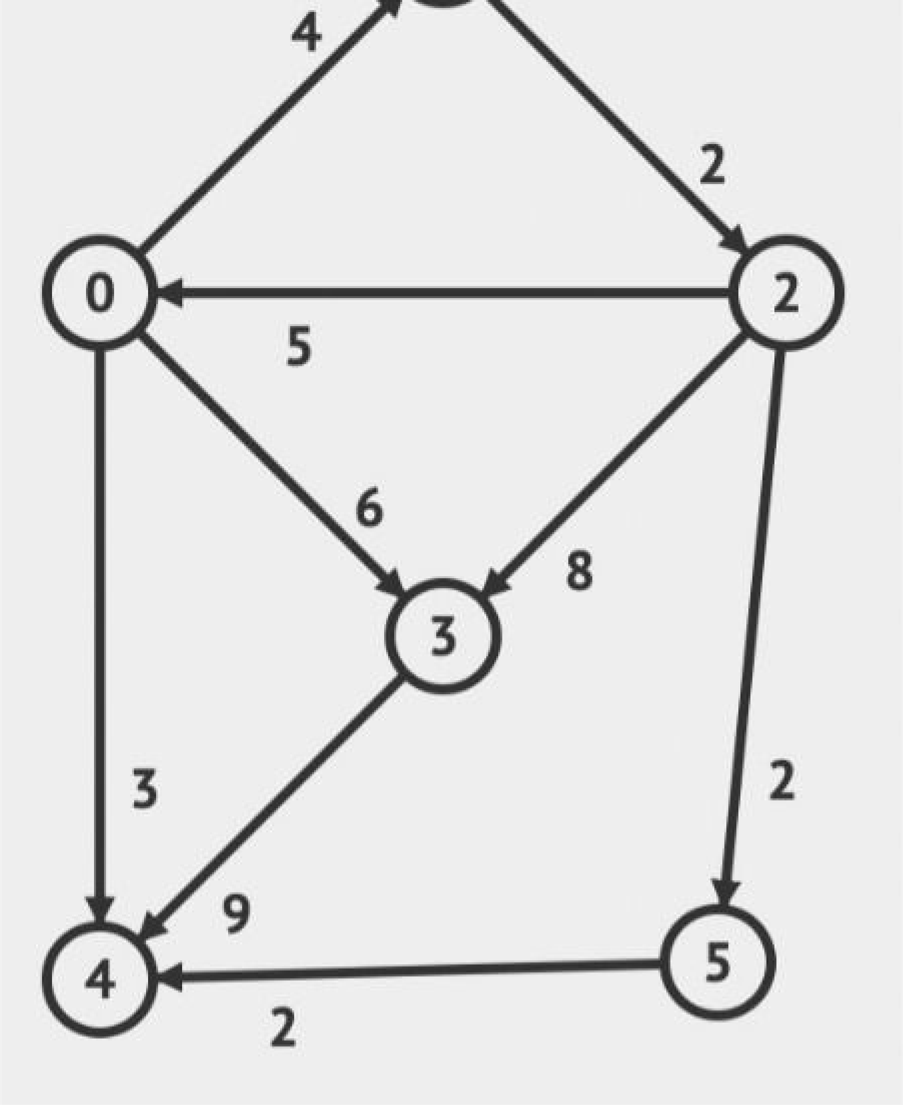
Save Answer

Q5.4 BFS
150 Points

Suppose we run BFS from vertex 2. Which vertices could be visited **last**? Assume vertices are added to the queue in an arbitrary order, i.e. we don't know the order that each neighbor appears in a node's adjacency list.

Recall that after a vertex is visited (i.e. marked), it is never re-added to the fringe.

For convenience, we show the graph again below:



□ 0

□ 1

□ 2

□ 3

□ 4

□ 5

Save Answer

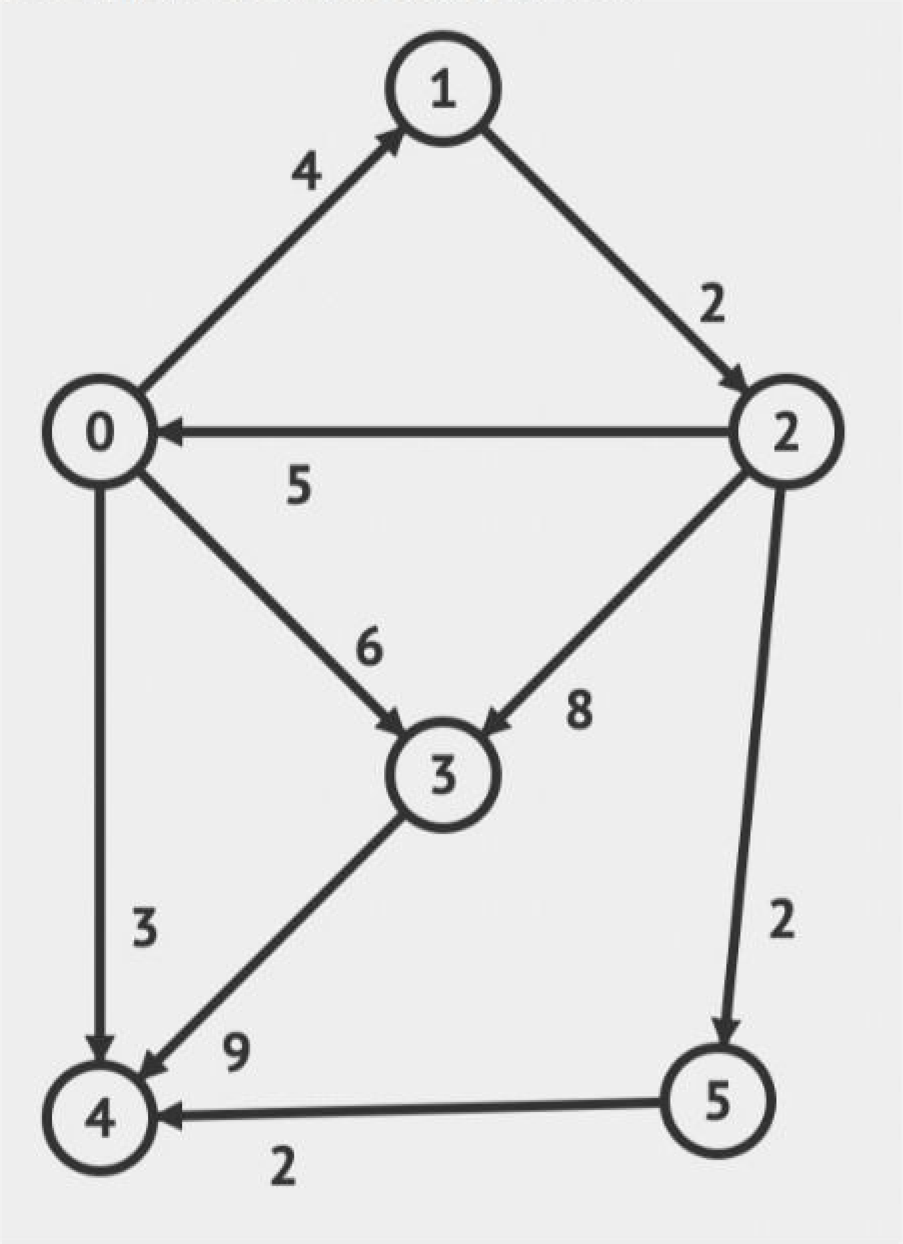
Q5.5 Bizarro Stack BFS

250 Points

Suppose we run BFS, but modified to use a Stack instead of a Queue for storing the fringe. Suppose we start from 2. Which vertices could be visited **last**? Assume vertices are added to the stack in an arbitrary order, i.e. we don't know the order that each neighbor appears in a node's adjacency list.

Recall that after a vertex is visited (i.e. marked), it is never re-added to the fringe.

For convenience, we show the graph again below:



☐ 0

☐ 1

☐ 2

☐ 3

☐ 4

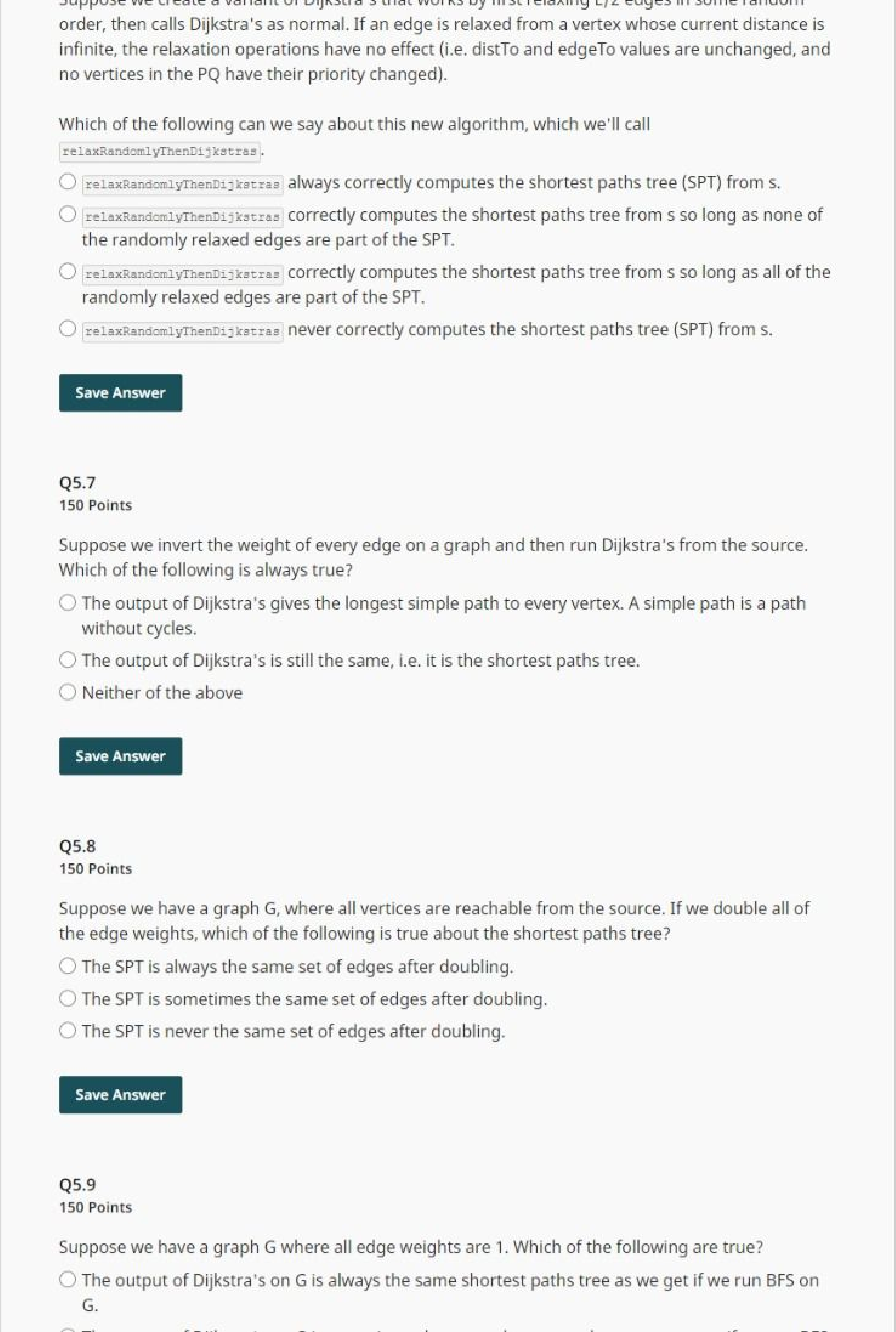
☐ 5

Save Answer

Q5.6 relaxRandomlyThenDijkstras

150 Points

Suppose we create a variant of Dijkstra's that works by first relaxing E/2 edges in some random order, then calls Dijkstra's as normal. If an edge is relaxed from a vertex whose current distance is



Suppose we create a variant of Dijkstra's that works by first relaxing E/2 edges in some random order, then calls Dijkstra's as normal. If an edge is relaxed from a vertex whose current distance is infinite, the relaxation operations have no effect (i.e. distTo and edgeTo values are unchanged, and no vertices in the PQ have their priority changed).

Which of the following can we say about this new algorithm, which we'll call `relaxRandomlyThenDijkstras`.

○ `relaxRandomlyThenDijkstras` always correctly computes the shortest paths tree (SPT) from s.

○ `relaxRandomlyThenDijkstras` correctly computes the shortest paths tree from s so long as none of the randomly relaxed edges are part of the SPT.

○ `relaxRandomlyThenDijkstras` correctly computes the shortest paths tree from s so long as all of the randomly relaxed edges are part of the SPT.

○ `relaxRandomlyThenDijkstras` never correctly computes the shortest paths tree (SPT) from s.

Q5.7
150 Points

Suppose we invert the weight of every edge on a graph and then run Dijkstra's from the source. Which of the following is always true?

○ The output of Dijkstra's gives the longest simple path to every vertex. A simple path is a path without cycles.

○ The output of Dijkstra's is still the same, i.e. it is the shortest paths tree.

○ Neither of the above

Q5.8
150 Points

Suppose we have a graph G, where all vertices are reachable from the source. If we double all of the edge weights, which of the following is true about the shortest paths tree?

○ The SPT is always the same set of edges after doubling.

○ The SPT is sometimes the same set of edges after doubling.

○ The SPT is never the same set of edges after doubling.

Q5.9
150 Points

Suppose we have a graph G where all edge weights are 1. Which of the following are true?

○ The output of Dijkstra's on G is always the same shortest paths tree as we get if we run BFS on G.

○ The output of Dijkstra's on G is sometimes the same shortest paths tree as we get if we run BFS on G.

○ The output of Dijkstra's on G is never the same shortest paths tree as we get if we run BFS on G.

Save Answer

Q5.10 Ocean Navigation

0 Points

During the Age of Sail (say, roughly the year 1600), navigators in the Northern Hemisphere had a somewhat easier time than navigators in the Southern Hemisphere. Why?

Enter your answer here

Save Answer

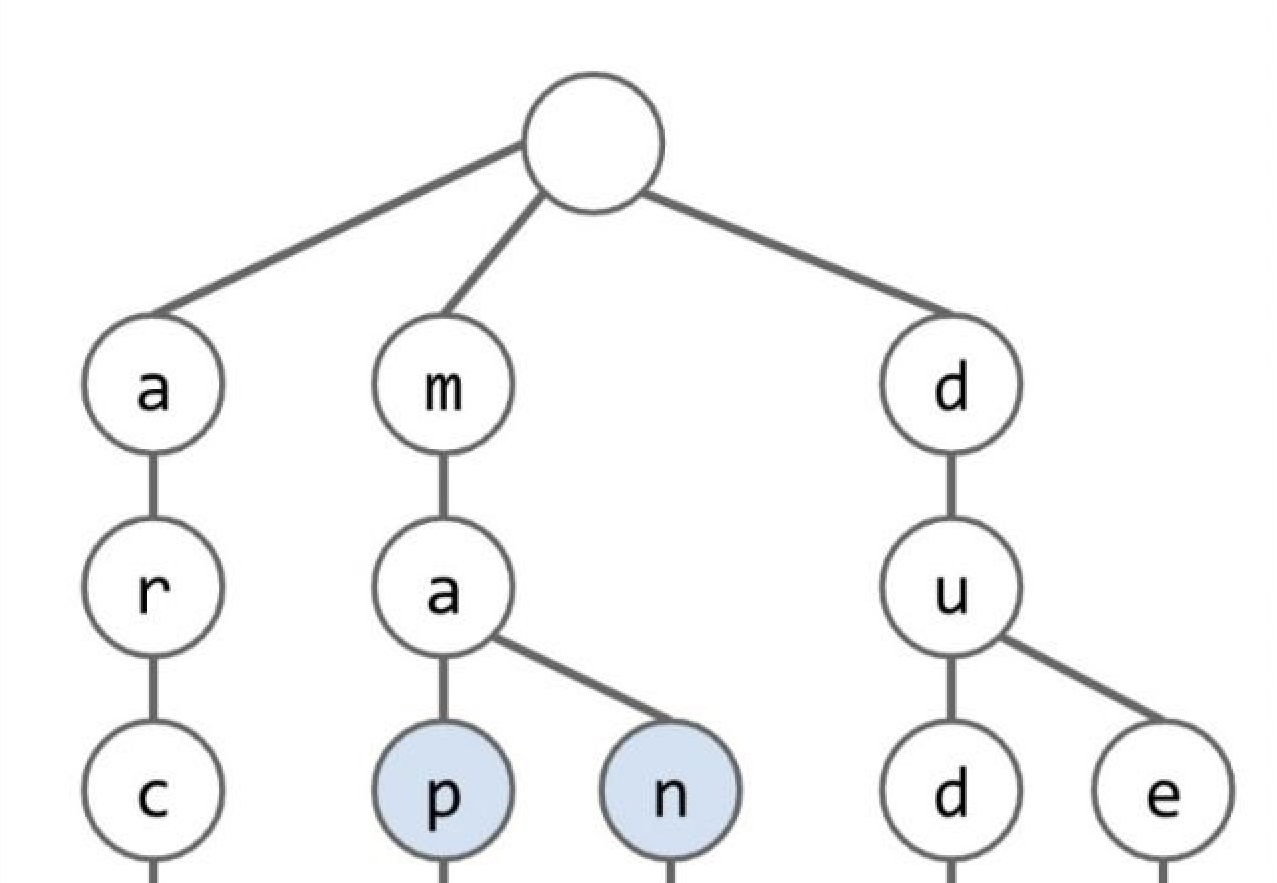
Q6 Tries

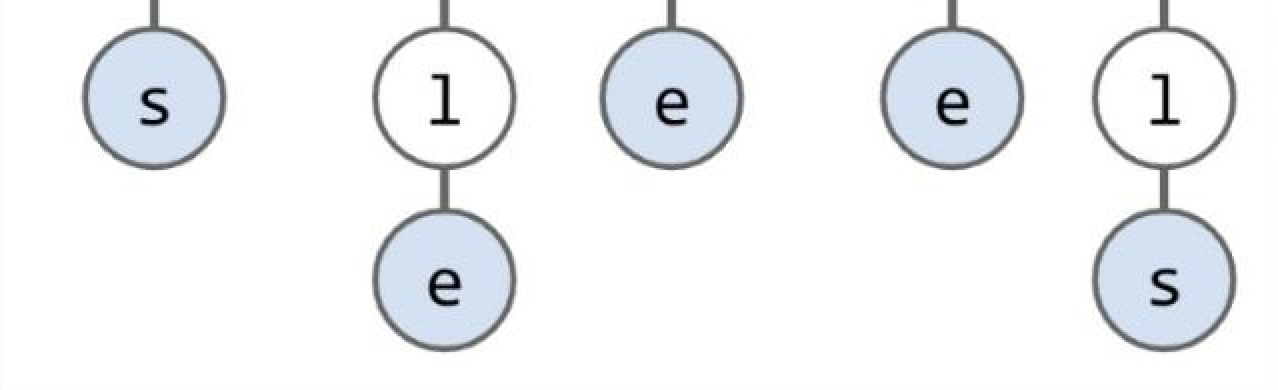
400 Points

Q6.1 keysWithPrefix

100 Points

Suppose we have the trie shown below. Recall that `isKey` is `true` for blue nodes and `isKey` is `false` for white nodes.





If we call `keysWithPrefix("ma")`, which keys will be returned. Recall that `keysWithPrefix` finds all keys with the given prefix.

□ p

□ e

□ n

□ ma

□ map

□ mapl

□ maple

□ man

□ mane

Save Answer

Q6.2
100 Points

What key, when added, creates the **fewest** new nodes? If multiple keys produce the minimum, select all that apply.

□ arck

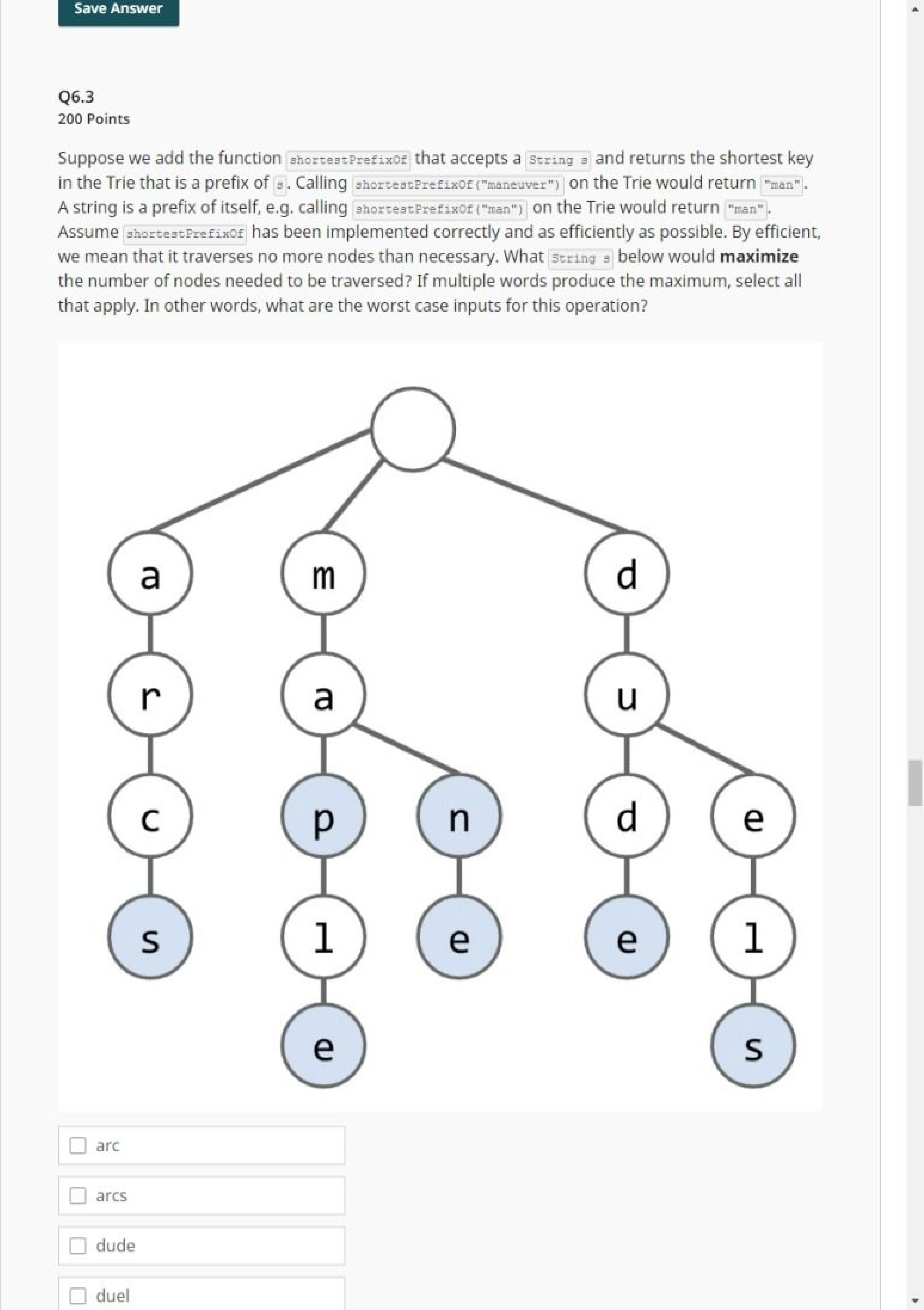
□ at

□ of

□ maples

□ marg

□ marge



Q6.3
200 Points

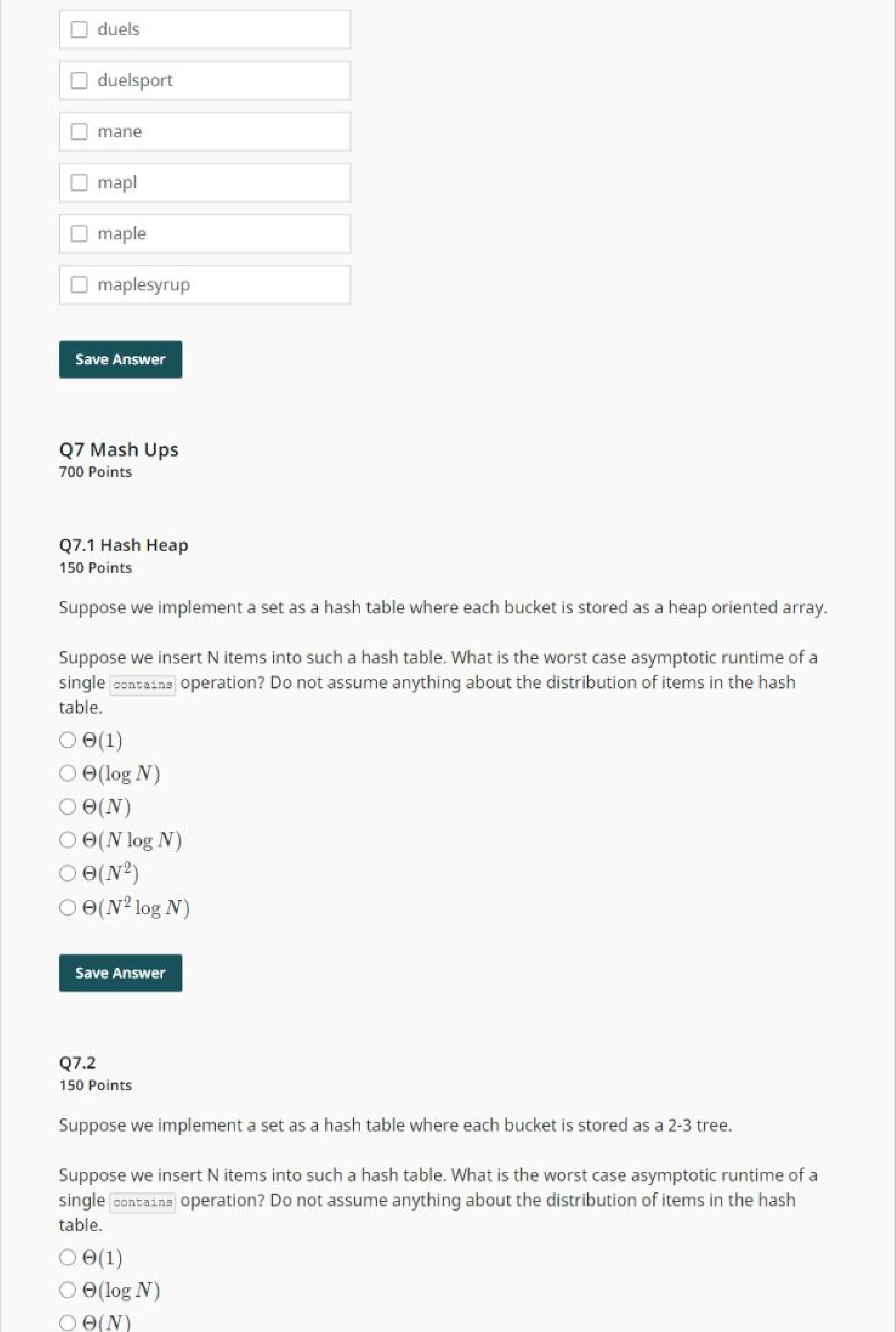
Suppose we add the function `shortestPrefixOf` that accepts a `String s` and returns the shortest key in the Trie that is a prefix of `s`. Calling `shortestPrefixOf("maneuver")` on the Trie would return `"man"`. A string is a prefix of itself, e.g. calling `shortestPrefixOf("man")` on the Trie would return `"man"`. Assume `shortestPrefixOf` has been implemented correctly and as efficiently as possible. By efficient, we mean that it traverses no more nodes than necessary. What `String s` below would **maximize** the number of nodes needed to be traversed? If multiple words produce the maximum, select all that apply. In other words, what are the worst case inputs for this operation?

□ arc

□ arcs

□ dude

□ duel



□ duels

□ duelsport

□ mane

□ mapl

□ maple

□ maplesyrup

Q7 Mash Ups

700 Points

Q7.1 Hash Heap

150 Points

Suppose we implement a set as a hash table where each bucket is stored as a heap oriented array.

Suppose we insert N items into such a hash table. What is the worst case asymptotic runtime of a single `contains` operation? Do not assume anything about the distribution of items in the hash table.

○ $\Theta(1)$

○ $\Theta(\log N)$

○ $\Theta(N)$

○ $\Theta(N \log N)$

○ $\Theta(N^2)$

○ $\Theta(N^2 \log N)$

Q7.2

150 Points

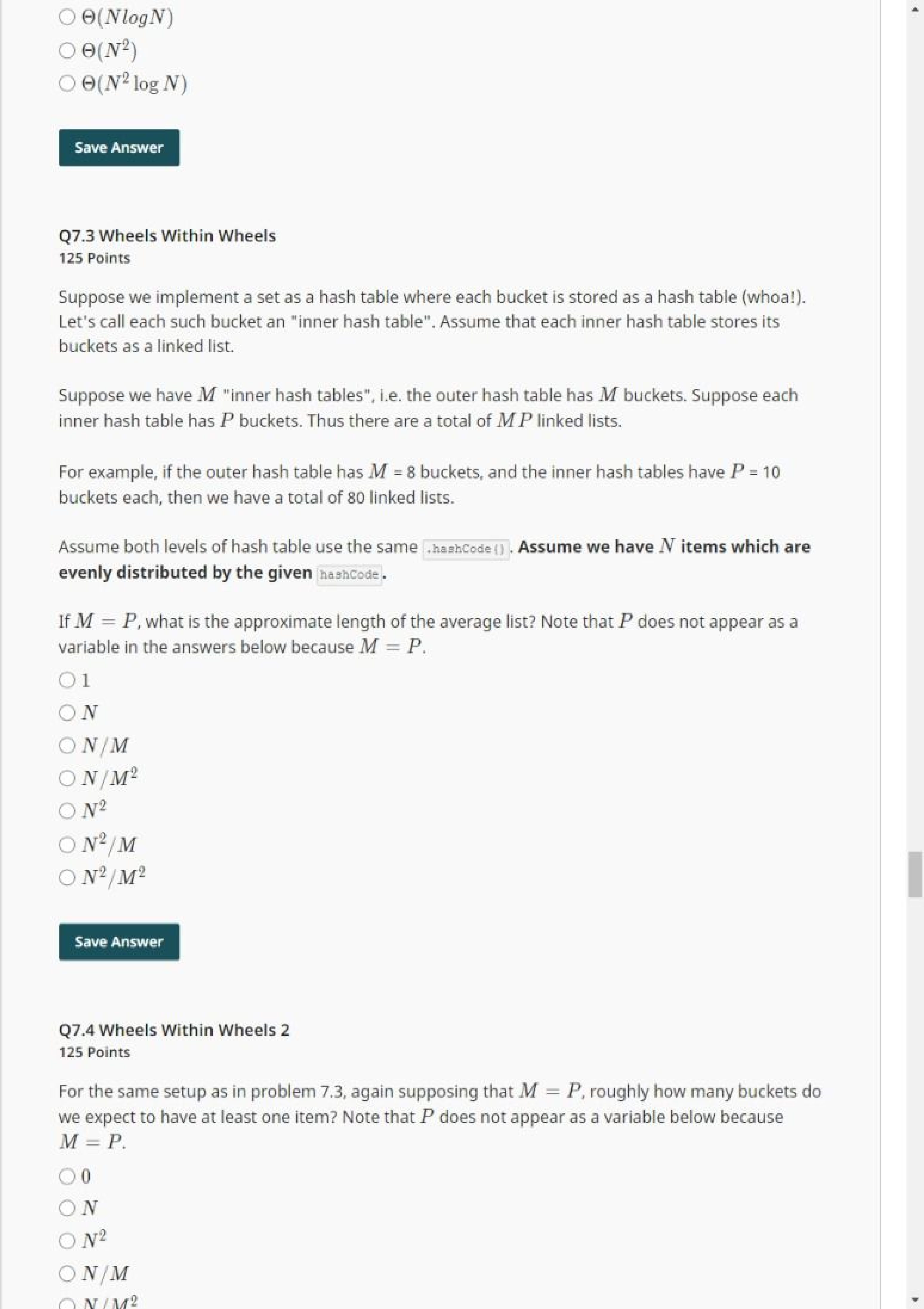
Suppose we implement a set as a hash table where each bucket is stored as a 2-3 tree.

Suppose we insert N items into such a hash table. What is the worst case asymptotic runtime of a single `contains` operation? Do not assume anything about the distribution of items in the hash table.

○ $\Theta(1)$

○ $\Theta(\log N)$

○ $\Theta(N)$



○ $\Theta(NlogN)$

○ $\Theta(N^2)$

○ $\Theta(N^2 \log N)$

Save Answer

Q7.3 Wheels Within Wheels

125 Points

Suppose we implement a set as a hash table where each bucket is stored as a hash table (whoa!). Let's call each such bucket an "inner hash table". Assume that each inner hash table stores its buckets as a linked list.

Suppose we have M "inner hash tables", i.e. the outer hash table has M buckets. Suppose each inner hash table has P buckets. Thus there are a total of MP linked lists.

For example, if the outer hash table has M = 8 buckets, and the inner hash tables have P = 10 buckets each, then we have a total of 80 linked lists.

Assume both levels of hash table use the same `.hashCode()`. **Assume we have N items which are evenly distributed by the given `hashCode`.**

If $M = P$, what is the approximate length of the average list? Note that P does not appear as a variable in the answers below because $M = P$.

○ 1

○ N

○ N/M

○ N/M^2

○ N^2

○ N^2/M

○ N^2/M^2

Save Answer

Q7.4 Wheels Within Wheels 2

125 Points

For the same setup as in problem 7.3, again supposing that $M = P$, roughly how many buckets do we expect to have at least one item? Note that P does not appear as a variable below because $M = P$.

○ 0

○ N

○ N^2

○ N/M

○ N/M^2

○ N^2/M

○ N^2/M^2

○ M

○ M^2

Save Answer

Q7.5 RLBST
150 Points

Suppose we create a right-leaning binary search tree (RLBST), i.e. 2-3 nodes are represented by a right leaning glue node.

Suppose we insert the number 5 then 3 into an RLBST, what operation should we call to maintain our invariant?

○ rotateLeft(3)

○ rotateRight(3)

○ rotateLeft(5)

○ rotateRight(5)

○ colorFlip(3)

○ colorFlip(5)

○ No operation necessary

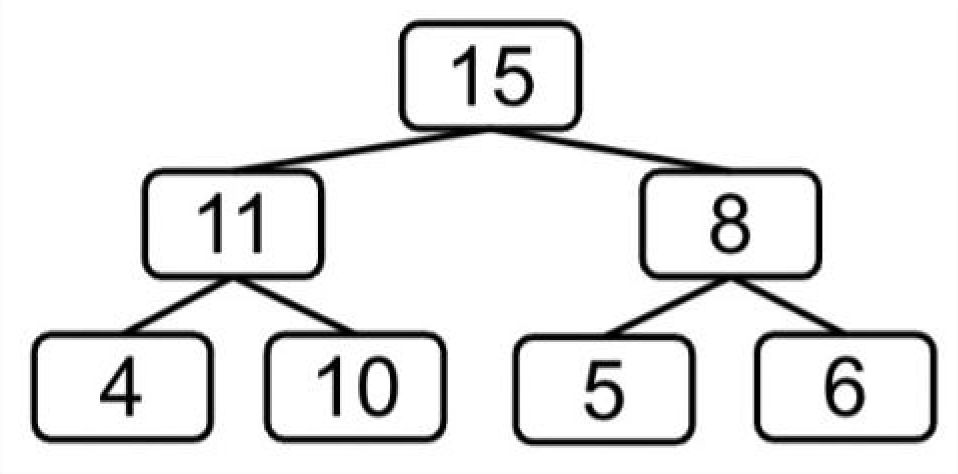
Save Answer

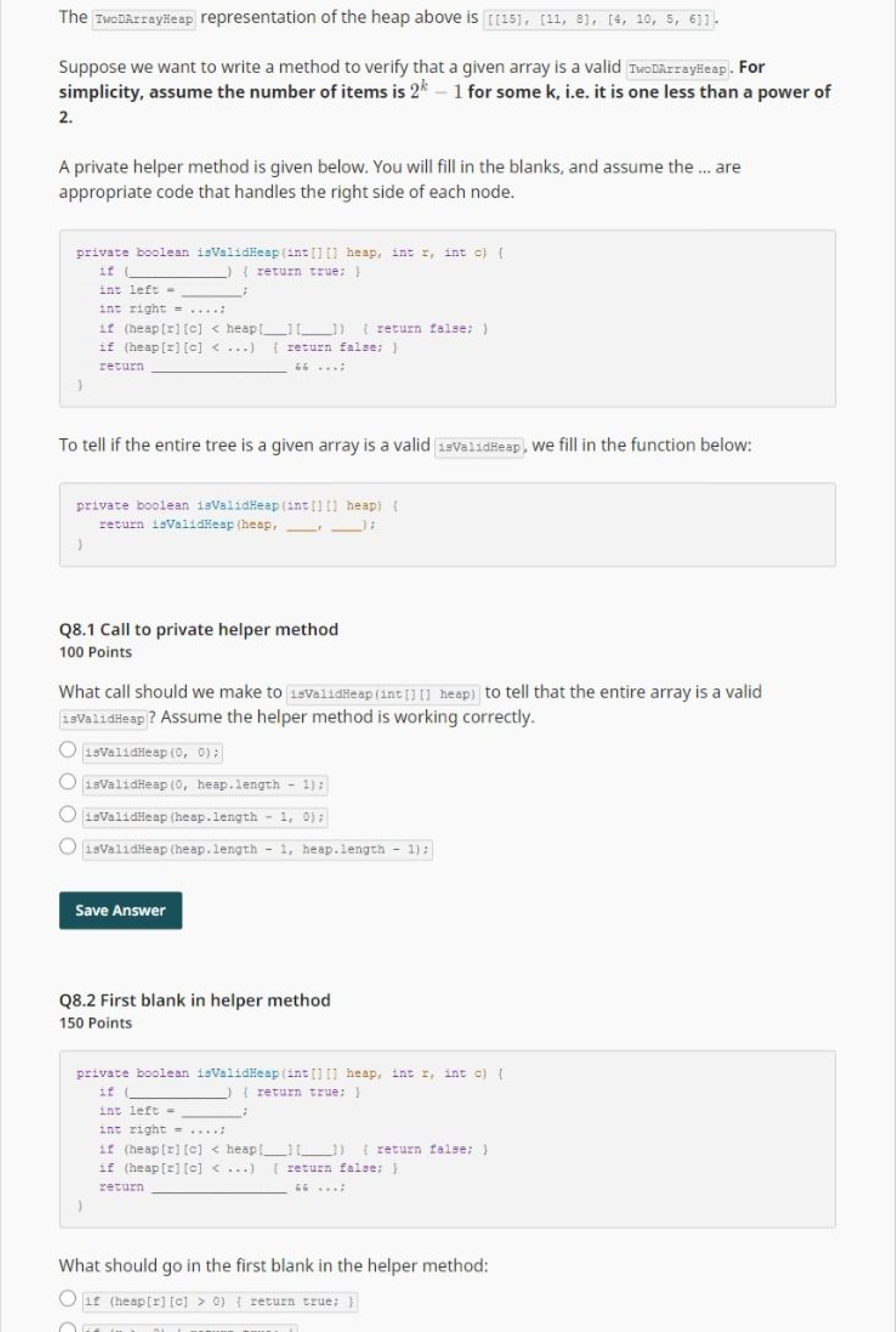
Q8 Heap Verification
700 Points

In class we talked about many possible representations of a tree.

Suppose we use a new representation of a complete tree which we'll call a `TwoDArrayHeap`.

As an example, consider the heap below.





The `TwoDArrayHeap` representation of the heap above is `[[15], [11, 8], [4, 10, 5, 6]]`.

Suppose we want to write a method to verify that a given array is a valid `TwoDArrayHeap`. **For simplicity, assume the number of items is $2^k - 1$ for some k, i.e. it is one less than a power of 2.**

A private helper method is given below. You will fill in the blanks, and assume the ... are appropriate code that handles the right side of each node.

```
private boolean isValidHeap(int[][] heap, int r, int c) {
   if (____________) { return true; }
   int left = ________;
   int right = ....;
   if (heap[r][c] < heap[___][____]) { return false; }
   if (heap[r][c] < ...) { return false; }
   return _________________ && ...;
}
```

To tell if the entire tree is a given array is a valid `isValidHeap`, we fill in the function below:

```
private boolean isValidHeap(int[][] heap) {
   return isValidHeap(heap, ____, ____);
}
```

Q8.1 Call to private helper method

100 Points

What call should we make to `isValidHeap(int[][] heap)` to tell that the entire array is a valid `isValidHeap`? Assume the helper method is working correctly.

○ `isValidHeap(0, 0);`

○ `isValidHeap(0, heap.length - 1);`

○ `isValidHeap(heap.length - 1, 0);`

○ `isValidHeap(heap.length - 1, heap.length - 1);`

Save Answer

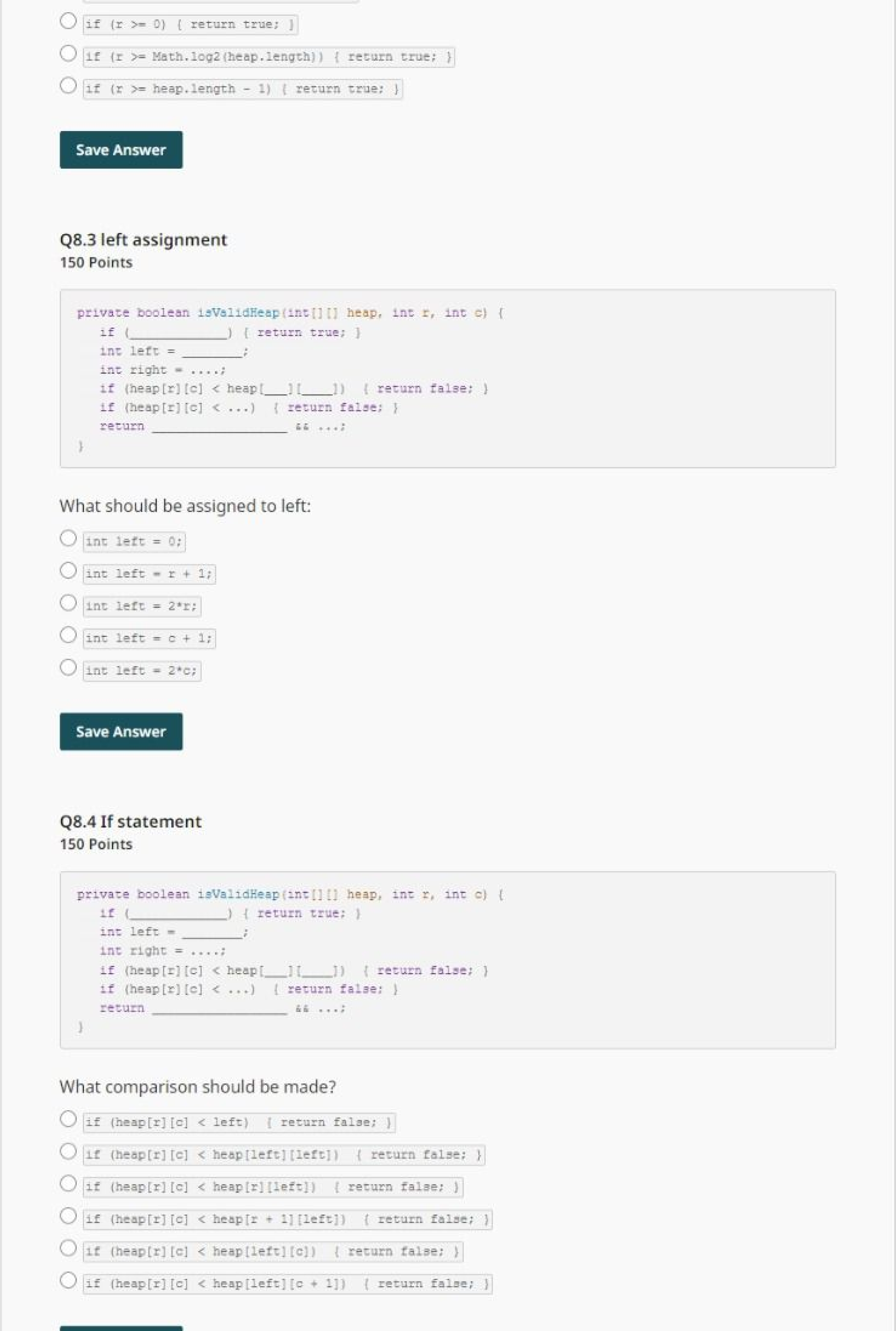
Q8.2 First blank in helper method

150 Points

```
private boolean isValidHeap(int[][] heap, int r, int c) {
   if (____________) { return true; }
   int left = ________;
   int right = ....;
   if (heap[r][c] < heap[___][____]) { return false; }
   if (heap[r][c] < ...) { return false; }
   return _________________ && ...;
}
```

What should go in the first blank in the helper method:

○ `if (heap[r][c] > 0) { return true; }`



○ `if (r >= 0) { return true; }`

○ `if (r >= Math.log2(heap.length)) { return true; }`

○ `if (r >= heap.length - 1) { return true; }`

Q8.3 left assignment

150 Points

```
private boolean isValidHeap(int[][] heap, int r, int c) {
   if (_____________) { return true; }
   int left = ________;
   int right = ....;
   if (heap[r][c] < heap[___][____])  { return false; }
   if (heap[r][c] < ...)  { return false; }
   return __________________ && ...;
}
```

What should be assigned to left:

○ `int left = 0;`

○ `int left = r + 1;`

○ `int left = 2*r;`

○ `int left = c + 1;`

○ `int left = 2*c;`

Q8.4 If statement

150 Points

```
private boolean isValidHeap(int[][] heap, int r, int c) {
   if (_____________) { return true; }
   int left = ________;
   int right = ....;
   if (heap[r][c] < heap[___][____])  { return false; }
   if (heap[r][c] < ...)  { return false; }
   return __________________ && ...;
}
```

What comparison should be made?

○ `if (heap[r][c] < left)  { return false; }`

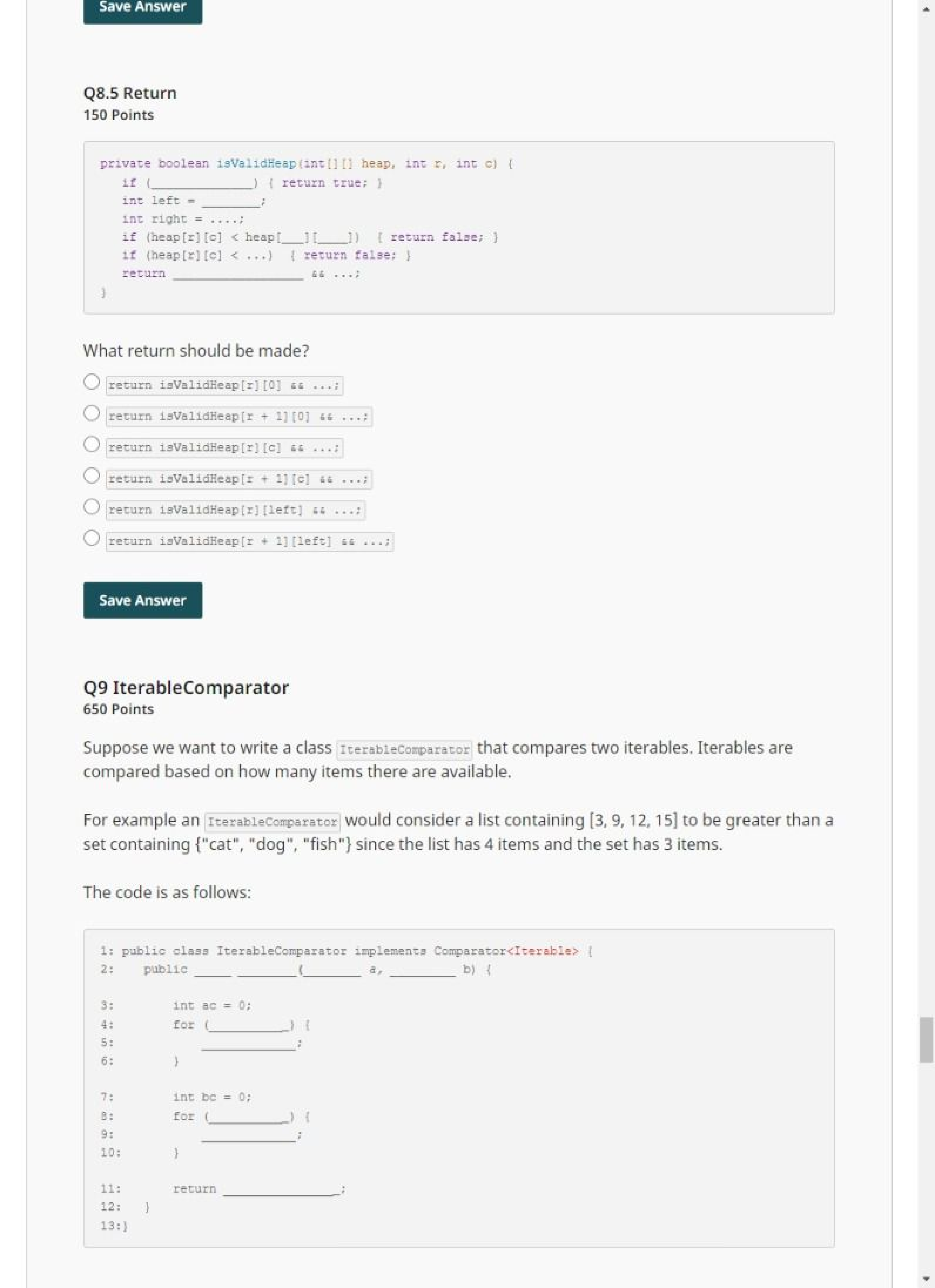
○ `if (heap[r][c] < heap[left][left])  { return false; }`

○ `if (heap[r][c] < heap[r][left])  { return false; }`

○ `if (heap[r][c] < heap[r + 1][left])  { return false; }`

○ `if (heap[r][c] < heap[left][c])  { return false; }`

○ `if (heap[r][c] < heap[left][c + 1])  { return false; }`



Save Answer

Q8.5 Return

150 Points

```
private boolean isValidHeap(int[][] heap, int r, int c) {
   if (______________) { return true; }
   int left = ________;
   int right = ....;
   if (heap[r][c] < heap[___][____])  { return false; }
   if (heap[r][c] < ...)  { return false; }
   return __________________  && ...;
}
```

What return should be made?

○ `return isValidHeap[r][0] && ...;`

○ `return isValidHeap[r + 1][0] && ...;`

○ `return isValidHeap[r][c] && ...;`

○ `return isValidHeap[r + 1][c] && ...;`

○ `return isValidHeap[r][left] && ...;`

○ `return isValidHeap[r + 1][left] && ...;`

Save Answer

Q9 IterableComparator

650 Points

Suppose we want to write a class `IterableComparator` that compares two iterables. Iterables are compared based on how many items there are available.

For example an `IterableComparator` would consider a list containing [3, 9, 12, 15] to be greater than a set containing {"cat", "dog", "fish"} since the list has 4 items and the set has 3 items.

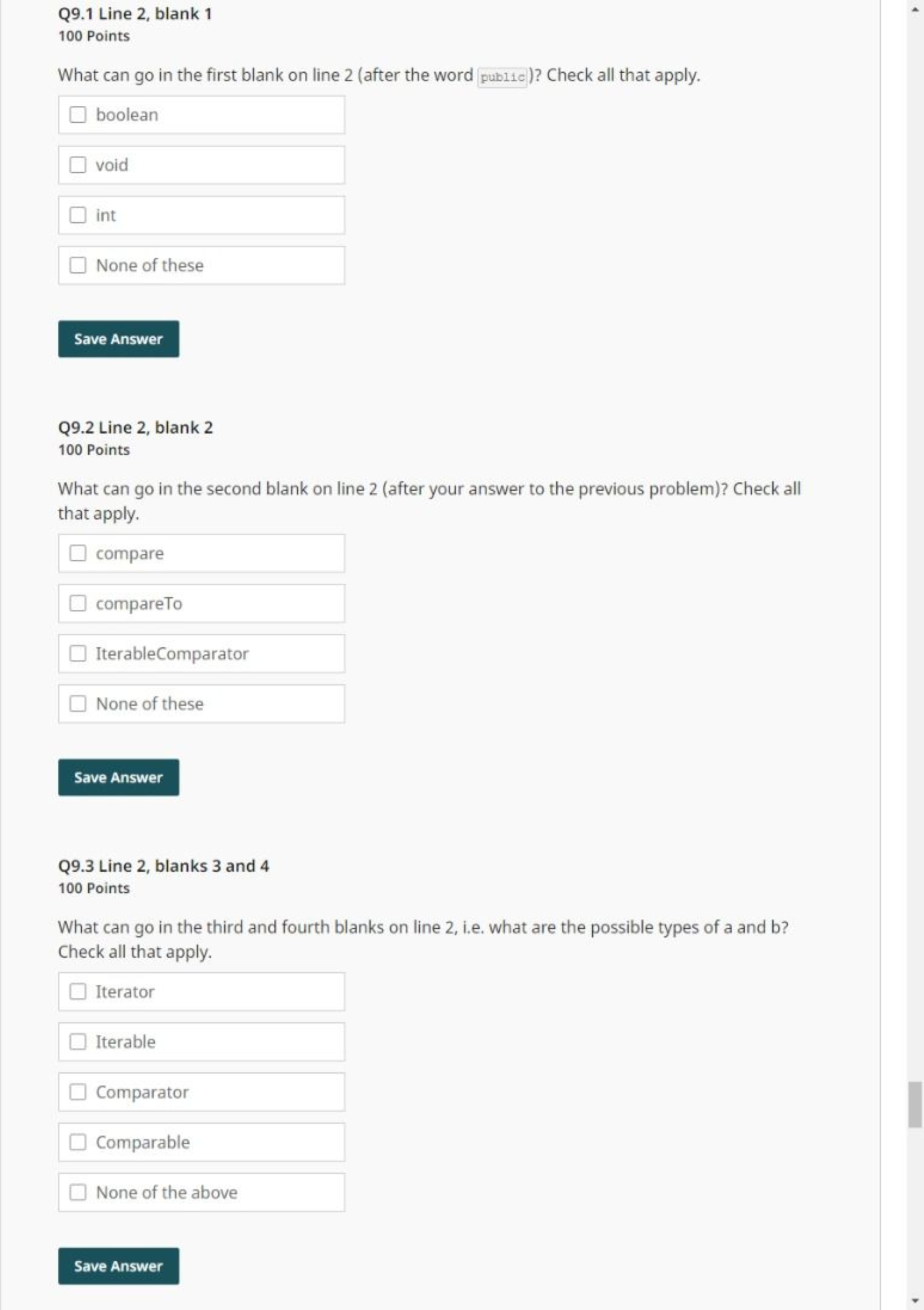
The code is as follows:

```
1: public class IterableComparator implements Comparator<Iterable> {
2:    public _____ ________(________ a, _________ b) {

3:        int ac = 0;
4:        for (__________) {
5:            _____________;
6:        }

7:        int bc = 0;
8:        for (__________) {
9:            _____________;
10:       }

11:       return ________________;
12:   }
13:}
```



Q9.1 Line 2, blank 1

100 Points

What can go in the first blank on line 2 (after the word `public`)? Check all that apply.

□ boolean

□ void

□ int

□ None of these

Q9.2 Line 2, blank 2

100 Points

What can go in the second blank on line 2 (after your answer to the previous problem)? Check all that apply.

□ compare

□ compareTo

□ IterableComparator

□ None of these

Q9.3 Line 2, blanks 3 and 4

100 Points

What can go in the third and fourth blanks on line 2, i.e. what are the possible types of a and b? Check all that apply.

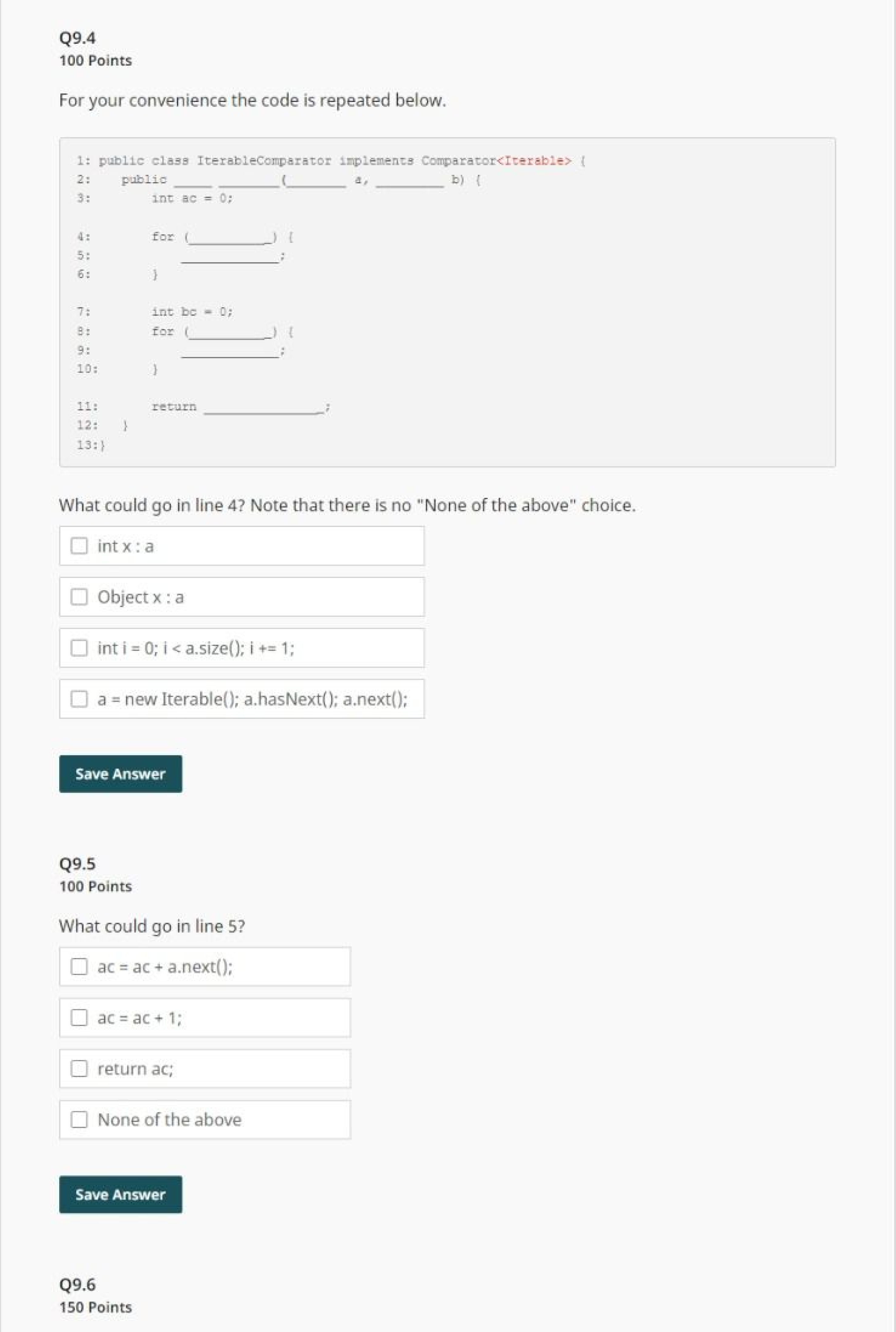
□ Iterator

□ Iterable

□ Comparator

□ Comparable

□ None of the above



Q9.4

100 Points

For your convenience the code is repeated below.

```
1: public class IterableComparator implements Comparator<Iterable> {
2:     public _____ _________(________ a, _________ b) {
3:         int ac = 0;

4:         for (__________) {
5:             _____________;
6:         }

7:         int bc = 0;
8:         for (__________) {
9:             _____________;
10:        }

11:        return _______________;
12:    }
13:}
```

What could go in line 4? Note that there is no "None of the above" choice.

□ int x : a

□ Object x : a

□ int i = 0; i < a.size(); i += 1;

□ a = new Iterable(); a.hasNext(); a.next();

Q9.5

100 Points

What could go in line 5?

□ ac = ac + a.next();

□ ac = ac + 1;

□ return ac;

□ None of the above

Q9.6

150 Points

What could go in line 11 and return the right answer? Assume lines 8 and 9 use the same idea as lines 4 and 5.

□ ac;

□ bc;

□ ac > bc;

□ ac - bc;

□ (ac - bc) * 2;

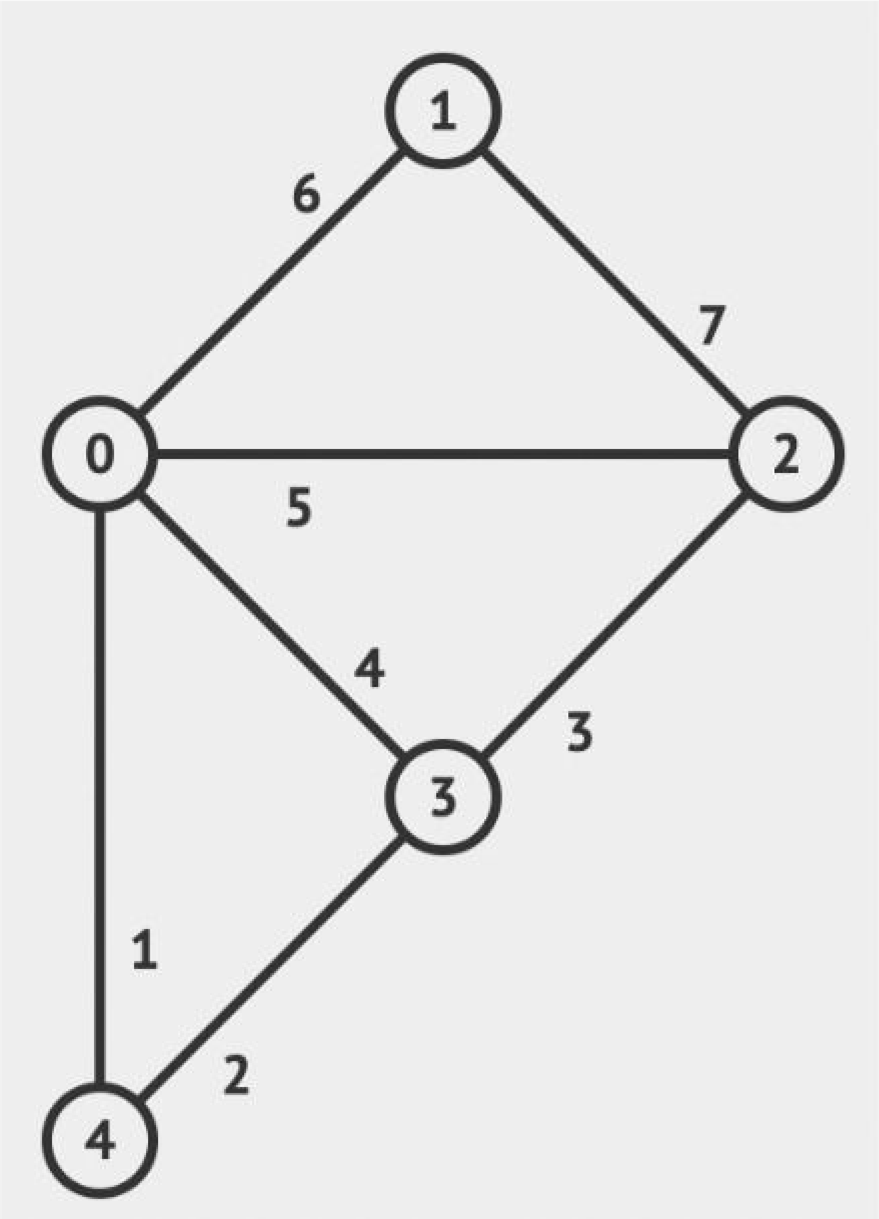
□ None of the above

Save Answer

Q10 MSTs

200 Points

Suppose we have the graph below.



Q10.1 Kruskal's Algorithm

100 Points

Recall that in Kruskal's algorithm, we use a disjoint sets object to check for cycles, making exactly one `isConnected` call per edge.

If we run Kruskal's algorithm, for which edges will `isConnected` return true during these cycle checks?

□ isConnected(0, 4)

□ isConnected(3, 4)

□ isConnected(2, 3)

□ isConnected(0, 3)

□ isConnected(0, 2)

□ isConnected(0, 1)

□ isConnected(1, 2)

Save Answer

Q10.2 Prim's algorithm

100 Points

If we run Prim's from vertex 2, in what order will the vertices be added to the MST?

○ 2, 3, 0, 1, 4

○ 2, 3, 0, 4, 1

○ 2, 3, 1, 0, 4

○ 2, 3, 1, 4, 0

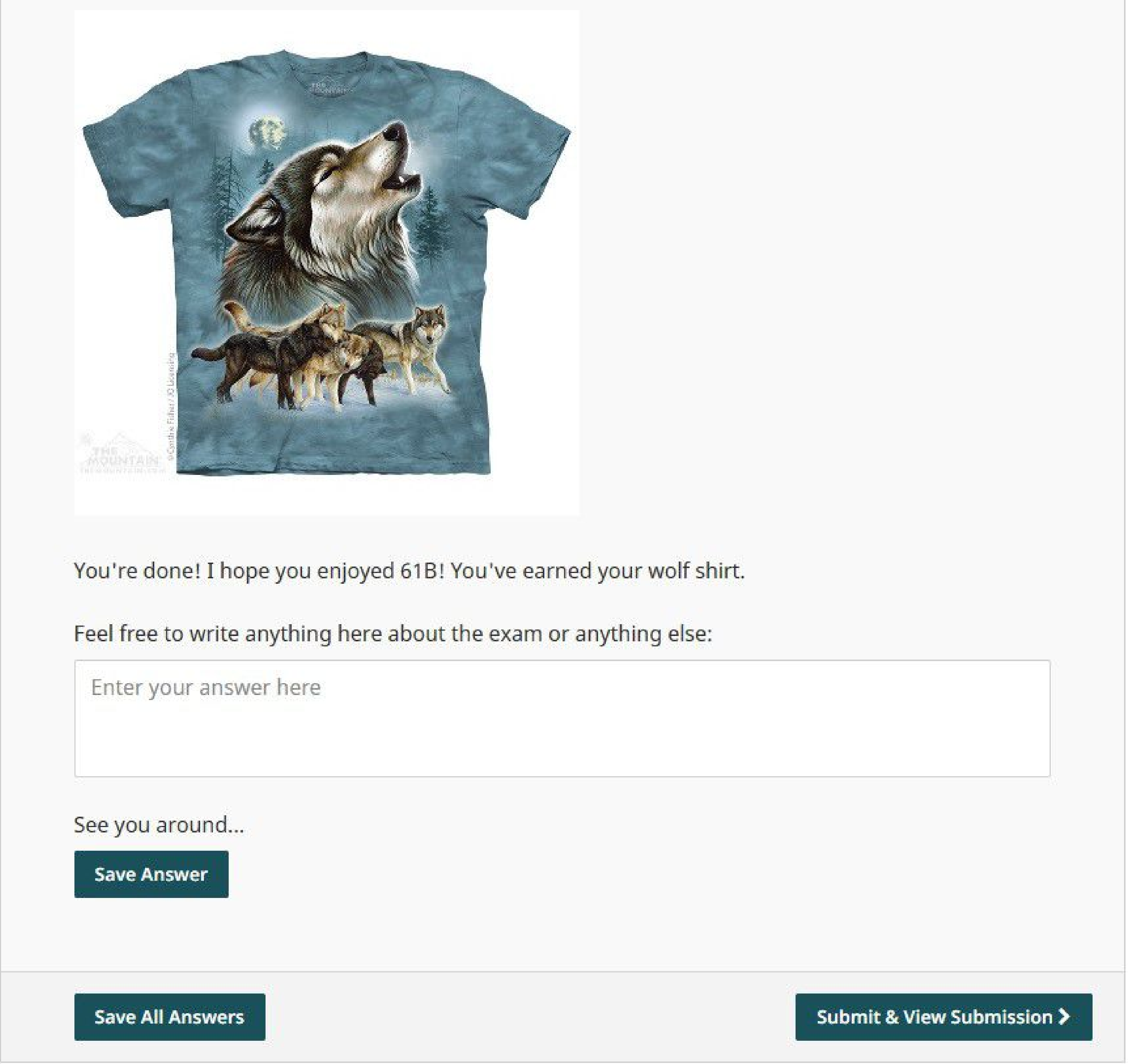
○ 2, 3, 4, 0, 1

○ 2, 3, 4, 1, 0

Save Answer

Q11 Wolf Shirt

0 Points



You're done! I hope you enjoyed 61B! You've earned your wolf shirt.

Feel free to write anything here about the exam or anything else:

See you around...