Data Structures
CSCI C343, Spring 2024 [A]

Final

Name: ______________________

This exam has 12 questions, for a total of 100 points.

1. 8 points The function `sort`, shown below, returns a linked-list that is sorted and that contains the elements from the input linked-list. What is the time complexity of `sort` in terms of the length n of the linked list? Explain your answer.

```
public class Node {
    int data;
    Node next;
    public Node(int d, Node n) { data = d; next = n; }
}
static Node insert(Node ls, int elt) {
    if (ls == null) {
        return new Node(elt, null);
    } else if (elt < ls.data) {
        return new Node(elt, ls);
    } else {
        return new Node(ls.data, insert(ls.next, elt));
    }
}
static Node sort_helper(Node todo, Node sorted) {
    if (todo == null) {
        return sorted;
    } else {
        return sort_helper(todo.next, insert(sorted, todo.data));
    }
}
static Node sort(Node lst) {
    return sort_helper(lst, null);
}
```

Solution: The time complexity of `sort` is $O(n^2)$. To see why, consider the recursive call tree for `sort_helper`. At each level of the tree, the algorithm does $O(n)$ work because of the calls to `insert` **(3 points)**. There are n levels of the tree because the linked list is reduced in size by 1 with each recursive call to `sort_helper` **(3 points)**. Multiplying the time per level times the number of levels yields $O(n^2)$. **(2 points)**

2. [10 points] The `max_heapify` method rearranges the subtree rooted at index `i` so that the subtree satisfies the max-heap property. The children of the node at index `i` must already satisfy the heap property. The `end` parameter is the number of elements in the heap, which is less than or equal to the length of `data`. Fill in the blanks to complete the implementation of `max_heapify`.

```
public class Heap {
    int[] data;

    static int left(int i) { return 2*i + 1; }
    static int right(int i) { return 2*(i + 1); }
    void swap(int i, int j) {
        E tmp = data[i];
        data[i] = data[j];
        data[j] = tmp;
    }
    void max_heapify(int i, int end) {
        int largest = i;
        if (left(i) < end && ___(a)___)
            largest = left(i);
        if (right(i) < end && ___(b)___)
            largest = right(i);
        if (___(c)___) {
            swap(___(d)___, largest);
            max_heapify(___(e)___, end);
        }
    }
}
```

Solution: (2 points each)

```
(a) data[i] < data[l]
(b) data[largest] < data[r]
(c) largest != i
(d) i
(e) largest
```

3. 12 points Recall that the first step of the Quicksort algorithm is to partition the sequence so that all the elements less-or-equal to the pivot appear to the left of the pivot and all the elements greater than the element appear to the right of the pivot. Finish the following generic version of the `partition` function by filling in the blanks. The `partition` function returns the position of the pivot element.

```
public interface Iterator<T> {
    T get();
    void set(T e);
    void advance();
    boolean equals(Iterator<T> other);
    Iterator<T> clone();
}
static <E extends Comparable<? super E>>
Iterator<E> partition(Iterator<E> begin, Iterator<E> end) {
    Iterator<E> last = Algorithms.last(begin, end);
    E pivot = ___(a)___;
    Iterator<E> i = begin.clone();
    for (Iterator<E> j = ___(b)___; ___(c)___; j.advance()) {
        if (j.get().compareTo(___(d)___) < 1) {
            Algorithms.iter_swap(i, j);
            ___(e)___
        }
    }
    Algorithms.iter_swap(last, i);
    return ___(f)___;
}
```

Solution: (2 points each)

```
(a) last.get()
(b) i.clone()  or  begin.clone()
(c) ! j.equals(last)
(d) pivot
(e) i.advance();
(f) i
```

4. $\boxed{\text{9 points}}$ Sort the following array A using the Counting Sort algorithm. Show each major step of the algorithm and the auxiliary arrays.

$$A = [3, 0, 4, 2, 0, 3]$$

Solution: The count for each element is computed and stored in the array C: **(3 points)** (It is OK for a solution to use a longer array C with more zeroes at the end.)

$$C = [2, 0, 1, 2, 1]$$

Then we calculate the cumulative sum: **(3 points)**

$$L = [2, 2, 3, 5, 6]$$

Now we can compute the output array B. Element 3 goes in index $5 - 1 = 4$ of B. We also update L. **(0.5 points)**

$$B = [\_, \_, \_, \_, 3, \_]$$
$$L = [2, 2, 3, 4, 6]$$

Element 0 goes in index $2 - 1 = 1$ of B and we update L. **(0.5 points)**

$$B = [\_, 0, \_, \_, 3, \_]$$
$$L = [1, 2, 3, 4, 6]$$

Element 4 goes in index $6 - 1 = 5$ of B and we update L. **(0.5 points)**

$$B = [\_, 0, \_, \_, 3, 4]$$
$$L = [1, 2, 3, 4, 5]$$

Element 2 goes in index $3 - 1 = 2$ of B and we update L. **(0.5 points)**

$$B = [\_, 0, 2, \_, 3, 4]$$
$$L = [1, 2, 2, 4, 5]$$

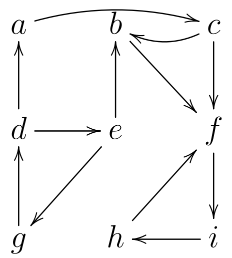
Element 0 goes in index $1 - 1 = 0$ of B and we update L. **(0.5 points)**

$$B = [0, 0, 2, \_, 3, 4]$$
$$L = [0, 2, 2, 4, 5]$$

Element 3 goes in index $4 - 1 = 3$ of B and we update L. **(0.5 points)**

$$B = [0, 0, 2, 3, 3, 4]$$
$$L = [0, 2, 2, 3, 5]$$

5. 6 points Draw the adjacency list representation of the following directed graph.



Solution: (0.5 point per correct edge)

$a \to c$

$b \to f$

$c \to b \to f$

$d \to a \to e$

$e \to b \to g$

$f \to i$

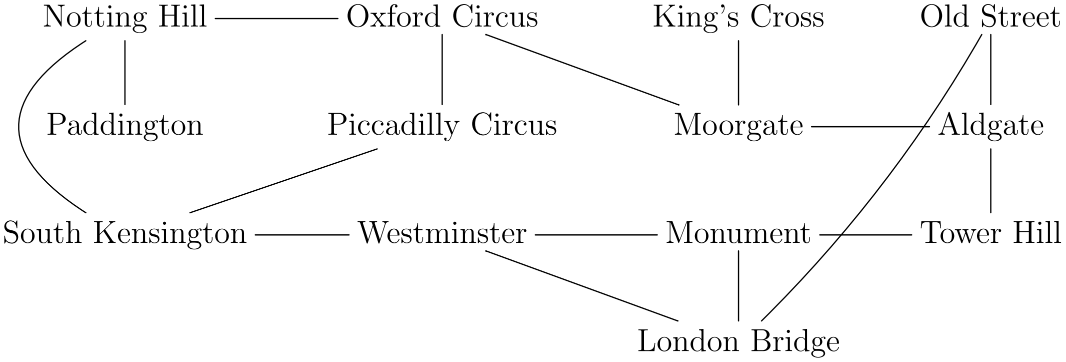
$g \to d$

$h \to f$

$i \to h$

Name: ______________________

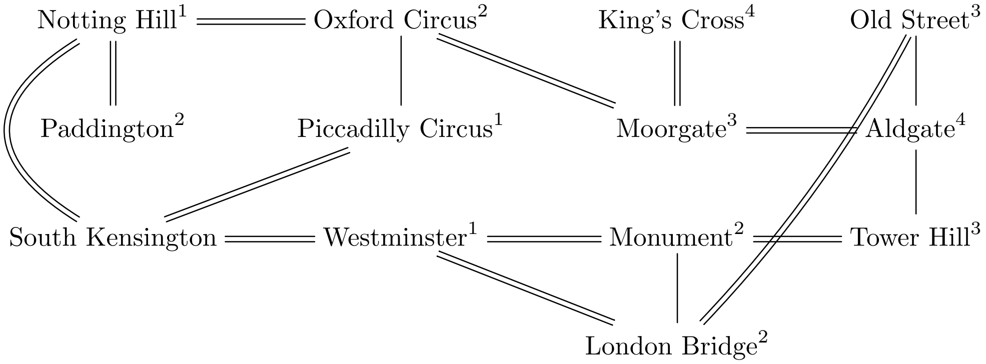
6. 7 points You have recently moved to London and have some errands to accomplish. You'll be taking the subway and need to plan how much money to bring to pay for the transit. It costs £1 for each connection (e.g. South Kensington to Piccadilly Circus). The following is a subway map of London:



Your flat is in South Kensington and you must return to your flat after each individual errand, to drop off your purchases. Here is the location of each errand: King's Cross, Old Street, Paddington, Monument, and Tower Hill.

What is the minimum amout of money needed to pay for the subway to travel on these five errands? What route would you take to each location?

Solution: Perform breadth-first search starting at South Kensington.



The following are shortest routes for each errand. (They may not be unique.) **(1 point each)**

1. South Kensington – Notting Hill – Oxford Circus – Moorgate – King's Cross, £4×2 =£8
2. South Kensington – Westminster – London Bridge – Old Street, £3×2 =£6
3. South Kensington – Notting Hill – Paddington, £2×2 =£4
4. South Kensington – Westminster – Monument, £2×2 =£4
5. South Kensington – Westminster – Monument – Tower Hill, £3×2 =£6

Total: £28 **(2 points)**

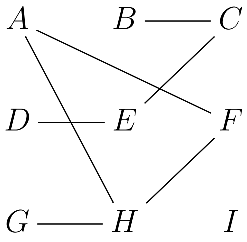
7. [6 points] Suppose the input graph G is undirected. What is the time complexity of the following function F in terms of the number of vertices in the graph (n) and the number of edges in the graph (m)?

```
<V> void F(Graph<V> G, HashSet<V> S) {
    for (V u : G.vertices())
        if (! S.contains(u))
          H(G, u, S);
}

<V> void H(Graph<V> G, V u, HashSet<V> S) {
    S.add(u);
    for (V v : G.adjacent(u))
        if (! S.contains(v))
            H(G, v, S);
}
```

Solution: The time complexity is $O(n + m)$ **(2 points)** because the algorithm processes every vertex exactly once **(2 points)** and every edge exactly twice, doing a constant amount of work for each one. The set S ensures that every vertex is only processed once. Each edge is only processed at most twice, once from each vertex that it is incident on. **(2 points)**

8. [9 points] Identify the connected components in the following graph. For each connected component, choose its representative to be the node that is lowest alphabetically (A before Z). For each vertex in the graph, list its representative.



Solution: (1 point each)

1. A: A
2. B: B
3. C: B
4. D: B

Name: ____________________

5. E: B
6. F: A
7. G: A
8. H: A
9. I: I

9. 10 points Fill in the blanks to complete the align method in the following DNA SequenceAligner class. The align method should return the score (an integer) for the best alignment for a prefix of string x and a prefix of string y. The parameter i is the length of the prefix of x and parameter j is the length of the prefix of y. Recall that the Judge class has a score method that takes two characters and returns the score for lining them up. Also recall that the character for a gap is given by GAP_CHAR.

```
class SequenceAligner {
    private String x, y;
    private Judge judge;
    public int align(int i, int j) {
        if (i == 0 && j == 0) {
            return ___(a)___;
        } else if (i == 0) {
            return ___(b)___ + judge.score(GAP_CHAR, y.charAt(j - 1));
        } else if (j == 0) {
            return ___(c)___ + judge.score(x.charAt(i - 1), GAP_CHAR);
        } else {
            char nextX = x.charAt(i - 1), nextY = y.charAt(j - 1);
            int diag = ___(d)___ + judge.score(nextX, nextY);
            int left = align(i, j - 1) + judge.score(GAP_CHAR, nextY);
            int up = align(i - 1, j) + judge.score(nextX, GAP_CHAR);
            return ___(e)___;
        }
    }
}
```

Solution: (2 points each)

```
(a) 0
(b) align(i, j - 1)
(c) align(i - 1, j)
(d) align(i - 1, j - 1)
(e) Math.max(diag, Math.max(left, up))
```

Name: ____________________

10. 10 points The Galatic Republic is growing and your assignment as a Hyperspace Engineer is to decide where to establish hyperspace routes, that is, selecting pairs of planets to place entrance/exit beacons for each route. The Republic currently consists of only five planets: Alderaan, Hoth, Dagobah, Tatooine, and Coruscant. Your goal is to make it possible to travel from any of these planets to any other of them via hyperspace routes. However, one needs to keep in mind that the cost of constructing a hyperspace route is proportional to the number of light years that separates the two planets. The distances in light years between pairs of planets is given in the following table. State which hyperspace routes to build so that the total length is minimized. State the total length of the hyperspace routes you choose. Show your work.

	Alderaan	Hoth	Dagobah	Tatooine	Coruscant
Alderaan	-	30	9	8	21
Hoth	-	-	14	35	11
Dagobah	-	-	-	6	12
Tatooine	-	-	-	-	10

Solution: This solution applies Kruskal's algorithm to compute the minimum spanning tree (MST). **(3 points)**

The first step is to order the edges by weight: **(2 points)**

$$D–T(6), A–T(8), A–D(9), T–C(10), H–C(11),$$
$$D–C(12), H–D(14), A–C(21), A–H(30), H–T(35)$$

Next we process each edge, deciding to make it part of the MST or not, depending on whether the endpoints of each edge are in different disjoint sets.

$$D–T(yes), A–T(yes), A–D(no), T–C(yes), H–C(yes),$$
$$D–C(no), H–D(no), A–C(no), A–H(no), H–T(no)$$

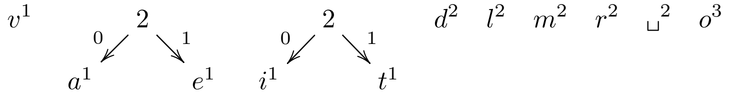
So we place hyperspace routes as indicated by the "yes" answers above. **(3 points)** The total length of the routes is $6 + 8 + 10 + 11 = 35$ light years. **(2 points)**

Name: ____________________

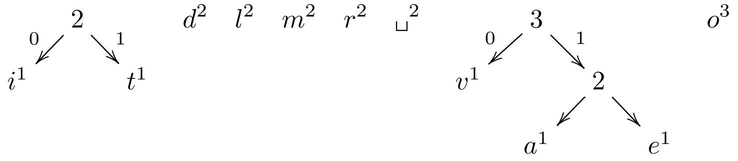
11. 9 points Construct a Huffman code for a message that contains the following letters and whitespace with the frequencies given below. That is, create a variable length encoding for each letter that minimizes the length of the entire message. Show your work.

$$a : 1, d : 2, e : 1, i : 1, l : 2, m : 2, o : 3, r : 2, t : 1, v : 1, \sqcup : 2$$

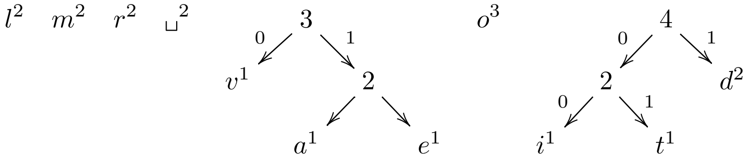
Solution: This is one of many solutions. The lowest frequency pairs of letters (frequency 1) are picked first and combined into trees. **(1 point)**



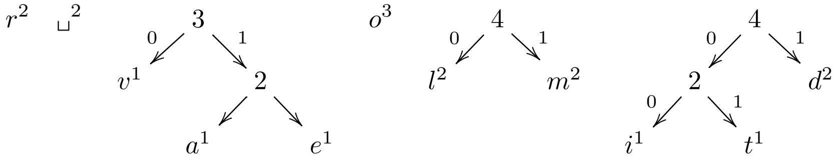
The last of the frequency 1 characters, in this case number v, gets combined with a frequency 2 subtree to become a frequency 3 subtree. **(1 point)**



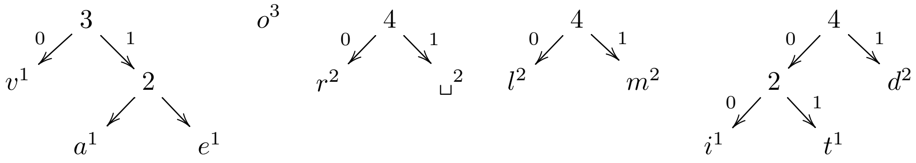
Now we combine the first two pairs of frequency 2 subtrees. **(1 point)**



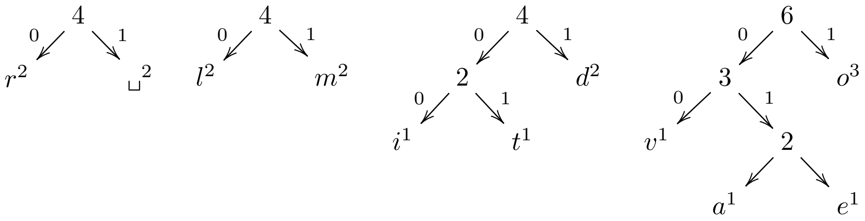
Again, we combine the first two pairs of frequency 2 subtrees. **(1 point)**



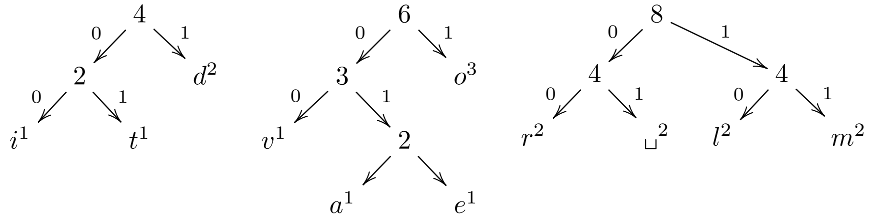
And again. **(1 point)**



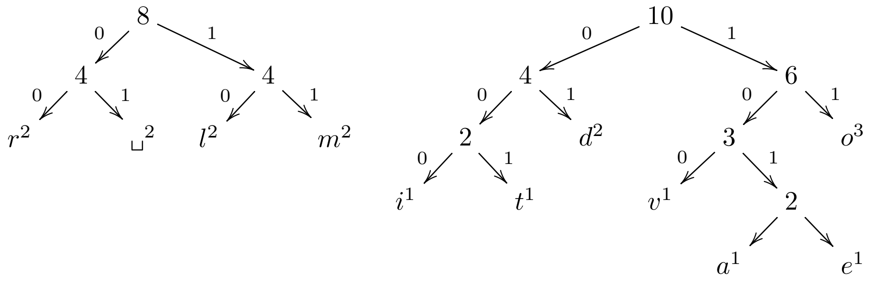
Next we combine the frequency 3 subtrees. **(1 point)**



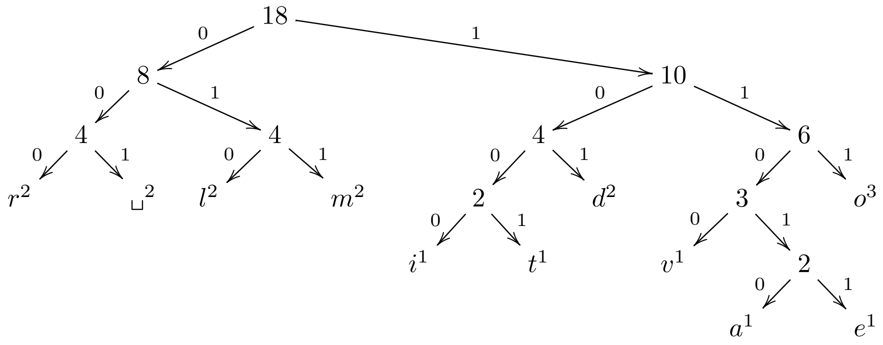
Then combine the first two frequence 4 subtrees. **(1 point)**



Combine the frequence 4 and 6 subtrees. **(1 point)**



And here's the final answer: **(1 point)**



12. $\boxed{\text{4 points}}$ What advice would you give a student taking Data Structures next year?

Solution: Open ended.