add the inverse of any edge). Please separate the edges in each list with commas. See example answer format below for reference.

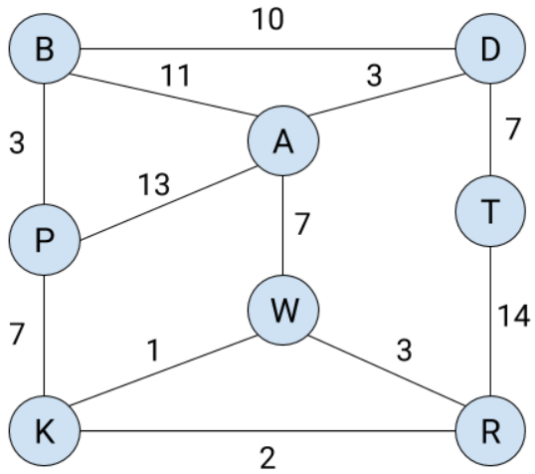
If you need to break ties between equal weighted edges, arrange each edge alphabetically and then compare the edges alphabetically. For example, consider breaking a tie between the edges ZA and BY. First, arrange each edge alphabetically, so ZA becomes AZ and BY remains BY. Then, compare them alphabetically. Since AZ comes before BY, AZ should be selected before BY.

Do NOT continue the algorithm once you have a complete MST.

If you need a starting node for either algorithm, use **K**.

Answer Format: In the box below, state the two lists and type out the edges. For example:

```
Kruskal's: AB, BC, CD, DE, EF, FG, GH
Prim's: GH, FG, EF, DE, CD, BC, AB
```



HTML Editor

12pt Paragraph

0 words

Question 2

10 pts

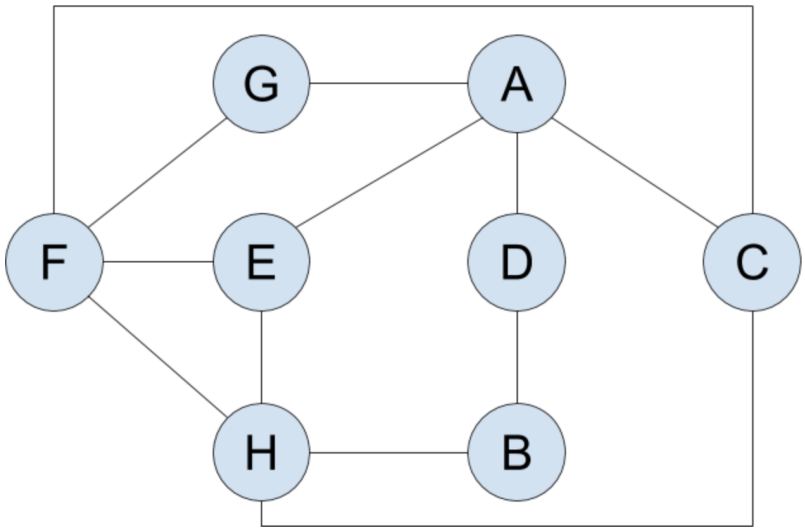
Part A: Given the graph below, perform the **Breadth First Search algorithm**. Write the vertices in the order that they are visited.

Part B: Given the same graph below, perform the **Depth First Search algorithm**. Assume DFS is implemented using an actual stack, NOT using the recursive stack. Keep in mind that with every iteration, you must add all neighboring vertices to the stack before popping for the next iteration. Write the vertices in the order that they are visited (not just added to the stack).

Instructions for both parts: If there are multiple edges incident to the current vertex, add the neighboring vertices to the stack one-by-one in alphabetical order. **If you need a starting node, use A**.

Answer Format: In the box below, type "BFS: ", and then type out the vertices in the order that they are visited. On the following line, type "DFS: ", and then type out the vertices in the order that they are visited. For example:

```
BFS: ABCDEFGH
DFS: HGFEDCBA
```



HTML Editor

B I U A A 12pt Paragraph

0 words

Question 3

10 pts

Below is an implementation of Dijkstra's algorithm. There are exactly 5 lines with minor mistakes in this implementation. Find 4 of these mistakes.

Note: You do **not** need to worry about Exception messages in the code (lines 2 - 11). You can assume that the code compiles. You also do **not** need to worry about Java errors (including Generics) - the errors are all associated with how the actual Dijkstra's algorithm works (semantic errors).

Answer Format: In the box below, for all mistakes you locate, state the line number where the code is incorrect, **and** type out how you would correct the line. For example:

```
Line 3: if (i == 0) {
```

Do NOT claim a line is a mistake if it is simply written differently than how you would write the code. If the code functions as expected, it is correct and therefore not a mistake. You will never need (and should not provide) multiple lines to fix a single line's error. List **ONLY 4** of the mistakes. If you choose to list more, you will **ONLY** be graded on the first 4 listed.

```java
public static <T> Map<Vertex<T>, Integer> dijkstras(Vertex<T> start, Graph<T> graph) {
    if (start == null) {
        throw new IllegalArgumentException("Cannot find shortest path "
            + "when starting vertex is null");
    } else if (graph == null) {
        throw new IllegalArgumentException("Cannot find shortest path "
            + "when graph is null");
    } else if (!graph.getVertices().contains(start)) {
        throw new IllegalArgumentException("Cannot find shortest path "
            + "when starting vertex is not in graph");
    }
    Map<Vertex<T>, Integer> shortestDistanceMap = new HashMap<>();
    for (Vertex<T> vertex : graph.getVertices()) {
        shortestDistanceMap.put(vertex, 30);
    }
    shortestDistanceMap.put(start, 0);
    Set<Vertex<T>> visited = new HashSet<>();
    PriorityQueue<VertexDistance<T>> queue = new PriorityQueue<>();
    queue.add(new VertexDistance<>(start, 0));
    while (visited.size() == 1 && !queue.isEmpty()) {
        Vertex<T> vertexA = queue.remove().getVertex();
        if (!visited.contains(vertexA)) {
            visited.add(vertexA);
            for (VertexDistance<T> edge : graph.getAdjList().get(vertexA)) {
                Vertex<T> vertexB = edge.getVertex();
                if (visited.contains(vertexB)) {
                    int dist = shortestDistanceMap.get(vertexA) + edge.getDistance();
                    if (shortestDistanceMap.get(vertexB) < dist) {
                        shortestDistanceMap.put(vertexB, dist);
                        queue.add(new VertexDistance<>(vertexB, shortestDistanceMap.get(vertexA)));
                    }
                }
            }
        }
    }
    return shortestDistanceMap;
}
```

HTML Editor

B I U A A Tx ≡ ≡ ≡ ≡ ≡ x² x₂ ☰ ☰

12pt Paragraph

0 words

Question 4

10 pts

Swap Notation: If two values in the array are to swap such as '9' and '4' in the following example, then in the row below, after swapping the elements, denote them with an 's' like '4s' and '9s' .

index	0	1	2
array	9	5	4
swapped	4s	5	9s

Goal: Given the following unsorted array of integers, **perform two iterations of in-place QuickSort** as taught in lecture. (Any implementation that is different than what was done in lecture will receive NO points). Each row of the table shows the next state of array after a swap has occurred. The algorithm is in-place.

Requirements: Sort in ascending order. Retype the array, on the line below, **each time** a swap is necessary, and type an 's' after **only** the elements that were swapped (see Swap Notation above). All rows may not be needed, but you should not need any more rows than what is provided.

Answer Format: Enter your answer using the table below following the **swap notation** and **requirements** provided above. You will not need any more rows that what is provided.

Iteration 1, use index 3 as the pivot:

i	0	1	2	3	4	5	6	7	8	9
a[i]	35	18	41	64	27	89	71	82	67	48

Iteration 2, use index 4 as the pivot: Recopy the LEFT subarray of the last row of the first iteration, up to the pivot, into the first row of the second iteration. Only these elements are needed in iteration 2. Enter your work into the table below. You will not need any more rows or columns than what is provided.

i	0	1	2	3	4	5	6	7	8	9

Question 5

5 pts

Given the strings, "BRUNCHEGG" and "BERRYBUSH", determine the length of the longest common subsequence by filling in the table below. Write the length of the longest common subsequence and the LCS in the spaces provided. The table must be accurately filled in to receive full credit.

		B	R	U	N	C	H	E	G	G
	0	0	0	0	0	0	0	0	0	0
B	0									
E	0									
R	0									
R	0									
Y	0									
B	0									
U	0									
S	0									
H	0									

Length:

Longest Common Subsequence:

Question 6

1 pts

For at least one TA that was particularly helpful to you this semester, write the name of a song or write them an original song. If you write an original song, do not write explicit lyrics or you will not get credit. Original songs that surpass the expectations of the TAs are eligible to receive **an additional point**. You can write the name of the TA(s) your song is for in the entry box below, and write the name of the song under the TA name(s). The final exam is **not** an appropriate medium for making **advances** or **inappropriate comments** towards a teaching assistant. Inappropriate submissions will earn a **0** for the entire exam.

Head TA - Adrianna Brown

Senior TA/B3 Grading - David Wang

Senior TA/B3 Grading - Rodrigo Pontes

Online Head TA - Caroline Kish

O1 - Jacob Allen

O1 - Landon Ryan

O1 - Isaac Weintraub

A1 - Paige Ryan

A1 - Tillson Galloway

A2 - Yotam Kanny

A2 - Aviva Kern

A3 - Destini Deinde-Smith

A3 - Siddu Dussa

A4/A5/A6/GR1 - Neha Deshpande

A4/A5/A6/GR1 - Isaac Tomblin

B1 - Reece Gao

B1 - Rena Li

B2 - Miguel de los Reyes

B2 - Smita Mohindra

B3 - Sanjana Tewathia

B4 - Mitchell Gacuzana

B4 - Eunseo Cho

B5 - Elena May

B5 - Ivan Leung

B6 - Anjana Nandagopal

B6 - Alex McQuilkin

C1 - Brooke Miller

C1 - Cliff Panos

C2/GR2 - Brandon Vu

C2/GR2 - Ila Vienneau

C3/C4 - Nick Worthington

C3/C4 - Keely Culbertson

HTML Editor

12pt Paragraph

0 words

Quiz saved at 6:13pm Submit Quiz