Started on	Monday, 5 May 2025, 2:48 PM
State	Finished
Completed on	Monday, 5 May 2025, 3:01 PM
Time taken	13 mins 19 secs
Marks	13.00/15.00
Grade	**86.67** out of 100.00

Question 1

Complete

Mark 0.00 out of 1.00

.Consider the following adjacency matrix representing a graph with 4 nodes (0, 1, 2, 3):

0 1 0 1

1 0 1 0

0 1 0 1

1 0 1 0

Which of the following is the correct graph representation?

a. A graph with 4 edges

b. A graph with 6 edges

c. A graph with 5 edges

d. A graph with 2 edges

Question 2

Complete

Mark 1.00 out of 1.00

Consider the following directed graph represented by its adjacency list:

graph.put(0, Arrays.asList(1, 2));

graph.put(1, Arrays.asList(3));

graph.put(2, Arrays.asList(3));

graph.put(3, Arrays.asList());

What will be the output of the following DFS traversal starting from node 0?

dfs(0, visited, graph);

- a. 1 0 2 3
- b. 0 2 1 3
- c. 0 1 3 2
- d. 0 1 2 3

Question 3

Complete

Mark 1.00 out of 1.00

Consider the following Java code snippet to detect a cycle in an undirected graph:

```
public boolean hasCycle(int node, int parent, Set<Integer> visited, Map<Integer, List<Integer>> graph) {
visited.add(node);
for (int neighbor : graph.get(node)) {
if (!visited.contains(neighbor)) {
    if (hasCycle(neighbor, node, visited, graph)) {
            return true;
    }
} else if (neighbor != parent) {
    return true;
}
return false;
}
```

In the context of this code, which of the following statements is TRUE?

a. The code will throw an error because it doesn't handle visited nodes correctly.

b. The code cannot detect cycles in graphs.

c. The code works for directed graphs only.

d. The code works for undirected graphs and detects cycles if any.

Question 4

Complete

Mark 0.00 out of 1.00

Let G be a simple graph with 20 vertices and 8 components. If we delete a vertex in G, then number of components in G should lie between ___.

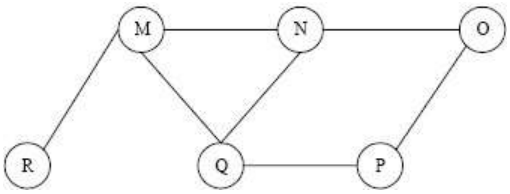
- a. 7 and 19
- b. 8 and 20
- c. 8 and 19
- d. 7 and 20

Question 5

Complete

Mark 1.00 out of 1.00

The Breadth First Search algorithm has been implemented using the queue data structure. One possible order of visiting the nodes of the following graph is



- a. QMNPRO
- b. QMNPOR
- c. NQMPOR
- d. MNOPQR

Question 6

Complete

Mark 1.00 out of 1.00

What is the output of this adjacency list code?

```
Map<String, List<String>> graph = new HashMap<>();
graph.put("A", Arrays.asList("B", "C"));
graph.put("B", Arrays.asList("A", "D"));
graph.put("C", Arrays.asList("A"));
graph.put("D", Arrays.asList("B"));

System.out.println(graph.get("B"));
```

- a. [A, D]
- b. [B, D]
- c. [D, A]
- d. [A, C]

Question 7

Complete

Mark 1.00 out of 1.00

What traversal algorithm is implemented here?

```
public void traverse(String start) {
Set<String> visited = new HashSet<>();
Queue<String> queue = new LinkedList<>();
queue.add(start);
visited.add(start);

while (!queue.isEmpty()) {
String node = queue.poll();
System.out.print(node + " ");
for (String neighbor : graph.get(node)) {
   if (!visited.contains(neighbor)) {
            queue.add(neighbor);
            visited.add(neighbor);
   }
}
}
}
```

- [x] a. BFS
- [ ] b. Dijkstra
- [ ] c. DFS
- [ ] d. Topological Sort

Question 8

Complete

Mark 1.00 out of 1.00

What does this method do?

```
public boolean Path(String src, String dest, Set<String> visited) {
if (src.equals(dest)) return true;
visited.add(src);
for (String neighbor : graph.get(src)) {
if (!visited.contains(neighbor)) {
    if (Path(neighbor, dest, visited)) return true;
}
}
return false;
}
```

- [x] a. Checks if a path exists using DFS
- [ ] b. Prints the graph
- [ ] c. Finds shortest path
- [ ] d. Detects a cycle

Question 9

Complete

Mark 1.00 out of 1.00

Which scenario causes a cycle in an undirected graph?

- [ ] a. Graph has a node with degree 1
- [x] b. A node connects back to a visited node that is not its parent
- [ ] c. Graph has a node with no outgoing edge
- [ ] d. All nodes are visited exactly once

Question 10

Complete

Mark 1.00 out of 1.00

What is the time complexity of BFS in an adjacency list representation?

- a. $O(V^2)$
- b. O(E + V)
- c. O(V log E)
- d. O(E log V)

Question 11

Complete

Mark 1.00 out of 1.00

What does this method count?

```
public int countComponents() {
Set<String> visited = new HashSet<>();
int count = 0;
for (String node : graph.keySet()) {
if (!visited.contains(node)) {
    dfs(node, visited);
    count++;
}
}
return count;
}
```

- a. Number of connected components
- b. Number of cycles
- c. Number of dead ends
- d. Number of leaf nodes

Question 12

Complete

Mark 1.00 out of 1.00

What is a dead-end node in an undirected graph?

- a. Node with no neighbors
- b. Node with maximum degree
- c. Node with only one neighbor
- d. Node in a cycle

Question 13

Complete

Mark 1.00 out of 1.00

What will this method return for a disconnected graph?

```
public boolean hasCycle(String node, String parent, Set<String> visited) {
visited.add(node);
for (String neighbor : graph.get(node)) {
if (!visited.contains(neighbor)) {
    if (hasCycle(neighbor, node, visited)) return true;
} else if (!neighbor.equals(parent)) {
    return true;
}
}
return false;
}
```

- a. Always detects a cycle
- b. Only works for directed graphs
- c. Only detects self-loops
- d. May return false even if there is a cycle in another component

Question 14

Complete

Mark 1.00 out of 1.00

The following code snippet is the function to insert a string in a trie. Find the missing line.

```
private void insert(String str){
  TrieNode node = root;
  for (int i = 0; i < length; i++){
    int index = key.charAt(i) - 'a';
    if (node.children[index] == null)
      node.children[index] = new TrieNode();
    _____
  }
  node.isEndOfWord = true;
}
```

a. node = node.children[index++];

b. node = node.children[index];

c. node = node.children[index++];

d. node = node.children[str.charAt(i + 1)];

Question 15

Complete

Mark 1.00 out of 1.00

Which of the following is an advantage of adjacency list representation over adjacency matrix representation of a graph?

a. DFS and BSF can be done in O(V + E) time for adjacency list representation. These operations take O(V^2) time in adjacency matrix representation. Here is V and E are number of vertices and edges respectively.

b. Adding a vertex in adjacency list representation is easier than adjacency matrix representation.

c. All of the above

d. In adjacency list representation, space is saved for sparse graphs.