Task 4

Graph Edge Addition Validation

Given a directed graph, write a function that adds an edge between two nodes and then checks if the graph still has no cycles. If a cycle is created, the edge should not be added.

ANS:

```
package Day13;
import java.util.*;
class Graph {
   private int V;
   private List<List<Integer>> adj;
   public Graph(int V) {
      this.V = V;
      adj = new ArrayList<>(V);
      for (int i = 0; i < V; i++) {
         adj.add(new ArrayList<>());
      }
   }
   public void addEdge(int u, int v) {
      adj.get(u).add(v);
   }
   public boolean isReachable(int source, int destination) {
      boolean[] visited = new boolean[V];
      Queue<Integer> queue = new LinkedList<>();
      visited[source] = true;
      queue.offer(source);
      while (!queue.isEmpty()) {
         int current = queue.poll();
         if (current == destination) {
            return true;
         }
         for (int neighbor : adj.get(current)) {
            if (!visited[neighbor]) {
               visited[neighbor] = true;
               queue.offer(neighbor);
            }
         }
      }
```

```
        return false;
    }
}
public class Task4 {
    public static boolean isSafeToAddEdge(Graph graph, int u, int v) {
        // Check if adding the edge creates a cycle
        return !graph.isReachable(v, u);
    }
    public static void main(String[] args) {
        int V = 4; // Number of vertices
        Graph graph = new Graph(V);

        graph.addEdge(0, 1);
        graph.addEdge(0, 2);
        graph.addEdge(1, 2);
        graph.addEdge(2, 0);
        graph.addEdge(2, 3);
        graph.addEdge(3, 3);
        int u = 1, v = 3;
        boolean isSafe = isSafeToAddEdge(graph, u, v);
        System.out.println("Is it safe to add edge from " + u + " to " + v
+ "? " + isSafe);
    }
}
```

OUTPUT:

```
Is it safe to add edge from 1 to 3? true
```