Given an algorithm, a configuration is an assignment of values to each variable. A configuration is feasible if it can actually occur during the execution of the algorithm. Let \( C \) denote the set of all configurations, whether feasible or not. We refer to \( C \) as the configuration space.

For instance, a configuration for Dijkstra’s algorithm would consist of a status value (white, grey, black), a cost value (a real number or \( \infty \)), and a parent link (possibly NIL) for each vertex, and a set \( Q \) (the priority queue; here we treat it simply as a set of keys).

A predicate over a set \( A \) is a Boolean function \( f : A \to \{0, 1\} \) (1: “true,” 0: “false”). A transformation of \( A \) is a function \( g : A \to A \).

Let \( P \) and \( Q \) be predicates over the configuration space \( C \) and let \( S \) be a set of instructions, viewed as a transformation of \( C \). Consider the loop “while \( P \) do \( S \).” We say that \( Q \) is a loop-invariant for this loop if for all configurations \( X \in C \) it is true that

\[
P(X) & Q(X) \Rightarrow Q(S(X)).
\]  

(1)

In other words, if \( P \& Q \) holds for the configuration \( X \) then \( Q \) also holds for the configuration \( S(X) \), where \( S(X) \) is the configuration obtained from \( X \) by executing \( S \).

Note that the highlighted statement has to hold even for infeasible configurations. This is analogous to chess puzzles: when showing that a certain configuration leads to checkmate in two moves, you do not investigate whether or not the given configuration could arise in an actual game.

Dijkstra’s algorithm consists of iterations of a single “while” loop. Consider the following statements:

\begin{align*}
Q_1 & : (\forall u \in V)( \text{ if } u \text{ is white then } c(u) = \infty) \footnote{In a previous version of this handout, \( Q_1 \) was erroneously stated as saying “(\( \forall u \in V \))(u \text{ is white if and only if } c(u) = \infty).” Show that this statement is not a loop invariant.}.
Q_2 & : (\forall u \in V)(u \text{ is grey if and only if } u \in Q).
Q_3 & : (\forall u, v \in V)( \text{ if } u \text{ is black and } v \text{ is not black then } c(u) \leq c(v)).
Q_4 & : (\forall v \in V)(c(v) \text{ is the minimum cost among all } s \rightarrow \ldots \rightarrow v \text{ paths that pass through black vertices only}).
\end{align*}

1. Prove that \( Q_1 \) and \( Q_2 \) are loop-invariants.
2. Prove that \( Q_1 \& Q_2 \& Q_3 \) is a loop-invariant.
3. Prove that \( Q_1 \& Q_2 \& Q_3 \& Q_4 \) is a loop-invariant.
4. Prove that \( Q_1 \& Q_2 \& Q_4 \) alone is not a loop-invariant. Explanation. You need to construct a weighted directed graph with nonnegative weights, a
source, and an assignments of all the variables (parent pointers, status colors, current cost values) such that $Q_4$ holds for your configuration, but $Q_4$ will no longer hold after executing Dijkstra’s while loop. Your graph should have very few vertices.