Question 1

There is not always a stable pair of positionings. To see this, consider the following example:

Each team has 5 players, i.e. n=5. Let U’s players be u₁, u₂, u₃, u₄ and u₅, and let Q’s players be q₁, q₂, q₃, q₄ and q₅. The players’ skill levels are:

\[
\begin{array}{cccc}
  u₁ & u₂ & u₃ & u₄ & u₅ \\
  1 & 3 & 5 & 7 & 9 \\
\end{array}
\quad
\begin{array}{cccc}
  q₁ & q₂ & q₃ & q₄ & q₅ \\
  2 & 4 & 6 & 8 & 10 \\
\end{array}
\]

It is obvious that for any positioning U gives, there is a positioning W can give so that W wins every match. But for any positioning W gives, there is a positioning U can give so that U wins all but one match (i.e. ‘sacrifice’ the skill 1 player, and have them play the skill 10 player, then match up the remaining players so that they win.)

Therefore, there cannot be any stability.

Question 2

For each boy’s dancing schedule, we have to choose a home girl: the girl he’ll be dancing with when he (and she) is sent home. We will say that a send-home schedule tearless if the resulting dance has no boys that end up crying because their next partner was sent home.

We set up a stable marriage problem involving boys and girls. Each boy ranks each girl in chronological order of his dance with them. Each girl ranks each boy in reverse chronological order of her dance with him. We only need show that:

Claim: A stable matching between boys and girls defines a tearless send-home schedule.
Proof. If the send-home schedule is not tearless, then it violates the condition (*). That is, some boy \( b_i \) goes to ask girl \( g_k \) to dance, but \( g_k \) has already been sent home with (well, at the same time as) boy \( b_j \). But in this case, under our preference relation above, boy \( b_i \) “prefers” girl \( g_k \) to the girl he actually gets sent home with, and girl \( g_k \) “prefers” boy \( b_i \) to boy \( b_j \). This contradicts the assumption that we chose a stable matching between boys and girls.

Question 3

Define a new graph \( G' = (V', E') \), based on the graph \( G \). For each vertex \( v \in V \) and each month \( m \), we create a vertex \( (v, m) \) in \( V' \) so that there is a vertex that represents a specific location at a given month. For each edge \( e = (u, v) \in E \) and each month \( m \), we create the edge \( ((u, m), (v, (m + d) \mod 12)) \) with cost \( d \), where \( d = C[e, m] \), so that an edge represents the time it takes to travel from vertex \( u \) at the beginning of month \( m \) to the vertex \( v \). We also add edges \( ((u, m), (u, (m + 1) \mod 12)) \) with cost 1, for all \( u \) and all \( m \), which correspond to staying in the same port for a month. Lastly, we add one supernode \( v^* \), to which all \((v, \cdot)\) nodes connect to, at 0-cost.

Now we run any shortest paths algorithm, starting at vertex \((u, m)\). We find out which path gets to the supernode vertex \( v^* \) the fastest, and that is the route we take. We want to show that:

Claim: The shortest path between vertex \((u, m)\) and \( v^* \) in the graph \( G' \) is equivalent to a fastest route between \( u \) and \( v \) in the graph \( G \), given that we start at the beginning of month \( m \).

Proof. The shortest path between vertex \((u, m)\) and \( v^* \) in the graph \( G' \) is also the shortest path between vertex \((u, m)\) and some vertex \((v, m')\). Let the cost of this shortest path \( P \) be \( c \). By our construction, this clearly corresponds to a journey from \( u \) to \( v \) starting at month \( m \) that takes \( c \) months. We want to show that no faster journey exists.

If \( P \) wasn’t a fastest route then there is some sequence of ports in \( G \) such that we can get from \( u \) to \( v \) in time \( c^* < c \), if we start at the beginning of month \( m \). But then, by our construction of \( G' \), we have a path in \( G' \) from \((u, m)\) to some \((v, m'')\) with cost \( c^* \), a contradiction to path \( P \) being the shortest path.