The algorithm $PARTITION(A, p, r)$ we have seen in class for the Quicksort algorithm partitions the array $A$ between indices $p$ and $r$ based on one entry, $A[r]$, and returns one index, $q$.

In an attempt to make the Quicksort algorithm more efficient (without changing the basic idea behind the algorithm), $PARTITION(A, p, r)$ was rewritten to partition the array $A$ between indices $p$ and $r$ based on two entries, $A[p]$ and $A[r]$. The new algorithm is called $PARTITION3(A, p, r)$, and it returns two indices, $q_1$ and $q_2$.

(1) $PARTITION3(A, p, r)$ was applied to the following array.

\[
\begin{array}{cccccccc}
p & 10 & 1 & 12 & 4 & 7 & 15 & 9 & 3 & 5 \\
\end{array}
\]

Without going into the details of how $PARTITION3()$ is implemented, for every one of the following arrays, mark whether it is possible for $PARTITION3()$ to produce it from the array given above. If so, specify the indices $q_1$ and $q_2$.

\[
\begin{array}{cccccccc}
p & 5 & 1 & 3 & 4 & 7 & 9 & 12 & 15 & 10 \\
\end{array}
\]

No

\[
\begin{array}{cccccccc}
p & 4 & 1 & 3 & 5 & 9 & 7 & 10 & 15 & 12 \\
\end{array}
\]

Yes, $q_1 = p + 3, q_2 = p + 6$

(2) Write a version of the Quicksort algorithm that uses $PARTITION3()$.

\[
QUICKSORT(A, p, r) \\
1 \text{ if } p < r: \\
2 \quad q_1, q_2 \leftarrow PARTITION3(A, p, r) \\
3 \quad QUICKSORT(A, p, q_1 - 1) \\
4 \quad QUICKSORT(A, q_1 + 1, q_2 - 1) \\
5 \quad QUICKSORT(A, q_2 + 1, r)
\]
The algorithm \textit{PARTITION}(A, p, r) we have seen in class for the Quicksort algorithm partitions the array \( A \) between indices \( p \) and \( r \) based on one entry, \( A[r] \), and returns one index, \( q \).

In an attempt to make the Quicksort algorithm more efficient (without changing the basic idea behind the algorithm), \textit{PARTITION}(A, p, r) was rewritten to partition the array \( A \) between indices \( p \) and \( r \) based on two entries, \( A[r] \) and \( A[r-1] \). The new algorithm is called \textit{PARTITION3}(A, p, r), and it returns two indices, \( q_1 \) and \( q_2 \).

(1) \textit{PARTITION3}(A, p, r) was applied to the following array.

\[
\begin{array}{cccccccc}
p & & & & & & r \\
3 & 1 & 12 & 4 & 7 & 15 & 10 & 9 & 5
\end{array}
\]

Without going into the details of how \textit{PARTITION3}() is implemented, for every one of the following arrays, mark whether it is possible for \textit{PARTITION3}() to produce it from the array given above. If so, specify the indices \( q_1 \) and \( q_2 \).

\[
\begin{array}{cccccccc}
p & & & & & & r \\
5 & 1 & 3 & 4 & 7 & 10 & 12 & 15 & 9
\end{array}
\]

No

\[
\begin{array}{cccccccc}
p & & & & & & r \\
4 & 1 & 3 & 5 & 7 & 9 & 10 & 15 & 12
\end{array}
\]

Yes \( q_1 = p + 3, \ q_2 = p + 5 \)

(2) Write a version of the Quicksort algorithm that uses \textit{PARTITION3}().

\begin{verbatim}
QUICKSORT(A, p, r)
1 if p < r:
2     q1, q2 ← PARTITION3(A, p, r)
3     QUICKSORT(A, p, q1 - 1)
5     QUICKSORT(A, q1 + 1, q2 - 1)
5     QUICKSORT(A, q2 + 1, r)
\end{verbatim}