NAME:

Consider \( n \) jobs \( j = 0, 1, \ldots, n - 1 \) to be assigned to \( n \) people \( i = 0, 1, \ldots, n - 1 \).
The matrix \( Q \) with entries \( Q[i, j] \) indicates the qualification of person \( i \) for job \( j \).
The higher \( Q[i, j] \), the better \( i \) is qualified for job \( j \). For example:

\[
Q = \begin{bmatrix}
7 & 4 & 2 & 4 \\
6 & 8 & 5 & 2 \\
4 & 7 & 1 & 3 \\
6 & 5 & 2 & 1
\end{bmatrix}
\]

Assigning person \( i \) to job \( i, i = 0, 1, 2, 3 \), and summing up their qualifications \( Q[i, i] \) gives 17.
An optimal matching assigns every job to exactly one person and maximizes the sum of their qualifications.

Represent a job assignment by a permutation \( P \) of \([0, 1, \ldots, n - 1]\): person \( i \) does job \( P[i] \).

1. Given an \( n \)-by-\( n \) numpy matrix \( Q \) and a job assignment in a list \( P \), define a Python function \texttt{Value} which returns the sum of the qualifications of the job assignment.

2. Write a Python function \texttt{Match} which takes on input an \( n \)-by-\( n \) numpy matrix \( Q \) and additional parameters to control the backtracking. \texttt{Match} returns a tuple \((P, m)\), where \( P \) is the optimal matching and \( m \) its corresponding value.