1. Consider Example 2.2.1 in the textbook and assume that the received sequence is 
\{z_1, z_2, z_3, z_4\} = \{101, 011, 110, 000\}.
Assume that the noisy channel transmits every binary symbol correctly with probability 0.9. Find a decoded sequence
\{\hat{w}_1, \hat{w}_2, \hat{w}_3, \hat{w}_4\}
that has the maximum likelihood.

2. Solve an instance of the traveling salesman problem given in Example 2.3.2 using a label correcting method when the distances between the cities are given by the matrix
\[
\begin{pmatrix}
\times & 6 & 2 & 2 & 3 \\
2 & \times & 4 & 3 & 7 \\
5 & 3 & \times & 2 & 1 \\
1 & 9 & 4 & \times & 3 \\
3 & 7 & 3 & 5 & \times
\end{pmatrix}
\]

3. Write a MATLAB script to solve the general binary programming problem
\[
\min_x f^T x,
\]
subject to
\[Ax \leq b\] and \[x_i \in \{0, 1\},\]
by using the branch-and-bound algorithm and the LP relaxation ideas discussed in the class.
Using your script solve the following assignment problem: \(N\) different tools need to be assigned to \(M\) different jobs. Tools are labeled as \(T_1, \ldots, T_N\) and jobs are labeled as \(J_1, \ldots, J_M\) (not to be confused with the cost-to-go functions). If a set \(\{T_i\}_{i \in I}\) of tools are assigned to the job \(J_j\) then the job gets done with probability \(1 - \prod_{i \in I} (1 - p_{ij})\) where \(p_{ij}\) is the probability that the tool \(T_i\) alone gets the job \(J_j\) done. If the job \(J_j\) gets done then a reward \(V_j\) is realized. Find a tool-job assignment that maximizes the expected total reward. Use the following numerical values:
\[V_j = j, \quad p_{ij} = |\sin(i)\sin(j)|, \quad i = 1, \ldots, 10, \quad j = 1, \ldots, 9\]
where \(\sin(\cdot)\) is in radians.

Hint: Let \(x_{ij}\) be a binary variable taking the value 1 if and only if \(T_i\) is assigned to \(J_j\). Define \(q_{ij} := 1 - p_{ij}\).
Our optimization problem can now be written as
\[
\min_{x_{ij}} \sum_j V_j \prod_i q_{ij}^{x_{ij}}
\]
subject to
\[\sum_j x_{ij} = 1, \quad \forall i.\]
Note that the cost above is a nonlinear function of \(x_{ij}\)'s. To be able to use the binary programming solver, find a straight line approximation to the cost as follows:
\[
\sum_j V_j \prod_i q_{ij}^{x_{ij}} = \sum_{j=1}^M e^{\log(V_j) + \sum_i x_{ij} \log(q_{ij})} \\
\approx \sum_{j=1}^M \alpha_j [\log(V_j) + \sum_i x_{ij} \log(q_{ij})] + \beta_j,
\]
where the line $\alpha_j y + \beta_j$ approximates $e^y$ on $[\log(V_j \prod_i q_{ij}), \log(V_j)]$. One reasonable way of choosing $\alpha_j$ and $\beta_j$ would be to minimize the error criterion

$$\int_{\log(V_j \prod_i q_{ij})}^{\log(V_j)} [\alpha_j y + \beta_j - e^y]^2 dy.$$ 

Straight forward calculations show that $\alpha_j$ and $\beta_j$ minimizing the error criterion above must satisfy

$$\begin{bmatrix}
\frac{2}{3} (\bar{y}_j^3 - \bar{y}_j^2) \\
\bar{y}_j^2 - \bar{y}_j^2
\end{bmatrix}
\begin{bmatrix}
\alpha_j \\
\beta_j
\end{bmatrix}
= 2
\begin{bmatrix}
e^{\bar{y}_j} (\bar{y}_j - 1) - e^{\bar{y}_j} (\bar{y}_j - 1) \\
e^{\bar{y}_j} - e^{\bar{y}_j}
\end{bmatrix},
$$

where $\bar{y}_j = \log(V_j)$ and $\bar{y}_j = \log(V_j \prod_i q_{ij})$.

To save you from the hassle of calculating $(\alpha_j, \beta_j)$'s, I uploaded a MATLAB script that solves the linear equation above and outputs $(\alpha_j, \beta_j)$'s. You can incorporate it into your own code, if you wish.