1. A set \( B = \{b_1, b_2, \ldots, b_n\} \) of rectangular boxes must be stored in a set of rectangular bins. Each box \( b_i \) has length \( h_i \) and width \( w_i \). The length \( h_i \) of a box is a positive number no larger than 1, i.e. \( 0 < h_i \leq 1 \), and its width \( w_i \) can be either 1 or 2. Each bin has length 1 and width 2. The bin-box-packing problem is to determine the minimum number of bins needed to store all the boxes in \( B \). This problem is NP-hard. For example, consider a set \( B = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8\} \) of boxes of heights 0.5, 0.5, 0.4, 0.3, 0.6, 0.6, 0.5, 0.5 and widths 2, 2, 2, 1, 1, 1, 1, 1, respectively. An optimum solution, which uses 3 bins, is shown below.

\[
\begin{array}{c}
\text{bin 1} & \text{bin 2} & \text{bin 3} \\
\begin{array}{c}
b_1 \\
b_2 \\
\end{array} & \begin{array}{c}
b_6 \\
b_3 \\
\end{array} & \begin{array}{c}
b_5 \quad b_8 \\
b_4 \quad b_7 \\
\end{array}
\end{array}
\]

- (5 marks) Write a polynomial time approximation algorithm for the problem with constant approximation ratio.
- (13 marks) Compute the approximation ratio of your algorithm. It must be constant.
- (3 marks) Compute the time complexity of your algorithm.

2. Let \( S \) be an array storing \( n \) different integer values; the array is not sorted. Consider the following algorithm for finding the \( k \)-largest value in \( S \), for a given value \( k \).

**Algorithm** select\((S, k, n)\)

**In:** Array \( S \) storing \( n \) different values, and value \( k \), \( 1 \leq k \leq n \)

**Out:** The \( k \)-largest value in \( S \)

**Repeat**

- Select a random value \( x \) from \( S \).
  - \( \text{larger} \leftarrow 0 \)
  - for \( i \leftarrow 1 \) to \( n \) do
    - if \( S[i] > x \) then \( \text{larger} \leftarrow \text{larger} + 1 \)
    - if \( \text{larger} = k - 1 \) then return \( x \)

(12 marks) Compute the expected running time of this algorithm. Assume that a random value \( x \) from \( S \) can be selected in \( O(1) \) time.
3. Consider a graph $G = (V, E)$ for which we want to color each node $u$ of $G$ with one of 4 possible colors: $c_2, c_3, c_4$. We say that an edge $(u, v)$ is satisfied if the colors assigned to $u$ and $v$ are different. We wish to assign colors to the nodes to maximize the number of satisfied edges. For example for the following graph, the number of satisfied edges (edges in bold) is 11.

- (5 marks) Write a randomized approximation algorithm that assigns a color to the nodes of $G$ in such a way that the expected number of satisfied edges is at least $\frac{3}{4}OPT$ where $OPT$ is the maximum number of edges that can be satisfied.
- (12 marks) Show that the expected number of satisfied edges is at least $\frac{3}{4}OPT$. 