1. a) We start by multiplying the second constraint by −1 so that the right hand side of all constraints is nonnegative. Then we introduce slack variables \( x_4 \) and \( x_5 \) to the second and the third constraints to put the problem on canonical form. The slack variable \( x_4 \) can be used as a basic variable for a first basic feasible solution, but we need two more basic variables, and so we introduce two artificial variable \( y_1 \) and \( y_2 \) in the second equation and third equation. (It would also have been correct to use three artificial variables, one for each constraint.) The objective is then to minimize \( y_1 + y_2 \) subject to these constraints or, equivalently, to maximize \( z = -y_1 - y_2 \). The LP problem that should be solved in phase 1 is therefore

maximize \( z = -y_1 - y_2 \)
subject to \[
\begin{align*}
-3x_1 & - x_2 + 2x_3 + x_4 = 3, \\
-x_1 + x_2 - 3x_3 & - x_5 + y_1 = 1, \\
4x_1 - 2x_2 + x_3 & + y_2 = 5, \\
x_1, x_2, x_3, x_4, x_5, y_1, y_2 & \geq 0,
\end{align*}
\]

In the tableau, we need \( z \) to be expressed in terms of the nonbasic variables \( x_1, x_2, x_3 \) and \( x_5 \), and so the equation in the objective row should not be \( z + y_1 + y_2 = 0 \), but instead \( z - 3x_1 + x_2 + 2x_3 + x_5 = -6 \). The first tableau is therefore

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( y_1 )</th>
<th>( y_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_4 )</td>
<td>(-3)</td>
<td>(-1)</td>
<td>(2)</td>
<td>(1)</td>
<td>(0)</td>
<td>(0)</td>
</tr>
<tr>
<td>( y_1 )</td>
<td>(-1)</td>
<td>(1)</td>
<td>(1)</td>
<td>(0)</td>
<td>(-1)</td>
<td>(0)</td>
</tr>
<tr>
<td>( y_2 )</td>
<td>(4)</td>
<td>(2)</td>
<td>(1)</td>
<td>(0)</td>
<td>(0)</td>
<td>(1)</td>
</tr>
<tr>
<td>( z )</td>
<td>(-3)</td>
<td>(1)</td>
<td>(2)</td>
<td>(0)</td>
<td>(1)</td>
<td>(0)</td>
</tr>
</tbody>
</table>

b) We have to choose \( x_1 \) as the incoming variable, and then \( y_2 \) has to be an outgoing variable. The next tableau is then

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( y_1 )</th>
<th>( y_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_4 )</td>
<td>(0)</td>
<td>(-5/2)</td>
<td>(11/4)</td>
<td>(1)</td>
<td>(0)</td>
<td>(0)</td>
</tr>
<tr>
<td>( y_1 )</td>
<td>(0)</td>
<td>(1/2)</td>
<td>(-11/4)</td>
<td>(0)</td>
<td>(-1)</td>
<td>(1)</td>
</tr>
<tr>
<td>( x_1 )</td>
<td>(1)</td>
<td>(-1/2)</td>
<td>(1/4)</td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
</tr>
<tr>
<td>( z )</td>
<td>(0)</td>
<td>(-1/2)</td>
<td>(11/4)</td>
<td>(0)</td>
<td>(1)</td>
<td>(0)</td>
</tr>
</tbody>
</table>

c) Yes, since there is a solution with the artificial variables being 0, there is a feasible solution for the original problem, and it is given by the nonbasic variables \( x_3 = 0, x_5 = 0 \) and the basic variables \( x_1 = 7/2, x_2 = 9/2, x_4 = 18 \). The first three rows of the tableau will be the same as the final tableau from phase 1 except that the \( y_1 \) and \( y_2 \) columns will be removed. The objective row comes from \( z = x_1 - 2x_2 - 4x_3 \), when expressed in only the nonbasic variables \( x_3 \) and \( x_5 \), i.e. \( z = -\frac{25}{2} x_3 - 3 x_5 - \frac{11}{2} \). The first tableau of phase 2 is therefore
d) Since all the coefficients in the objective row are nonnegative, the optimality condition is satisfied in the tableau above. We have found an optimal solution \( x = (7/2, 9/2, 0, 18, 0)^T \) with the optimal value \( z = -11/2 \).

2. a) As the primal problem \((P)\) is given by

\[
\begin{align*}
\text{maximize} \quad & z = c^T x, \\
\text{subject to} \quad & Ax \leq b, \\
& x \geq 0,
\end{align*}
\]

the dual problem \((D)\) is given by

\[
\begin{align*}
\text{minimize} \quad & v = b^T y, \\
\text{subject to} \quad & A^T y \geq c, \\
& y \geq 0,
\end{align*}
\]

Suppose that \( x \) is feasible for \((P)\) and \( y \) is feasible for \((D)\). Then we have (using the feasibility conditions of the respective problems) that

\[
c^T x \leq (A^T y)^T x = y^T Ax \leq y^T b = b^T y.
\]

b) Assume that \((P)\) is unbounded. Then there exists a sequence \( x_j \geq 0 \) such that \( Ax_j \leq b \) and \( c^T x_j \to \infty \) as \( j \to \infty \). From subproblem a) we get that every feasible solution \( y \) to \((D)\) satisfies \( b^T y \geq c^T x_j \) for all \( j \geq 1 \), but since \( c^T x_j \to \infty \) as \( j \to \infty \) such \( y \) cannot exist, and so the feasible set is empty.

3. a) Let \( x_1 \) be the number of bowls and \( x_2 \) the number of vases that Helen makes per week. Then the ILP problem can be stated as

\[
\begin{align*}
\text{maximize} \quad & z = 50x_1 + 40x_2 \\
\text{subject to} \quad & \begin{cases} 
3x_1 + 2x_2 \leq 20 \\
2x_1 + 5x_2 \leq 35 \\
x_1, \quad x_2 \geq 0, \text{ integers.}
\end{cases}
\end{align*}
\]

b) The feasible set is plotted together with some level curves (the dashed lines), with the optimal one in blue. It is clear that the optimum is at the intersection point of the lines \( 3x_1 + 2x_2 = 20 \) and \( 2x_1 + 5x_2 = 35 \), i.e. at \((30/11, 65/11)\). The optimal value is \(4100/11\).
c) We choose $x_1$ as a branching variable, and will explore the two cases $x_1 \leq 2$ and $x_1 \geq 3$ separately (since the $x_1$ value of the optimal solution of subproblem b) satisfies $2 < x_1 = 30/11 < 3$). The (noninteger) optimal solutions are illustrated below and are in the two cases given by $(2, 31/5)$ with value 348 (left figure below) and $(3, 11/2)$ with value 370 (right figure), respectively.

The branch $x_1 \geq 3$ has a larger value and therefore it seems more promising. We will use $x_2$ as our next branching variable, and explore the two cases $x_2 \leq 5$ and $x_2 \geq 6$ separately (since the previous optimum has an $x_2$-value in the interval $(5, 6)$). Clearly, the feasible set for the $x_2 \geq 6$ branch is empty, and so it suffices to explore the branch $x_2 \leq 5$. A graphical solution gives that the (noninteger) optimum is attained at $(10/3, 5)$ with value $1100/3$. 
We continue with this branch as it is more promising than the dangling node with optimal value 348. Now \( x_1 \) is a branching variable and we have the two branches \( x_1 \leq 3 \) and \( x_1 \geq 4 \). We plot both the domains below (the first domain is the line segment \( x_1 = 3, \ 0 \leq x_2 \leq 5 \)). We find an integer optimum in each branch: \((3, 5)\) with value 350 in the left branch and \((4, 4)\) with value 360 in the right branch.

The integer optimum value for the right branch is clearly the largest of these. It is also larger than the noninteger value of the dangling node (with value 348) found earlier. The dangling node can therefore be cut off, and \((4, 4)\) is the optimal solution with the optimal value 360. So Helen should make 4 bowls and 4 vases each week.

The tree that was used in the solution is given below. The numbers in the nodes are the upper bounds for the ILP problem of that branch (obtained from taking the integer part of the optimal value of the corresponding LP problem), with the convention that the maximum of the empty set is \(-\infty\).

4. a) We start but letting \( N_r \) consist of the origin node \( A \). For every \( k \neq A \), we let the temporary node price \( \nu_k = \infty \). We organize the work in the following table, where the distances from node \( A \) to all other nodes are given by the node prices \( y_k \).
Step | $N_r$ | New $\nu_k$ | New $q_k$ | $y_k$ | $p_k$ | $k$  \\
--- | --- | --- | --- | --- | --- | ---  \\
1 | $\{A\}$ | $\nu_B = 15$ | $q_B = A$ | $y_D = 5$ | $p_D = A$ | $k = D$  \\
2 | $\{A, D\}$ | $\nu_E = 9$ | $q_E = D$ | $y_E = 9$ | $p_E = D$ | $k = E$  \\
3 | $\{A, D, E\}$ | $\nu_C = 12$ | $q_C = E$ | $y_I = 23$ | $p_I = E$ | $k = I$  \\
4 | $\{A, D, E, F\}$ | $\nu_H = 27$ | $q_H = F$ | $y_I = 23$ | $p_I = E$ | $k = I$  \\
5 | $\{A, C, D, E, F\}$ | $\nu_B = 14$ | $q_B = C$ | $y_B = 14$ | $p_B = C$ | $k = B$  \\
6 | $\{A, B, C, D, E, F\}$ | $y_G = 18$ | $q_G = C$ | $y_H = 25$ | $p_H = G$ | $k = H$  \\
7 | $\{A, B, C, D, E, F, G\}$ | $\nu_H = 25$ | $q_H = G$ | $y_I = 23$ | $p_I = E$ | $k = I$  \\
8 | $\{A, B, C, D, E, F, G, H, I\}$ | $\nu_B = 14$ | $q_B = C$ | $y_B = 14$ | $p_B = C$ | $k = B$  \\
9 | $\{A, B, C, D, E, F, G, H, I\}$ | $y_G = 18$ | $q_G = C$ | $y_H = 25$ | $p_H = G$ | $k = H$  \\

b) The graph is not acyclic, and so Bellman’s equations cannot be used.

5. By labelling nodes according to the Ford-Fulkerson method, we find for example the following paths (starting from the trivial flow where all $x_{ij} = 0$):

<table>
<thead>
<tr>
<th>Step</th>
<th>Path</th>
<th>Increase of flow value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$A \to B \to H$</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>$A \to C \to F \to H$</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>$A \to D \to E \to F \to H$</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$A \to D \to E \to G \to H$</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>$A \to D \to I$</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>$A \to C \to D \to I$</td>
<td>7</td>
</tr>
</tbody>
</table>

The value of the resulting flow is $12 + 6 + 1 + 3 + 1 + 7 = 30$. The updated network with the final excess capacities is given below:

![Flow Network](image)

When using the Ford-Fulkerson algorithm on this updated network, we find that we can only label the node $B$, and so we define the cut $C$ to be the edges from
A or B to any other node in the original network. The capacity of this cut is 
5 + 13 + 12 = 30. We have found a cut with the same capacity as the value of our 
flow. The flow must therefore be maximal, by the max flow-min cut theorem.

6. To put the problem in the form of the assignment problem we need to have the same 
number of jobs as there are people. In this case, there will be one person without a 
job, and so we introduce a fictitious job which corresponds to unemployment. Doing so 
results in an extra column in the cost matrix whose entries should all be zero, since if 
there was a task corresponding to unemployment on the test, then all the applicants can 
be assumed to perform it perfectly.
The task that cannot be performed by one of the applicants corresponds to the corre-
sponding entry being ∞ (or a very large number). The first table for the assignment 
problem is then given by the first table below, and we modify it right away by replacing 
each column by the difference of that column and the smallest number of the same 
column. The result is the second table below. We also start putting ∗ on some zeros of 
the second matrix according to step 2 of the Hungarian algorithm.

\[
\begin{array}{cccccc}
16 & 4 & 17 & 3 & 0 \\
13 & 14 & 8 & 11 & 0 \\
2 & 19 & \infty & 9 & 0 \\
21 & 12 & 13 & 16 & 0 \\
22 & 16 & 25 & 12 & 0 \\
\end{array}
\quad \sim \quad
\begin{array}{cccccc}
14 & 0^* & 9 & 0 & 0 \\
11 & 10 & 0^* & 8 & 0 \\
0^* & 15 & \infty & 6 & 0 \\
19 & 8 & 5 & 13 & 0^* \\
20 & 12 & 17 & 9 & 0 \\
\end{array}
\]

We managed to star all rows except for the last one. According to step 3 of the algorithm, 
we would like to exchange the 0 in the last row with a 0*. But then we would need to 
find another 0 in the fourth row, and there are none. Hence we need to mark the last 
column as necessary. We draw a line through it below and also through all necessary 
rows, i.e. the rows that have a 0* in a column which is not necessary (i.e. without a line 
through it). The result is shown below:

\[
\begin{array}{cccccc}
14 & 0^* & 9 & 0 & 0 \\
11 & 10 & 0^* & 8 & 0 \\
0^* & 15 & \infty & 6 & 0 \\
19 & 8 & 5 & 13 & 0^* \\
20 & 12 & 17 & 9 & 0 \\
\end{array}
\]

Next, we go to step 4 of the algorithm, and note that the smallest uncovered entry is 
5. According to a theorem, we can replace the cost matrix by the matrix that is formed 
when subtracting 5 from all uncovered entries and adding 5 to all entries that are covered 
by two lines. We obtain the matrix below, where all the stars have also been removed. 
We immediately start putting new stars in, and this is done in the same (left) matrix 
below. We cannot star a zero in the last row, and so we try to make a chain of 0 and 0* 
that we can swap (see the right matrix below):

\[
\begin{array}{cccccc}
14 & 0^* & 9 & 0 & 5 \\
11 & 10 & 0^* & 8 & 5 \\
0^* & 15 & \infty & 6 & 0 \\
14 & 3 & 0 & 8 & 0^* \\
15 & 7 & 12 & 4 & 0 \\
\end{array}
\quad \sim \quad
\begin{array}{cccccc}
14 & 0^* & 9 & 0 & 5 \\
11 & 10 & 0^* & 8 & 5 \\
0^* & 15 & \infty & 6 & 0 \\
14 & 3 & 0 & 8 & 0^* \\
15 & 7 & 12 & 4 & 0 \\
\end{array}
\]
We cannot star a zero in the last row, and so we try to make a chain of 0 and 0* that we can swap. Clearly, the chain cannot continue further, and so we must mark the third column as necessary. When backtracking the path, we find that the last column is necessary too (since there are no other 0 in the third row to the right of column 5). There are two necessary rows, and these are the first and the third ones. See the left matrix below. The matrix is modified according to the same principles as above (the value of the smallest uncovered entry is subtracted from the uncovered entries, and the same number is added to the twice covered entries), resulting in the right matrix below, where we have removed the old stars and put new ones according to the principles of step 2:

\[
\begin{array}{ccc|c|c|c|c|c|c|c|c|c|c|c|c}
14 & 0^* & 4 & 0 & 4 & 14 & 0^* & 12 & 0 & 8 \\
11 & 10 & 0^* & 8 & 5 & 8 & 7 & 0^* & 5 & 5 \\
0^* & 15 & \infty & 6 & 0 & 0^* & 15 & \infty & 6 & 8 \\
14 & 3 & 0 & 8 & 0^* & 11 & 0 & 0 & 5 & 0^* \\
15 & 7 & 12 & 4 & 0 & 12 & 4 & 12 & 1 & 0
\end{array}
\]

\[\sim\]

\[
\begin{array}{ccc|c|c|c|c|c|c|c|c|c|c|c|c}
14 & 0^* & 4 & 0 & 4 & 14 & 0^* & 12 & 0 & 8 \\
8 & 7 & 0^* & 5 & 5 & 8 & 7 & 0^* & 5 & 5 \\
0^* & 15 & \infty & 6 & 8 & 0^* & 15 & \infty & 6 & 8 \\
11 & 0 & 0 & 5 & 0^* & 11 & 0^* & 0 & 5 & 0 \\
12 & 4 & 12 & 1 & 0 & 12 & 4 & 12 & 1 & 0^*
\end{array}
\]

As we cannot put a star on a 0 in the last row, we try to find a chain of 0 and 0* that can be swapped (left matrix), and this is done in the right matrix below:

\[
\begin{array}{ccc|c|c|c|c|c|c|c|c|c|c|c|c}
14 & 0^* & 12 & 0 & 8 & 14 & 0^* & 12 & 0^* & 8 \\
8 & 7 & 0^* & 5 & 5 & 8 & 7 & 0^* & 5 & 5 \\
0^* & 15 & \infty & 6 & 8 & 0^* & 15 & \infty & 6 & 8 \\
11 & 0 & 0 & 5 & 0^* & 11 & 0^* & 0 & 5 & 0 \\
12 & 4 & 12 & 1 & 0 & 12 & 4 & 12 & 1 & 0^*
\end{array}
\]

We have found a permutation where the ones are corresponding to the 0* entries in the last cost matrix above. Consequently, applicant A1 should take job number D, applicant A2 should take job number C, applicant A3 should take job number A, applicant A4 should take job number B, and applicant A5 will be the one who will not be employed.