Question 1

a) Let u_{jk} be 1 if item type k exists in bin j and 0 otherwise. Let I_k be the set of items with type k.

We then add two new constraints:

$$\mathrm{M}u_{jk} \geq \sum_{i \in I_k} y_{ij} \qquad \forall j, k \tag{1}$$

$$\sum_k u_{jk} \geq 1 \qquad \forall j \tag{2}$$

$$u_{jk} \quad \text{binary} \tag{3}$$

where M is some sufficiently large number. (1) enforces the definition of u_{jk}. Because $\sum_k (u_{jk}) - 1$ counts the number of different types of items in bin j, (2) prevents empty bins from affecting the objective function.

The second objective function is then:

$$\mathrm{C}_2 = \sum_j \left(\left(\sum_k u_{jk} \right) - 1 \right)$$

b) $\mathcal{W} = 12$, $\mathcal{T} = 3$, $\lambda = 0.01$

Wastage is integer, so the smallest distance between wastage objective values is 1.
The highest possible value for $\mathcal{T}$ is (number of bins) * (number of item types - 1), which corresponds to the allocation where all bins contains all item types. For problem data 1, this is $10 * 4 = 40$ as max(number of bins) = (number of items). The lowest possible value is 0, which corresponds to the allocation where one bin contains all items. This means the greatest difference in $\mathcal{T}$ is 40.

Therefore it is sufficient (but not necessary) that $1 > \lambda * 40 \rightarrow \lambda < \frac{1}{40}$ so the first objective will always be a priority over the second objective. $\lambda = 0.01$ was chosen as it fulfills the above condition and is also not so small as to induce numerical error.

c) The epsilon-constraint method was used to solve this bi-objective integer problem as it will find all efficient solutions, even the non-supported ones. I assume the set of efficient solutions is not very large - that it can be enumerated within a reasonable time-frame.

The IP for minimising wastage only is solved, and $\mathcal{T}$ is recorded. $\mathcal{T}$ is then constrained to be less than the previous solution's $\mathcal{T}$ and the IP is re-solved. This repeats until $\mathcal{T}$ is negative or the problem becomes infeasible. The solutions found make up the set of efficient solutions for the bi-objective problem.

The implementation can be run via `> python bin_pack_MO_bi.py` with no supporting files needed.

The objective function values for all efficient (not including weakly efficient) solutions are:

W	T
0	7
25	3
50	2
75	1
100	0

The solution that minimises waste ($\mathcal{W} = 0$):

Bin	Items
1	3, 15, 16
2	6, 8, 9, 12
3	2, 7, 11
4	1, 20
5	5, 10, 17, 18
7	4, 13, 14, 19

The next solution ($\mathcal{W} = 25$):

Bin	Items
1	2, 9, 10, 12, 18
2	3, 5, 13, 14
3	1, 6
4	11, 19, 20
5	15, 16
6	4, 8, 17
7	7

d) Introduce another objective C_3 that minimises the maximum number of different types of items in each bin.

This is done by adding a new variable:

$$m \geq \sum_k u_{jk} \quad \forall j$$

with objective function:

$$\mathrm{C}_3 = \text{minimise } m$$

We can then redefine $\mathrm{C}_{2,\text{new}} = \text{lexmin}\{\mathrm{C}_{2,\text{old}}, \mathrm{C}_3\}$ and solve as above.