Problem 1

a) In the worst case, the number of edge-swaps required is the number of edges in any spanning tree of the graph which equals V-1, where V is the number of vertices. Because intuitively, if T shares no common edge with T', then this implies T has to swap all of its edges in order to become T'.

So to perform the swaps, create a set E_{common} which contains all common edges between T and T', then repeat the following steps: copy any edge in T' that's not in E_{common}, add it to E_{common}, then add it to T to create a cycle, then remove an edge in this cycle that does not belong to E_{common}.

The goal is to increase E_{common} so it would eventually contain all edges of T/T'.

b) Similar as question a), create a set E_{common} which contains all common edges between T and T', and repeat the following steps: copy the max-weighted edge in T' that's not in E_{common}, add it to E_{common}, then add it to T to create a cycle, then remove an edge in this cycle that does not belong to E_{common}.(Note the edge removed in this cycle should be greater than or equal to the edge that's added in this cycle, otherwise, the resulting spanning tree T' is not a MST)

c) According to question b), this algorithm returns a MST, and according to a), the worst case is when T and T' share no common edge at all, and all spanning trees have to be checked to find the MST. The number of spanning trees can be calculated by choosing V-1 edges from E, so it's $C(E, V-1) = \frac{E!}{(V-1)!}$

Problem 2

a) There are two cases for any optimal solution, the first case is when all the bins are exactly filled by elements at full capacity, in this case, the number of bins is exactly $\frac{S}{B}$. The second case is a general case when there are some bins that are not fully used, in this case, let the number of bins be x, then S must be less than x*B, then S < x*B $\rightarrow \frac{S}{B} <$ x, since $\frac{S}{B}$ might not be integers, then $\frac{S}{B} <$ x $\rightarrow \lceil \frac{S}{B} \rceil \leq x$. Therefore, two cases combined gives: the optimum number of bins required is at least $\lceil \frac{S}{B} \rceil$

b) The first-fit algorithm always tries to fill a bin as long as it can be filled, so if there are more than one bin that are less than half full, then the algorithm will try to merge them to reduce the number of bins used(according to the nature of FF algorithm), so it's impossible to have more than one bin that are less than half full.

c) Consider the worst case where every single bin is only filled with one element which occupies at least half space (Note FirstFit algorithm leaves at most one bin less than half full).

let the number of bins required to be x, then S $\geq \frac{B*x}{2}$, and this equation can be rewrote as $x \leq \frac{2S}{B}$, since $\frac{2S}{B} \leq \lceil \frac{2S}{B} \rceil$, we can conclude $x \leq \lceil \frac{2S}{B} \rceil$.

d) since the optimum solution is bounded by $\lceil \frac{S}{B} \rceil$ according to question a) and the worst case solution is bounded by $\lceil \frac{2S}{B} \rceil$ according to question c), then it's obvious that the approximation ratio for First-Fit algorithm is $\frac{\lceil \frac{2S}{B} \rceil}{\lceil \frac{S}{B} \rceil} = 2$

Problem 3

Reduce this problem to the Rudrata Cycle introduced in DPV 8.3.

The Zero Weight Cycle problem is NP, because if we go through all the edges in this cycle, it's easy to check if the edge weights sum up to zero.

Build a graph G': let the number of vertices be V, assign one edge in G' with a weight of V-1, and assign the rest edges with weight of -1.

When G' has a solution that contains the edge with weights of V-1, then this implies that the sum of edge weights in this solution is $V-1+(-1)*(V-1)=0$, thus it has a solution that contains a zero weight cycle.

Similarly, to show " when there's no Rudrata Cycle solution to G', then there's no Zero Weight Cycle solution to G' ", we just need to prove the contrapositive, that is to show if there is a Zero Weight Cycle in G', then there is a Rudrata Cycle in G'. So if Zero Weight Cycle returns yes for G', then this implies the edge with weight V-1 is included (given that the rest edges all have weight -1), and so the rest of the edges must all be included too, this then implies all vertices are visited exactly once, thus we have a Rudrata Cycle solution for G'.

Since Rudrata Cycle is NP-Complete, and Zero Weight Cycle problem can be reduced to Rudrata Cycle, then we can conclude that the Zero Weight Cycle problem is also NP-Complete.

Problem 4

a) Denote the error equation as ± error(so two values for error because the absolute sign is removed).

```
min: error;
cl1: a + 3 b - c  -error <= 0;
     a + 3 b - c + error >= 0;
     2 a + 5 b - c - error <= 0;
     2 a + 5 b - c + error >= 0;
     3 a + 7 b - c -error <= 0;
     3 a + 7 b - c + error >= 0;
     5 a + 11 b - c - error <= 0;
     5 a + 11 b - c + error >= 0;
     7 a + 14 b - c -error <= 0;
     7 a + 14 b - c +error >= 0;
     8 a + 15 b - c -error <= 0;
     8 a + 15 b - c + error >= 0;
     10 a + 19 b - c - error <= 0;
     10 a + 19 b - c + error >= 0;
     a >= 0;
     b >= 0;
     c >= 0;
     error >= 0;
```

b) let t, l, s, c, o represent tomato, lettuce, spinach, carrot, oil.

```
min: 21 t + 16 l + 371 s + 346 c + 994 o;
cl1: 0.85 t + 1.62 l + 12.78 s + 8.39 c >= 15;
     0.33 t + 0.2 l + 1.58 s + 1.39 c + 100 o >= 2;
     0.33 t + 0.2 l + 1.58 s + 1.39 c + 100 o <= 6;
     4.64 t + 2.37 l + 74.69 s + 80.70 c >= 4;
     9 t + 8 l + 508.2 c + 100 o <= 100;
     l + s <= 0.5 t + 0.5 l + 0.5 s + 0.5 c + 0.5 o;
     t >= 0;
     l >= 0;
     s >= 0;
     c >= 0;
     o >= 0;
```

The running result is shown below:

Results:

```
Value of objective function: 232.51469403393088
t = 5.884801497699927
l = 5.843176036801243
s = 0.041625460898685684
c = 0.0
o = 0.0
Using Simplex Dual
```