Candidate number: 2031B
Paper 1
Question 7

a.

i.

$$f_D(n) = \begin{cases} \{\} \text{ if } n = 0 \\ \text{null if } \nexists d \in D : d \leq n \\ \text{mincard}(f_D(n-d) \cup \{d\}, f_{D\setminus\{d\}}(n)) \text{ where } d = \min(\{d \in D : d \leq n\}) \text{ otherwise} \end{cases}$$

where

$$\text{mincard}(a, b) = \begin{cases} a \text{ if } b = \text{null } \vee |a| < |b| \\ b \text{ otherwise} \end{cases}$$

This function returns the minimum-cardinality multiset of elements from the set D whose sum is n. It works by taking the stamp d which is the largest stamp in D which is less than or equal to n. It finds the optimal solution which includes d, and the optimal solution which does not include d, and then returns whichever of those has the lowest cardinality. The degenerate cases are when a solution does not exist (in which case "null" is returned), and when n=0 (in which case the empty set is returned).

ii. The data structure would be a table (or two-dimensional array). Along one axis would be all the integers from 1 to n inclusive. Along the other axis would be all the subsets of D, including D but not the empty set. In each cell would be the minimum-cardinality multiset of stamps from the subset of D whose sum is the integer along the first axis. An example is given below for n=5 and D={1,2,3}

D\n	*5*	*4*	*3*	*2*	*1*
{1,2,3}	{2,3}			{2}	
{1,2}	{1,2,2}		{1,2}		{1}
{1,3}				{1,1}	{1}
{2,3}					
{1}	{1,1,1,1,1}	{1,1,1,1}	{1,1,1}	{1,1}	{1}
{2}					
{3}					

iii. The time complexity is $\theta(n|D|)$. The space complexity is $\theta(n2^{|D|})$

b.

i.

$$f(r, k) = \left\{ \begin{array}{c} \text{null if } r < 0 \\ \{\} \text{ if } k = n \\ \text{mincard}(f(l, k+1) \cup \{s_k\}, f(r - d_k, k+1) \text{ otherwise}) \end{array} \right\}$$

where

$$\text{mincard}(a, b) = \begin{cases} a \text{ if } b = \text{null } \vee |a| < |b| \\ b \text{ otherwise} \end{cases}$$

Candidate number: 2031B
Paper 1
Question 7

This function takes as its parameters the remaining distance r which can be travelled without refuelling, and the index k of the station from which we are departing. It would initially be called with f(l, 0) to find the solution to the problem. It works by considering the optimal solution in which we choose to refuel at the next station, and the optimal solution in which we do not, and then returns whichever consists of the fewest stops. The degenerate cases are if we run out of fuel, or if we reach B (at which point k=n).

ii. The data structure would be a table (or two-dimensional array). Along the first axis would be the value of k from 0 to n inclusive, and along the other axis would be the values of r from 0 to l inclusive. Each cell would contain the optimal solution travelling from station k to B, starting with an amount of fuel which could travel a distance r. Below is an example for n=4 and l=6

r\k	*0*	*1*	*2*	*3*	*4*
6					
5					
4					
3					
2					
1					
0					

iii. The time complexity is θ(nl). The space complexity is θ(nl)

c. The stamp problem would be better to solve with a greedy algorithm. The greedy algorithm would be at each step to choose the largest-valued stamp which does not exceed the remaining value needed.

The time complexity is θ(n).