Shane Califano

Question 1

A) _

$$f(n) = \min\{F(n-d_j)\} + 1 \text{ for } n > 0$$
$$F(0) = 0$$

$n = 9$
$m = 3$

i	math	f(i)
0	$F(0) = 0$	0
1	$F(1) = F(1-1)+1 = 1$	1
2	$F(2) = \min(F(2-1)+1, F(2-2)+1) = 1$	1
3	$F(3) = \min(F(3-1)+1, F(3-2)+1) = 2$	2
4	$F(4) = \min(F(4-1)+1, F(4-2)+1) = 2$	2
5	$F(5) = \min(F(5-1)+1, F(5-2)+1, F(5-5)+1) = 1$	1
6	$F(6) = \min(F(6-1)+1, F(6-2)+1, F(6-1)+1) =$	2
7	$F(7) = \min(F(7-1)+1, F(7-2)+1, F(7-5)+1) = 2$	2
8	$F(8) = \min(F(8-1)+1, F(8-2)+1, F(8-5)+1) = 3$	3
9	$F(9) = \min(F(9-1)+1, F(9-2)+1, F(9-5)+1) = 3$	3

$F(9) = 3$

B) _

```
F[0] = 0        // sets the base case

FOR i FROM 1 TO N:          // sets a default value to infinity to everything
    F[i] = inf

FOR i FROM 1 TO N:          // loops n times
    FOR EACH coin IN denominations:     // checks the formula with each coin type
            IF i >= coin:
                    F[i] = min(F[i], F[i – coin] + 1)
IF F(n) == inf:
    RETURN -1       // there is no value
ELSE
    RETURN F(n)     // returns the value
```

C) _

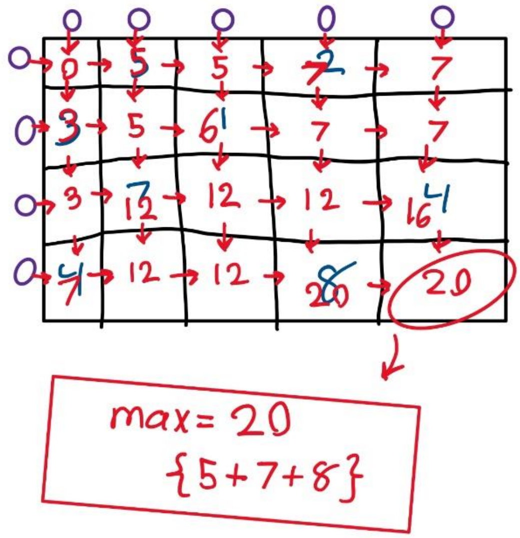
Time Complexity: O(MN) – N [the target amount], M [# of coin denominations]
Space Complexity: O(N) – N [the target amount]

The time complexity is O(MN) because it iterations N amount of times, and in each iteration, it loops M times to shuffle through the dominations.

The space complexity is O(N) because we store a table of size N.

Question 2

A) _



B) _

F[i][j] = max(F[i-1][j], F[i][j-1]) + current_cell_value

We essentially take the greater of the cell above it or behind it and add the value of the current cell (if any).

Question 3

A) _

X = {C, T, A, A, T, G, G, A}

Y = {T, A, G, A, C, T, G}

		T	A	G	A	C	T	G
	0	0	0	0	0	0	0	0
C	0	0	0	0	0	1	1	1
T	0	1	1	1	1	1	2	2
A	0	1	2	2	2	2	2	2
A	0	1	2	2	3	3	3	3
T	0	1	2	2	3	3	4	4
G	0	1	2	3	3	3	4	5
G	0	1	2	3	3	3	4	5
A	0	1	2	3	4	4	4	5

max = 5

B) _

$$L[i][j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ L[i-1][j-1] + 1 & \text{if } X[i-1] = Y[j-1] \\ \max(L[i-1][j], L[i][j-1]) & \text{if } X[i-1] \neq Y[j-1] \end{cases}$$

C) _

Time Complexity: O(MN) – M [Length of X], N [Length of Y]

We run a time complexity of O(MN) because we iterate M x N times to fill in the table used for the dynamic programming approach