Task 1:

Base Case: When $N = 1$, the logic returns a movement of disc 1 from *s* to *d*, such that $s,d \in \{0,1,2\}$ and s does not equal d.

Hypothesis: The logic returns the sequence that moves the stack n – 1 discs (i.e. disc 1, 2, … disc n – 1) from s to d, where $s,d \in \{0,1,2\}$ and s does not equal d, such that $n - 1 \geq 1$

Induction: For this example, consider moving a stack of n discs using the recursion, where $n \geq 2$. By the hypothesis, the call to **makeMoves**(n – 1, s, t, d) moves disc 1, disc 2, …, disc n – 1 from posts s to t, post s being left of disc n, which is moved to post d, which is currently empty. As such, we moved disk n, the largest disc, to its destination post. Then, by the hypothesis, the call to **makeMoves**(n – 1, t, d, s) moves disc 1, 2, …, n – 1 from post t to post d. As the discs 1, 2, …, n – 1 are smaller than disc n, the moves made by **makeMoves**(n – 1, t, d, s) are valid for the fact that disk n is currently on post d. All discs are moves to post d, thus we are done.

Task 2:

Base Case: When n = 1, the logic returns a single movement of the first disc from s to *d*, correct due to it starting with a movement of disc 1, when $s,d \in \{ 0,1,2\}$ and *s* does not equal *d*.

Hypothesis: Returns a sequence beginning with a movement of disc 1 for moving a stack of $n - 1$ discs from *s* to *d*, where $s,d \in \{ 0,1,2\}$ and *s* does not equal *d*.

Induction: If moving a stack of *n* discs using this logic, that is where $n \geq 2$, said logic beings with a call to **makeMoves**($n - 1$, *s,t,d*), then by our hypothesis it will start with a movement of disc 1.

Task 3:

Base Case: When n = 1, the logic returns a single movement of disc 1 from S to D, such that it ends with a move of disk 1, where $s,d \in \{ 0,1,2\}$ and *s* does not equal *d*.

Hypothesis: The logic returns a correct sequence that ends with a move of disc 1 or moving a stack of $n - 1$ disks from *s* to *d*, where $s,d \in \{ 0,1,2\}$ and *s* does not equal *d*, note that $n - 1 \geq 1$.

Induction: Consider moving a stack of *n* discs using the recursive logic, where $n \geq 2$. The logic ends with a call to **makeMoves**($n - 1$, t, d, s), and by the induction hypothesis, it ends with a move of disc 1.

Task 4:

Base cases:

When n = 1, the logic returns a single move of disc 1 from s to d, which is correct since there are no other discs in this scenario other than disc 1.

When n = 2, the logic returns 3 movements of disc 1, 2, and 1 respectively. This is correct since the movement of disc 2 is surrounded by to movements of disc 1.

Hypothesis: For moving a stack of n – 1 discs using our recursive logic, where $n \geq 3$, the logic returns a sequence such that every move of a disc other than disc 1 is surrounded before and after by two moves of disc 1.

Induction: Consider moving a stack of n discs with the recursion, such that $n \geq 3$, and a move of disc *k* when *k* > 1, for this solution.

First Case: The movement of disc k is a move in the solution of the recursive call to **makeMoves**(n – 1, s, t, d). By our hypothesis, the movement of disc k is surrounded by two moves of disc 1 in the sequence returned by **makeMoves**(n – 1, s, t, d).

Second Case: The movement of disc k is a move generated in step two of the recursion. As the moved comes after the call to **makeMoves**(n – 1, s, t, d), this last move, by the result of Task 2, is movement of disc 1. Disc k's movement is also followed by a call to **makeMoves**(n – 1, t, d, s), and this first move, by Task 3's logic, is a movement of disc 1. As such, the movement of disc k is surrounded by two movements of disc 1.

Thrid Case: The movement of disc k is a move in the solution of the call to **makeMoves**(n – 1, t, d, s). by our hypothesis, the movement of disc k is surrounded by two moves of disc 1 in the sequence returned by **makeMoves**(n – 1, t, d, s).

Task 5:

Base Cases:

When n = 1, the logic returns a single movement of disc 1 from s to d, which is correct as disc as disc 1 doesn't move twice.

When n = 2, the logic returns 3 moves, which are movements of disc 1, 2, and 1 respectively, such that disc 1 does not move consecutively.

Hypothesis: For moving a stack of n – 1 discs, where $n - 1 \geq 2$, the recursion returns a sequence such that disc 1 is not moved consecutively.

Induction: For this scenario, we move a stack of n discs using the recursion where $n \geq 3$, and a movement of disc 1, where k > 1.

First Case: The movement of disc 1 is a move in the solution of the recursive call to **makeMoves**(n – 1, s, t, d). By our hypothesis, the movement of disc 1 is not followed by another movement of disc 1 in the sequence returned by **makeMoves**(n – 1, s, t, d).

Second Case: The movement of disc 1 is a move by the solution of the call to **makeMoves**(n – 1, s, t, d). By the hypothesis, the move of disc 1 is not followed by another move of disc 1 in the sequence returned by **makeMoves**(n – 1, t, d, s).

Third Case: The last move returned by **makeMoves**(n – 1, s, t, d) is separated from the first move returned by **makeMoves**(n – 1, t, d, s) by a move of disc n which is not disc 1 as n $\geq$ 3.

Task 6:

Base Cases:

When n = 1, the logic returns a movement of disc 1 from s to d, which can be a cyclical sequence when moving forward.

When n = 2. The logic returns three moves, which moves disc 1 from s to t, disc 2 from s to d, and disc 1 from t to d respectively, which his correct as disc 1 is moved backward cyclically.

Hypothesis: For a stack of n – 1 discs, where n – 1 $\geq$ 2, the recursion returns a sequence such that disc 1 moves cyclically forward when n – 1 is odd, and backwards when n – 1 is even.

Induction: Consider moving a stack of n discs using the recursion such when n $\geq$ 3.

First Case: N is odd.

As n – 1 is even, disc 1 is moved cyclically backwards in the call to **makeMoves**(n – 1, s, t, d); that is, from s to d, d to t, t to s, etc, Disc 1 is also moved cyclically backwards in the call **makeMoves**(n – 1, t, d, s); that is, from s to d, d to t, t to s, etc, which is going forward from the view point of **makeMoves**(n, s, d, t).

Second Case: N is even.

As n – 1 is odd, disc 1 is moved cyclically forward in the call to **makeMoves**(n – 1, s, t, d); from s to t, t to d, d to s, etc. Disc 1 is also moved cyclically forward in the call to **makeMoves**(n – 1, t, d, s); from t to d, d to s, s to t. etc. In both sequences, disc 1 moved from s to t, t to d, d to s, etc, which is backwards cyclically from the view point from **makeMoves**(n , s, d, t).