# Student Information

Full Name : Mehmet Rüçhan Yavuzdemir
Id Number : 2522159

# Answer 1

To begin with, $x \in Z$, $5|x$ and $7|x \leftrightarrow 35|x$, since 5 and 7 are coprime. If we somehow get that for $n \in N^+, 6^{2n} - 1$ is divisible by 35, we are done.

p(n) $= n \in N^+$, $6^{2n} - 1$ is divisible by 35.
**Base case**: n = 1

$6^2 - 1 = 35$, which is obviously divisible by 35.

**Inductive Step**: Suppose that for some n = k, p(k) holds. Then, p(k+1) should also hold.

p(k) $= 6^{2n} - 1 = 35t$, $t \in Z$
$6^{2n} = 35t + 1$

p(k+1) $= 6^{2n+2} - 1$
$= 36 \cdot 6^{2n} - 1$
$= 36(35t + 1) - 1$
$= 36 \cdot 35t + 35$
$= 35(36t + 1)$

Hence, for all values of t, p(k+1) is divisible by 35, which means p(k+1) also holds. By mathematical induction, $n \in N^+$, $6^{2n} - 1$ is divisible by 35, and 5 and 7 simultaneously as explained above.

# Answer 2

**Base Cases**: $H_0 = 1$, $H_1 = 5$, $H_2 = 7$, $1 \leq 9^0$, $5 \leq 9^1$, $7 \leq 9^2$

**Inductive Step**: Let $k \geq 3$, $H_n = 8H_{n-1} + 8H_{n-2} + 9H_{n-3}$, $H_n \leq 9^n$ holds for all integers i such that $0 \leq i \leq k$.

$$H_{k+1} = 8H_k + 8H_{k-1} + 9H_{k-2}$$

The values k, k-1, and k-2 are in the interval $0 \leq i \leq k$. By Strong Induction, we can write the following equations:

$H_k \leq 9^k$
$H_{k-1} \leq 9^{k-1}$
$H_{k-2} \leq 9^{k-2}$

If we multiply both sides of each line with the values that are the coefficients at the recurrence relation above, we get:

$8H_k \leq 8 \cdot 9^k$
$8H_{k-1} \leq 8 \cdot 9^{k-1}$
$9H_{k-2} \leq 9 \cdot 9^{k-2}$

After adding all of them up:

$8H_k + 8H_{k-1} + 9H_{k-2} \leq 8 \cdot 9^k + 8 \cdot 9^{k-1} + 9 \cdot 9^{k-2}$

$H_{k+1} \leq 8 \cdot 9^k + 8 \cdot 9^{k-1} + 9 \cdot 9^{k-2}$

$H_{k+1} \leq 8 \cdot 9^k + 8 \cdot 9^{k-1} + 9 \cdot 9^{k-2} = 9^k(8 + 8/9 + 1/9) = 9^{k+1}$

$H_{k+1} \leq 9^{k+1}$

Hence, we verified that for given recurrence relation $H_n = 8H_{n-1} + 8H_{n-2} + 9H_{n-3}$, $H_n \leq 9^n$ for $n \in N$.

# Answer 3

Let me give me the algorithm and way of thinking first, then explain it. First, we should subtract the number of bit strings that don't contain "0000" from $2^8$. Second, we should subtract the number of bit strings that don't contain "1111" from $2^8$. Finally, we'll add them up but by the inclusion-exclusion method, we should subtract the intersection points, "11110000" and "00001111", which are counted twice.

Let f be the function that returns the number of strings with size n that doesn't contain the sequence "0000". We'll reduce the problem in each step, and then by using initial conditions, we'll complete our formula.

Let g be the function that returns the number of strings with size n that doesn't contain the sequence "1111". We'll reduce the problem in each step, and then by using initial conditions, we'll complete our formula.

If the first bit is "1", then the rest is again the number of strings with size n-1 that doesn't contain the sequence "0000".
If the first bits are "01" then the rest is again the number of strings with size n-2 that doesn't contain the sequence "0000".
If the first bits are "001" then the rest is again the number of strings with size n-3 that doesn't contain the sequence "0000".
If the first bits are "0001" then the rest is again the number of strings with size n-4 that doesn't contain the sequence "0000".

It seems that our recurrence relation becomes $\mathbf{f(n) = f(n\text{-}1) + f(n\text{-}2) + f(n\text{-}3) + f(n\text{-}4)}$

If the first bit is "0", then the rest is again the number of strings with size n-1 that doesn't contain the sequence "1111".
If the first bits are "10" then the rest is again the number of strings with size n-2 that doesn't contain the sequence "1111".
If the first bits are "110" then the rest is again the number of strings with size n-3 that doesn't contain the sequence "1111".
If the first bits are "1110" then the rest is again the number of strings with size n-4 that doesn't contain the sequence "1111".

It seems that our recurrence relation becomes $\mathbf{g(n) = g(n\text{-}1) + g(n\text{-}2) + g(n\text{-}3) + g(n\text{-}4)}$

$f(0) = 1$, the number of strings with size 0 that doesn't contain "0000" is 1, which is an empty string.
$f(1) = 2$, the number of strings with size 1 that doesn't contain "0000" is 2, which are "1" and "0".
$f(2) = 4$, the number of strings with size 2 that doesn't contain "0000" is 4, which is $2^2$, for each bit, we can either put 1 or 0.
$f(3) = 8$, the number of strings with size 3 that doesn't contain "0000" is 8, which is $2^3$, for each bit, we can either put 1 or 0.

$g(0) = 1$, the number of strings with size 0 that doesn't contain "1111" is 1, which is an empty string.
$g(1) = 2$, the number of strings with size 1 that doesn't contain "1111" is 2, which are "1" and "0".
$g(2) = 4$, the number of strings with size 2 that doesn't contain "1111" is 4, which is $2^2$, for each bit, we can either put 1 or 0.

g(3) = 8, the number of strings with size 3 that doesn't contain "1111" is 8, which is $2^3$, for each bit, we can either put 1 or 0.

f(0) = g(0) = 1
f(1) = g(1) = 2
f(2) = g(2) = 4
f(3) = g(3) = 8
f(4) = g(4) = 15
f(5) = g(5) = 29
f(6) = g(6) = 56
f(7) = g(7) = 108
f(8) = g(8) = 208

f(8) = 208 → the number of strings with size 8 that doesn't contain "0000".
g(8) = 208 → the number of strings with size 8 that doesn't contain "1111".
　　If the subtract f(8) and g(8) from the universe, which is $2^8$, 256.
256 - f(8) = 48 → the number of strings with size 8 that contain "0000".
256 - g(8) = 48 → the number of strings with size 8 that contain "1111".

　　However, there are two cases which are "11110000" and "00001111", and we count both of them twice. We should subtract duplicate cases. By the inclusion-exclusion method, 48 + 48 - 2 is our result, which evaluates to **94**.

# Answer 4

First, we should select our stars and habitable and non-habitable planets. There are $C(10,1) \cdot C(20,2) \cdot C(80,8)$ possibilities. After selecting the stars and planets for our galaxy, we can arrange their positions, and multiply all the possibilities.

　　Since there should be at least 6 non-habitable planets between habitable ones, we consider each case separately and add all of them up.

Case 6:

2! for habitable planets, $C(8,6) \cdot 6!$ for selecting and placing the non-habitable planets which ones are between the habitable planets. Finally, we will fix the 2 habitable and 6 habitable planet structure and the rest can be calculated as 3!

Case 7:

2! for habitable planets, $C(8,7) \cdot 7!$ for selecting and placing the non-habitable planets which ones are between the habitable planets. Finally, we will fix the 2 habitable and 7 habitable planet structure and the rest can be calculated as 2!

Case 8:

2! for habitable planets, $C(8,8) \cdot 8!$ for selecting and placing the non-habitable planets which ones are between the habitable planets. Finally, we will fix the 2 habitable and 8 habitable planet structure and the rest can be calculated as 1!


To sum up, the total number of possibilities that we have for forming a galaxy is

$$C(10,1) \cdot C(20,2) \cdot C(80,8) \cdot 2! \cdot (C(8,8) \cdot 8! \cdot 1! + C(8,7) \cdot 7! \cdot 2! + C(8,6) \cdot 6! \cdot 3!)$$

# Answer 5

**a)** First, we need to break the problem into pieces. We are given that there will be n cells that are passed, and the robot can jump one, two, or three cells away. So, after each jump, we'll reduce the problem, if we jump k cells away, n-k cells are left. At the beginning of the problem, we may jump 1, 2, or 3 steps. It's like a decision tree, we divide the problem and its subproblems into 3 pieces. Let f be the function that returns the number of ways to get to the nth cell. So we end up with this result:


**f(n) = f(n-1) + f(n-2) + f(n-3)**

**b)** Of course initial conditions matter. If we lack of them, there will be holes in our recurrence relation. We need f(1), f(2) and f(3).


f(1) = 1, [[1]]
f(2) = 2, [[1,1],[2]]
f(3) = 4, [[1,1,1],[1,2],[2,1],[3]]


**c)** By using the recurrence relation **f(n) = f(n-1) + f(n-2) + f(n-3)**,
f(1) = 1
f(2) = 2
f(3) = 4
f(4) = 7
f(5) = 13

f(6) = 24
f(7) = 44
f(8) = 81
f(9) = 149
Hence, f(9) = 149, there are 149 ways that the robot can use to move to 9 cells away.