1 Proposition Short Answer (16 Points)

a. Let $P(n)$ be a predicate where $n \in \mathbb{N}$. In each part, suppose you know the listed propositions are True. Is the given information enough to conclude that $(\forall n \in \mathbb{N})P(n)$? Answer Yes/No and briefly justify your answer (≤ 2 sentences). [2 pts each]

i. $P(0)$; $\quad (\forall n \in \mathbb{N})(\neg P(n+1) \implies \neg P(n))$.
Solution: Yes. This is standard induction, taking the contrapositive of the inductive step.

ii. $P(0), P(1)$; $\quad (\forall n \in \mathbb{N})((P(n) \vee P(n+1)) \implies (P(n+2) \vee P(n+3)))$.
Solution: No. Since we have $P(0)$ and $P(1)$, we can conclude that $P(2)$ or $P(3)$ is true. Then, this allows us to conclude $P(4)$ or $P(5)$ is true, and so on. However, we can never conclude for certain that $P(2)$ is true on its own, for instance.

iii. $P(0), P(1), P(2)$; $\quad (\forall n \in \mathbb{N})((P(n) \wedge P(n+1) \wedge P(n+2)) \implies P(n+3))$.
Solution: Yes. This is just a strong induction with three base cases.

b. For each $n \in \mathbb{N}$, let $P_n(x)$ be a predicate where x is in some nonempty universe U. Suppose we know that $(\forall x \in U)(P_n(x) \implies P_m(x))$ if and only if n divides m. For each of the following parts, label the statement as True, False, or not enough information, and briefly justify your answer (≤ 2 sentences). [2 pts each]

i. $(\exists x \in U)(P_2(x) \implies P_4(x))$.
Solution: True. We know $(\forall x \in U)(P_2(x) \implies P_4(x))$, and so in particular there exists $x \in U$ so $P_2(x) \implies P_4(x)$.

ii. $(\forall x \in U)(\neg P_3(x) \implies \neg P_9(x))$.
Solution: False. This is the contrapositive of $(\forall x \in U)(P_9(x) \implies P_3(x))$. Since 9 does not divide 3, there is some $x \in U$ for which the implication is false.

iii. $(\forall x \in U)((P_3(x) \vee P_4(x)) \implies P_{24}(x))$.
Solution: True. Since both 3 and 4 divide 24, for each $x \in U$, if either $P_3(x)$ or $P_4(x)$ is true, we know $P_{24}(x)$ is true.

iv. $(\exists x \in U)((\neg P_7(x) \vee \neg P_8(x)) \implies \neg P_4(x))$.
Solution: Not enough information. Taking the contrapositive and applying DeMorgan's law, the statement says $(\exists x \in U)(P_4(x) \implies P_7(x) \wedge P_8(x))$. While we do know that $P_4(x) \implies P_8(x)$ for all $x \in X$, we cannot relate $P_4(x)$ to $P_7(x)$.

v. $(\forall x \in U)((P_2(x) \implies P_4(x)) \implies P_3(x)) \implies (\exists y \in U)(\neg P_6(y))$.
Solution: False. For all $x \in U$, $P_2(x) \implies P_4(x)$ is true, so we assume $P_3(x)$ is true for all $x \in U$. But, $P_3(x) \implies P_6(x)$ for all $x \in U$, so there is no y for which $P_6(y)$ is false.

2 Proofpourri! (18 Points)

a. Suppose we have three piles of stones, each with a positive integer number of stones. Prove that we can always pick two piles to combine such that the combined pile has an even number of stones. [6 pts]

Solution: The only possibilities are that we have 3 even piles, 3 odd piles, 2 even piles and 1 odd pile, or 2 odd piles and 1 even pile. So, we can always choose to combine either 2

even piles or 2 odd piles. The combination of 2 even piles is even since $2k + 2l = 2(k + l)$ for $k, l \in \mathbb{Z}$, and the combination of 2 odd piles is even since $(2k + 1) + (2l + 1) = 2(k + l + 1)$ for $k, l \in \mathbb{Z}$. So, we can always make an even pile.

b. Let $n \geq 3$ be an odd integer. Suppose you have n red books and $n - 2$ blue books which you make into a single stack. Show that, no matter how you stack the books, you will always be able to find some book which is in between two red books. [6 pts]

Solution: For the sake of contradiction, suppose there is no book in between two red books. This means there are no red books exactly one position apart from each other. Now, consider splitting your stack into a stack of the even positioned books and a stack of the odd positioned books. Each stack has $n - 1$ books (an even number). Since no red books were exactly one position apart in the original stack, no red books are adjacent in either new stack. Thus, each new stack has at most $\frac{n-1}{2}$ red books, meaning we have a total of $n - 1$ red books. Contradiction! We have n red books. So, there is a book between two red books.

c. Recall that for $n \in \mathbb{N}$, we define $n! = n \cdot (n-1) \cdots 2 \cdot 1$. Prove that $(n+1)! = 1 + \sum_{k=1}^{n}(k! \cdot k)$ for all $n \geq 1$. [6 pts]

Solution: We proceed by induction on n.

Base Case: $(n = 1)$. $1 + \sum_{k=1}^{1} k! \cdot k = 2 = 2!$.

Induction Hypothesis: Suppose that $1 + \sum_{k=1}^{n} k! \cdot k = (n+1)!$ for some $n \geq 1$.

Inductive Step: Consider $n + 1$.

$$\begin{aligned} 1 + \sum_{k=1}^{n+1} k! \cdot k &= 1 + \sum_{k=1}^{n} k! \cdot k + (n+1)! \cdot (n+1) \\ &= (n+1)! + (n+1)(n+1)! \\ &= (n+2)(n+1)! \\ &= (n+2)! \end{aligned}$$

3 Stable Matching Constructions (16 Points)

Suppose we have a set of n jobs $\{J_1, \ldots, J_n\}$ and a set of n candidates $\{C_1, \ldots, C_n\}$.

In each part, you must describe preference lists for the jobs and candidates that meet the given conditions, or state that none exist. In either case, **you must prove** that your answer is correct.

(Hint: Only describe as much of each preference list as is necessary. For instance, if you were asked to give preference lists such that all jobs propose to C_1 on the first day of the job-propose algorithm, it would be enough to say "All jobs should have C_1 as their first choice, and the rest job's lists and all of the candidate's lists are arbitrary.")

a. Are there preference lists such that there is no stable matching? [3 pts]

Solution: There are no such preference lists. We can always run the job-propose algorithm which will give us a stable matching for any preference lists.

b. Suppose you are given two specific matchings T_1 and T_2. Are there preference lists such that they are both stable? [5 pts]

Solution: For each job J, pick its preference list so that its match in T_1 is its first choice, and the rest of each list is arbitrary. Likewise, for each candidate C, pick its preference list so that its match in T_2 is its first choice, and the rest of each list is arbitrary. We claim T_1 has no rogue couples. To see this, notice that if (J, C) is a rogue couple, J prefers C over its match in T_1. However, J's match in T_1 is its first choice by construction, so this is a contradiction. Similarly, T_2 is stable.

Note: Alternatively, you could prove that T_1 is stable by noting it will be the result of the job-propose algorithm, and prove that T_2 is stable by noting it will be the result of the candidate-propose algorithm.

c. Are there preference lists such that the job-propose algorithm halts in exactly n days, the resulting matching is $\{(J_1, C_1), \ldots, (J_n, C_n)\}$, and the resulting matching is the only stable matching?

(Hint: Once you have designed your preference list, try to show the first two properties together. Can you come up with a stronger statement about what proposals look like on each day? Try to prove it by induction.) [8 pts]

Solution: We design the job's preferences so that each job prefers C_1 the most, then C_2, and so on, with all jobs preferring C_n the least. We design the candidate's preference lists so that C_1 prefers J_1 the most, C_2 prefers J_2 the most, and so on.

To prove the first two properties, we claim that on day i, J_1 through J_i propose to C_1 through C_i, respectively, and $J_{i+1}, \ldots, J_n$ propose to C_i. We proceed by induction on the day.

Base Case: $(i = 1)$. On the first day, all jobs propose to candidate C_1, as claimed.

Induction Hypothesis: Suppose the claim holds on day i.

Inductive Step: Consider day $i + 1$. By our inductive hypothesis, we know on the previous day J_1 through J_i proposed to C_1 through C_i, respectively, and $J_{i+1}, \ldots, J_n$ proposed to C_i. Therefore, none of J_1 through J_{i-1} were rejected, so they will continue their previous proposals. Meanwhile, C_i most prefers J_i, so J_i will propose to C_i, and the remaining jobs will be rejected and propose to their next most favored candidate. Thus, jobs J_{i+1} through J_n will propose to C_{i+1}, as claimed.
Using our claim, it follows that the job-propose algorithm will halt after exactly n days, and will end with the matching $\{(J_1, C_1), \ldots, (J_n, C_n)\}$.

Finally, notice that the candidate-propose algorithm will halt in exactly 1 day with the same pairing since each candidate has a distinct first choice. Since the job-propose algorithm is job-optimal and the candidate-propose algorithm is job-pessimal and yet they give the same matching, we know this is the only stable matching.

4 Inductive Fibbing (10 Points)

The Fibonacci sequence is a recursive sequence given by $F_0 = 0, F_1 = 1$, and then $F_n = F_{n-1} + F_{n-2}$ for all $n > 1$. So, for instance, $F_2 = F_1 + F_0 = 1 + 0 = 1$ and $F_3 = F_2 + F_1 = 1 + 1 = 2$, and so on.

a. Prove that for any integer $n > 0$, there is an integer $l \geq 0$ such that $\frac{n}{2} < F_l \leq n$. [5 pts]

Solution: For the sake of contradiction, suppose there is no such l. Let m be the largest integer so that $F_m \leq n$. $n > 0$, so $m \geq 2$ since $F_2 = 1$. By assumption, $F_m \leq \frac{n}{2}$. But then, since $F_{m-1} \leq F_m$,

$$F_{m+1} = F_{m-1} + F_m \leq \frac{n}{2} + \frac{n}{2} = n$$

which contradicts the fact that m is the largest integer with $F_m \leq n$. So, there is $l \geq 0$ such that $\frac{n}{2} < F_l \leq n$.

b. Prove that every integer $n \geq 0$ can be written as the sum of distinct Fibonacci numbers (For example, $5 = F_1 + F_2 + F_4 = 1 + 1 + 3$) (Hint: Use part (a)). [5 pts]

Solution: We proceed by strong induction.

Base Case: ($n = 0$). If $n = 0$, we have $n = F_0$.

Induction Hypothesis: Suppose k can be written as the sum of distinct Fibonacci numbers for all $0 \leq k \leq n$, with $n \geq 0$.

Inductive Step: Consider $n + 1$. By part (a), there is an integer l so that $\frac{n+1}{2} < F_l \leq n + 1$. Consider $n + 1 - F_l$. $1 \leq F_l \leq n + 1$ so $0 \leq n + 1 - F_l \leq n$, and thus by the induction hypothesis $n + 1 - F_l$ can be written as a sum of distinct Fibonacci numbers, say $F_{l_1}, \ldots, F_{l_m}$. Since $n + 1 - F_l < \frac{n+1}{2} < F_l$, none of these Fibonacci numbers are F_l, so $n + 1$ can be written as the sum of distinct Fibonacci numbers $F_l, F_{l_1}, \ldots, F_{l_m}$.