# Supplementary Exercises- Mathematical Induction

## Summations

- Prove by induction on positive $n$ that $\displaystyle\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$.
- Prove by induction that for all $n \geq 1$, $\displaystyle\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$.
- Using a proof by induction, show that $\displaystyle\sum_{i=1}^{n} (i)(i!) = (n+1)! - 1$, for all $n \geq 1$.
- Prove that $\displaystyle\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ for all $n \geq 0$ by induction.

## Divisibility

- Show that $n^3 + 2n$ is divisible by 3 for all $n \geq 1$ by induction.
- Show that $n^4 - 4n^2$ is divisible by 3 for all $n \geq 2$ by induction.
- Show that $7^n - 1$ is divisible by 6 for all $n \geq 1$ by induction.
- Let $a, b, m$ be positive integers such that $m \mid (a - b)$. Prove by induction on $k \geq 0$ that $m \mid (a^k - b^k)$.

## Inequalities

- Prove that $\displaystyle\sum_{i=1}^{n} \frac{1}{\sqrt{i}} \geq \sqrt{n}$ for all $n \geq 1$ by induction.
- Prove by induction that for $n \in \mathbb{Z}^+$, $\displaystyle\sum_{i=1}^{n} \frac{1}{i^2} \leq 2 - \frac{1}{n}$.
- Using induction, prove that $3^n < n!$ for all $n \geq 7$.

## Sequences and Strong Induction

- Let $F_0 = F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$. Prove for all $n \geq 0$:
  - $\displaystyle\sum_{i=0}^{n} F_i = F_{n+2} - 1$.
  - $\displaystyle\sum_{i=0}^{n} (F_i)^2 = F_n \cdot F_{n+1}$.
  - $\displaystyle\sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n-i}{i} = F_n$.
  - $\gcd(F_{n+1}, F_n) = 1$.

- Show that postage of 24 cents or more can be achieved by using only 5-cent and 7-cent stamps.

- One way of generating the $n^{\text{th}}$ Fibonacci number is by using a recursive algorithm like the following.

```
int fibonacci(int n)    // generates Fn
  if ((n = 0) OR (n = 1)) then
    return 1;
  else
    return fibonacci(n-1) + fibonacci(n-2);
  end if
end fibonacci;
```

  Effectively, the algorithm mimics the definition of the Fibonacci sequence shown above.

  In computing science, the efficiency of an algorithm is measured by the amount of computing time expended as a function of the input. In this example, the input is the integer $n$, so we will let $T_n$ represent the amount of time expended to run `fibonacci(n)`.

  Note that this algorithm is *recursive* which means that it calls an instance of itself to help solve the problem. Function calls are very costly time-wise, so we can measure $T_n$ by counting the total number of calls to `fibonacci` during execution.

  For example, an initial call to `fibonacci(3)` will call `fibonacci(2)` and `fibonacci(1)`. `fibonacci(2)` will (in turn) call `fibonacci(1)` and `fibonacci(0)`. There are a total of 5 calls to `fibonacci`, which means $T_3 = 5$. (Note also that $T_2 = 3$.)

  - Determine the values of $T_0$ and $T_1$.
  - Write a recursive definition for $T_n$, for $n \geq 2$, $n \in \mathbb{N}$.
  - Prove by induction that for all $n \in \mathbb{N}$, $T_n = 2F_n - 1$.