**American University of Beirut**
**Department of Computer Science**
**CMPS 211 - Spring 2015-16**
**Assignment 4 - Solution Key**

**The following exercises are to be completed by Wednesday 23rd of March 2016. Late submissions will not be accepted.**
**You are expected to submit to moodle and another printed copy during class.**
**Only typed submissions will be accepted from now on, and those who submit in LaTeX will get extra credits.**

**Associated Readings: Sections 3.1, 5.1 and 5.2 of your text book**

# Exercise 1 (10 Points)

Can you guess the next number?

$$\frac{1}{3}, \frac{1}{15}, \frac{1}{35}, \frac{1}{63}, \ldots$$

a) Find a formula for the nth element in the list (Hint: Check the divisors of the denominator).

*Solution:*

Notice that the denominator of the first element is equal to $3 = 3 \times 1$.
The denominator of the second element is $15 = 5 \times 3$.
The denominator of the second element is $35 = 7 \times 5$.
The denominator of the second element is $63 = 9 \times 7$.
So it seems like the nth denominator is the product of the nth and (n+1)th odd number. In other words

$$a_n = \frac{1}{(2n-1)(2n+1)} \qquad n \geq 1$$

b) Find a formula for the sum of the first n elements:

$$S_n = \sum_{i=1}^{n} a_i = a_1 + a_2 + a_3 + \ldots + a_n$$

*Solution:*

Using the formula for the nth element from part (a) we have:

$$S_n = \sum_{i=1}^{n} a_i = \sum_{i=1}^{n} \frac{1}{(2i-1)(2i+1)}$$

Using few values for $n$ we get:

$$S_1 = \frac{1}{3}$$
$$S_2 = \frac{1}{3} + \frac{1}{15} = \frac{2}{5}$$
$$S_3 = \frac{1}{3} + \frac{1}{15} + \frac{1}{35} = \frac{3}{7}$$
$$S_4 = \frac{1}{3} + \frac{1}{15} + \frac{1}{35} + \frac{1}{63} = \frac{4}{9}$$

It seems as if the partial sum $S_n$ has numerator $n$, and its denominator is the (n+1)th odd number. We can conjuncture that:

$$S_n = \frac{n}{2n+1}$$

c) Prove the formula you conjectured in part (b) using Induction.

*Solution:*

We want to prove (By induction) that

$$\sum_{i=1}^{n} \frac{1}{(2i-1)(2i+1)} = \frac{n}{2n+1}$$

Base case ($n = 1$):

$$\sum_{i=1}^{n} \frac{1}{(2i-1)(2i+1)} = \frac{n}{2n+1} = \frac{1}{3}$$

Inductive Step: Assume the result for $n$

$$\sum_{i=1}^{n} \frac{1}{(2i-1)(2i+1)} = \frac{n}{2n+1}$$

We want to show that it holds for $n+1$.

$$\begin{aligned}
\sum_{i=1}^{n+1} \frac{1}{(2i-1)(2i+1)} &= (\sum_{i=1}^{n} \frac{1}{(2i-1)(2i+1)}) + \frac{1}{(2(n+1)-1)(2(n+1)+1)} \\
&= \frac{n}{2n+1} + \frac{1}{(2n+2-1)(2n+2+1)} \textit{ by inductive hypothesis} \\
&= \frac{n}{2n+1} + \frac{1}{(2n+1)(2n+3)} \\
&= \frac{n(2n+3)+1}{(2n+1)(2n+3)} \\
&= \frac{2n^2+3n+1}{(2n+1)(2n+3)} \\
&= \frac{(2n+1)(n+1)}{(2n+1)(2n+3)} \\
&= \frac{(n+1)}{(2n+3)} \\
&= S_{n+1}
\end{aligned}$$

**which completes the proof**

*Note: You could have broken up the fractions more, and then recollapsed it and reached same result.*

# Exercise 2 (10 Points)

a) Show that 4 divides $n^4 - n^2$ using mathematical induction.

*Solution:*

Base Case ($n = 0$): $n^4 - n^2 = 0$ divides 4.

Inductive Step: Assume $n^4 - n^2$ divides 4, show that $(n+1)^4 - (n+1)^2$ divides 4.

$(n+1)^2 = n^2 + 2n + 1$
$(n+1)^4 = (n+1)^2(n+1)^2 = (n^2+2n+1)(n^2+2n+1) = n^4 + 4n^3 + 6n^2 + 4n + 1$
$(n+1)^4 - (n+1)^2 = n^4 + 4n^3 + 6n^2 + 4n + 1 - (n^2+2n+1) = n^4 - n^2 + 4n^3 + 6n^2 + 2n$

We know that $n^4 - n^2$ *(by inductive hypothesis)*, and $4n^3$ are both divisible by 4, so it remains to show that $6n^2 + 2n = 2n(3n+1)$ is divisible by 4.

Now we complete the proof by considering the following cases:

(i) $n = 4k$, clearly $2n(3n+1)$ is divisible by 4 then.

(ii) $n = 4k+1$, then $2n(3n+1) = 2 \cdot (4k+1)(3(4k+1)+1)$
$= 2 \cdot (4k+1)(12k+4)$
$= 2 \cdot 4 \cdot (4k+1)(3k+1)$ which is divisible by 4.

(iii) $n = 4k+2$, then $2n(3n+1) = 2 \cdot (4k+2)(3(4k+2)+1)$
$= 2 \cdot 2 \cdot (2k+1) \cdot (3(4k+2)+1)$
$= 4 \cdot (2k+1)(3(4k+2)+1)$, which is divisible by 4.

(iv) $n = 4k+3$, then $2n(3n+1) = 2 \cdot (4k+3)(3(4k+3)+1)$
$= (8k+6)(12k+10)$
$= 2 \cdot 2(4k+3)(6k+5)$
$= 4(4k+3)(6k+5)$, which is divisible by 4.

**Therefore, $n^4 - n^2$ divides 4.**

b) Show that 3 divides $n^4 - n^2$ using mathematical induction.

Base Case ($n = 1$): $n^4 - n^2 = 0$ divides 3.

Inductive Step: Assume $n^4 - n^2$ divides 3, show that $(n+1)^4 - (n+1)^2$ divides 4.

$(n+1)^2 = n^2 + 2n + 1$
$(n+1)^4 = (n+1)^2(n+1)^2 = (n^2+2n+1)(n^2+2n+1) = n^4 + 4n^3 + 6n^2 + 4n + 1$
$(n+1)^4 - (n+1)^2 = n^4 + 4n^3 + 6n^2 + 4n + 1 - (n^2 + 2n + 1) = n^4 - n^2 + 4n^3 + 6n^2 + 2n$

We know that $n^4 - n^2$ *(by inductive hypothesis)*, and $6n^2$ are both divisible by 3, so it remains to show that $4n^3 + 2n = 2n(2n^2+1)$ is divisible by 3.

Now we complete the proof by considering the following cases:

(i) $n = 3k$, clearly $2n(2n^2+1)$ is divisible by 3 then.

(ii) $n = 3k+1$, therefore $2n(2n^2+1) = 2(3k+1)(2(3k+1)^2+1)$
$= (6k+2)(18k^2+12k+3)$
$= 3(6k+2)(6k^2+2k+1)$, which is divisible by 3.

(iii) $n = 3k+2$, then $2n(2n^2+1) = 2(3k+2)(2(3k+2)^2+1)$
$= (6k+4)(18k^2+24k+9)$
$= 3(6k+4)(6k^2+8k+3)$, which is divisible by 3.

**Therefore, $n^4 - n^2$ divides 3.**

c) Conclude that 12 divides $n^4 - n^2$.

*Solution:*

Since $n^4 - n^2$ divides 4 and 3, it can be written as $4m$ and $3k$ for naturals $m, k$.
Since 4 and 3 are **relatively prime** then $4m = 3k = 4.3z = 12z$ for some natural $z$.
Since $n^4 - n^2 = 12z$ then clearly $n^4 - n^2$ divides 12.

# Exercise 3 **(10 Points)**

Consider the proposition $2^n > n^3$.

a) Find an integer N such that the proposition is true whenever n is greater than N.

*Solution:*

We need to look for n such that the proposition is true, by trying different values of $n$, we can find that when $n = 10$:

$$2^{10} = 1024 > 1000 = 10^3$$

Therefore $N = 10 - 1 = 9$

b) Prove your result for all $n > N$ using mathematical induction.

*Solution:*

Base case ($n = N + 1 = 10$) Already shown.
Inductive Step: Assume $2^n > n^3$ for $n > 9$, prove it for $n + 1$.

$(n+1)^3 = (n+1)(n^2 + 2n + 1) = n^3 + 3n^2 + 3n + 1$

$$\begin{aligned}
2^{n+1} &= 2^n \times 2 \\
&> 2 \times n^3 \text{ (Using Inductive Hypothesis)} \\
&= n^3 + n^3 > n^3 + 9n^2 \text{ (since n ¿ 9)} \\
&= n^3 + 3n^2 + 6n^2 \\
&> n^3 + 3n^2 + 6 \cdot 9n \text{ (since n ¿ 9)} \\
&= n^3 + 3n^2 + 54n \\
&= n^3 + 3n^2 + 3n + 51n \\
&> n^3 + 3n^2 + 3n + 1 \text{ (since 51n ¿ 1 for n ¿ 9)} \\
&= (n+1)^3
\end{aligned}$$

Therefore $2^{n+1} > (n+1)^3$ for all $n > N = 9$.

# Exercise 4 (10 Points)

Assume you can only use 7-cent and 11-cent stamps.

a) Determine which amounts of postage can be formed by the given stamps.

*Solution:*

We can only form amounts of the form $n = 7a + 11b$ for $a, b \geq 0$. So we can form $0, 7, 11, 14, 18, 21, 22, 25, 28, 29, 32, 33, 35, 36, 39, 40, 42, 43, 44, 46, 47, 49, 50, 51, 53, 54, 55, 56, 57, 58,$ **60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 71, ...**

We can form any amount greater than or equal to 60.

b) Prove your answer to (a) using the principle of mathematical induction. Be sure to state explicitly your inductive hypothesis in the inductive step.

*Solution:*

Base case: We can form 60 as shown in part a.

Inductive step:

We assume that we can form some amount $n \geq 60$, we want to show that we can form amount $n + 1$.
Since we can form amount $n$ then there exist some naturals $a, b$ such that $n = 7a + 11b$.

We have $n + 1 = 7a + 11b + 1$ we want to show that $n + 1 = 7a' + 11b'$ for some $a', b'$ naturals.

We want to find combinations of $a$ and $b$ such that $|7a - 11b| = 1$, Notice that $7 * 3 = 21$ and $11 * 2 = 22$, therefore $7(a - 3) + 11(b + 2) = 7a - 7 * 3 + 11b + 22 = 7a + 11b + 1 = n + 1$.

So if $n = 7a + 11b$ where $a \geq 3$ we can use the derivation above to form $n + 1$.
However it is possible that $a < 3$.

Notice that $7 * 8 = 56$, while $11 * 5 = 55$. Therefore, if $a < 3$ we can try $7(a + 8) + 11(b - 5) = 7a + 56 + 11b - 55 = n + 1$.

So if $n = 7a + 11b$ and either $a \geq 3$ or $b \geq 5$ we can form $n + 1$.

Take the remaining case where $a < 3$ and $b < 5$, the biggest amount we can form from such a combination is $7 * 2 + 11 * 4 = 14 + 44 = 58$, However we assumed that $n \geq 60$. Therefore it is never the case that $a < 3$ and $b < 5$, and we can always form $n + 1$ for $n \geq 60$.

c) Prove your answer to (a) using strong induction. How does the inductive hypothesis in this proof differ from that in the inductive hypothesis for a proof using mathematical induction?

*Solution:*

Base case: We can form amounts $60, 61, 62, 63, 64, 65, 66$ as shown in part a.

Inductive Step:

We assume that we can form amounts from 60 to $n - 1$ inclusive, in other words $\forall k : 60 \geq k \leq n - 1, \exists a, b \in \mathbb{N} : k = 7a + 11b$.

We need to show that we can form $n + 1$.
Take $n + 1 - 7$ since $n + 1 - 7 < n$ then by the inductive hypothesis we have

$$n + 1 - 7 = 7a + 11b \quad \text{for some } a, b \in \mathbb{N}$$

.

Consider $a' = a + 1$, we have

$$7a' + 11b = 7a + 7 + 11b = (n + 1) - 7 + 7 = n + 1$$

Therefore we can form $n + 1$.

# Exercise 5 **(30 Points)**

Given a list of $n$ distinct integers, we wish to design and analyse an algorithm for each of the following:

1. To produce the product of the two elements with value closest to the average of the list, such that one of them is greater than the average and the other is lower than the average.

2. To produces all pairs in the list with sum equal to a given value $k$.
This methods take the list, and an integer k as input arguments. So for example, if the sum of the pair $a_1$ and $a_4$ is equal $k$, then the algorithm should print $a_1, a_4$, and all other pairs with the same property

3. To locate the position of the element with largest difference with the element before it.

For each algorithm above, do the following:

a) Give a complete design using pseudo-code.

b) Calculate the number of operations required by each algorithm.

c) Calculate the amount of space required by each algorithm.

d) State a loop invariant (don't prove it).

## First Algorithm

**Procedure avg_prod(integer $c_1$, ..., integer $c_n$)**

```
avg := 0
for i := 1 to n
    avg := avg + c_i
avg := avg / n

a := 0
b := +∞
for i := 1 to n
    if c_i < avg and c_i > a then
        a := c_i
    else if c_i > avg and c_i < b then
        b := c_i
return a × b
```

*Number of operations*

1. the first loop (to calculate average) requires 1 assignment and 1 addition in each iteration. It has n iterations: so the number of operations associated with this step is:
$$\sum_{i=1}^{n} c = cn \to n \qquad \text{as } n \to \infty$$
2. the second loop has $n$ iterations, in each iteration we can have at most 4 comparisons and 1 assignment. The number of operations associated with this step is:
$$\sum_{i=1}^{n} c' = c'n \to n \qquad \text{as } n \to \infty$$
3. we have 1 division, 1 multiplication, and 3 assignment outside of both loops.

In total, the maximum number of operations tends to $n$ as $n \to is\ \infty$.

*Space:* The algorithm uses $n$ slots of memory for the input and a constant amount of space for intermediary and final output. In total, the space is dominated by $n$ as $n \to \infty$.
*Invariant:* $P(i)$ = "After the $i$'th iteration, $a$ is the closest value to the average that is smaller than it inside $c[0...i]$ and $b$ is the closest value to the average that is greater than it inside $c[0...i]$.

## Second Algorithm

**Procedure kpair(integer $c_1$, ..., integer $c_n$, integer $k$)**
  **for** i := 1 **to** n
    count = 0
    **for** j := i+1 **to** n
      **if** $c_i + c_j$ **equals** $k$
        count = count + 1
        $A[count] = (c_i, c_j)$

*Number of operations:*
Each iteration from the inner loop contains a comparison and a write in the worst case. For every iteration of the outer loop with value i, the inner loop has $n - i$ iterations. The outer loop has $n$ iterations.

$$\begin{array}{rcl} T(n) & = & \sum_{i=1}^{n} \sum_{j=i+1}^{n} c \\ & = & c\left(\sum_{i=1}^{n} \sum_{j=i+1}^{n} 1\right) \\ & = & c\left(\sum_{i=1}^{n} (n-i)\right) \\ & = & c\left(\sum_{i=1}^{n} n - \sum_{i=1}^{n} i\right) \\ & = & c\left(n^2 - \frac{n(n+1)}{2}\right) \\ & \to & n^2 \text{ as } n \to \infty \end{array} \quad rcl$$

*Space:* The algorithm requires about $n$ records for input and at most $n$ records for output and so the space complexity is dominated by $n$ as $n$ goes to $\infty$.

*Invariant:* $P(i)$ = "After the $i$'th iteration, *count* represents the number of all pairs $(c_p, c_q)$ such that $c_p + c_q = k$ and $p \leq i, q \leq j$ and all elements in $A[1, \ldots, count]$ record those pairs.

## Third Algorithm

**Procedure largestDiff(integer $c_1$, ..., integer $c_n$)**
  max := -$\infty$
  i_max := 0
  **for** i := 2 **to** n
    **if** $c_i - c_{i-1} > max$
      **max** := $|c_i - c_{i-1}|$
      **i_max** := i
  return **i_max**

*Number of operations:*
Each iteration incurs at most 2 subtractions, 2 assignments, and 1 comparison.
There are $n - 1$ iterations in total. So $T(n) \to n$ as $n \to \infty$.
*Space:* The algorithm requires $n$ records for storing the input and three variables for tracing the output (constant amount of memory independent of $n$). Hence, the space complexity is domianted by $n$ as $n \to \infty$.

*Invariant:* $P(i)$ = "After the $i$'th iteration, **max** = $\max\{|c_i - c_{i-1}|\}$ and **i_max** is such that $c_{i_{max}} - c_{i_{max-1}} = \max\{|c_i - c_{i-1}|\}$.

# Exercise 6 (30 Points)

Give an estimate as $n \to \infty$ for the run-times of the following segments of algorithms.

**Procedure alg1(integer $n$, integer $m$)** (3 points)
$s := 0$
**while** $n > 0$
    $s := s + m$
**return** $s$

*Solution:*

The loop does not terminate, the terminating condition for the loop is when $n \leq 0$, However $n$ is not changed during or before the loop. We have two cases: Either $n$ is given as a non positive number (in which case no loops will be excuted and run time is constant), or $n$ is given positive, in which case the loop goes on forever, and the runtime is $+\infty$.

**Procedure alg2(integer $n$)** (3 points)
$t := 0$
**for** $i := 1$ **to** $n$
    **for** $j := i + 1$ **to** $n$
        **for** $k := j + 1$ **to** $n$
            $t := i + j + k$

*Solution:*

Outermost loop has n iterations, for each of which the loop within it has n - i iterations, for each of which the inner most loop has n - j iterations.

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n} c \\
&= c\sum_{i=1}^{n}\sum_{j=i+1}^{n}(n-j) \\
&= c\left(\sum_{i=1}^{n}\sum_{j=i+1}^{n} n - \sum_{i=1}^{n}\sum_{j=i+1}^{n} j\right) \\
&= c\left(\sum_{i=1}^{n} n\sum_{j=i+1}^{n} 1 - \sum_{i=1}^{n}\sum_{j=i+1}^{n} j\right) \\
&= c\left(\sum_{i=1}^{n} n(n-i) - \sum_{i=1}^{n}(\frac{n(n+1)}{2} - \frac{i(i+1)}{2})\right) \\
&= c\left(n^3 - \frac{n^2(n+1)}{2} - \frac{n^2(n+1)}{2} + \sum_{i=1}^{n}\frac{i(i+1)}{2}\right) \\
&= c\left(-n^2 + \frac{1}{2}\left(\sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} i\right)\right) \\
&= c\left(-n^2 + \frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right)\right)
\end{aligned}
$$

This goes to $n^3$ as n goes to $\infty$

**Procedure alg3(integer $n$)** (4 points)

$i := 1$
$t := 0$
**while** $i \leq n$
    $t := t + i$
    $i := 10 * i$

*Solution:*

The while loop begins with $i = 1$, at every iteration $i$ gets multiplied by 10 and $n$ remains unchanged. $i$ takes the values of 1, 10, 100, 1000, ... in other words, for $n < 10$ the loop needs one iteration to terminate, for $10 < n < 100$ it needs 2, for $100 < n < 1000$ it needs 3, for $10^{k-1} < n < 10^k$ it needs $k$ iterations.
Taking the log of both sides, we realize that for $n$ the loop will terminate within $\log_{10}(n)$ iterations. The runtime tends to $\log_{10}(n)$ as $n \to \infty$.

**Procedure alg4(integer $n$)** (5 points)

$k := 1$
**for** $i := 1$ **to** $n$
    $t := i$
    **while** $t > 0$
        $k := k + t$
        $t := t/2$

*Solution:*

Outermost loop has n iterations, The inner loop depends on $t = i$.
For $t = 2$ the loop is executed twice, for $t = 4$ the loop is executed 3 times, for $t = 8$ the loop is executed 4 times, for $t = 2^k$ the loop is executed $k$, therefore the inner loop executes $\log_2(i) + 1$ times for each $i$.

$$\sum_{i=1}^{n} \sum_{t=1}^{\log_2(i)+1} c < \sum_{i=1}^{n} \sum_{t=1}^{\log_2(n)+1} c = n(\log_2(n) + 1)$$

This tends to $n \log(n)$ as $n \to \infty$.

# Exercise 7 (15 Points)

For each of the following algorithms, state a loop invariant and then **prove it**.

**Algorithm 1**: This procedure compute $x^n$

**Procedure power(integer** $x$**, integer** $n$**)**
*power* := 1
**for** $i := 1$ **to** $n$
*power* := *power* $\times$ $x$

**return** *power*;

*Solution:*

**The loop invariant:** $P(i)$ = "At the start of the $i$'th iteration we have power = $x^{i-1}$".

**Initialization:** Show $P(1)$: at the start of the first iteration, we have power = 1 which is equal to $x^{i-1} = x^0$.

**Maintenance:** Assume that at the start of iteration $i$ we have power = $x^{i-1}$.
During the $i$'th iteration, the algorithm updates power by a multiplication with $x$ and so we now have power = $x^i$ by the end of the $i$'th iteration (which is by the beginning of the $i + 1$'st iteration. We thus maintain $P(i + 1)$.

**Termination:** Since the for loop varies over the range [1, $n$].
Upon Termination we have $i = n + 1$ (Termination Condition), and power = $x^{i-1}$ (Invariant).
At this stage of the code, we have

$$i = n + 1 \wedge \text{power} = x^{i-1} \to \text{power} = x^{n+1-1} \to \text{power} = x^n$$

**Algorithm 2**: This procedure compute $|n|$

**Procedure abs(integer** $n$**)**
int $absolute := 0$
**if** $n < 0$
$absolute := -n$
**else**
$absolute := n$

**return** *absolute*

*Solution:*

The invariants and assertions are stated as in-line comments.

*Idea: There is one and only one iteration. All we have to do is state the "meaning" of every instruction mathematically. More or less, this is an example of a loop invariant that only has* $P(1)$ *instance.*

**Procedure abs(integer** $n$**)**
int $absolute := 0$
**if** $n < 0$
{{ n is negative }}
$absolute := -n$
{{ n is negative and absolute $= -n$ }}
{{ absolute $= |n|$ }}
**else**
{{ n is non-negative }}
$absolute := n$
{{ n is non-negative and absolute $= n$ }}
{{ absolute $= |n|$ }}

{{ absolute $= |n|$ }}
**return** *absolute*

The proof is a straightforward justification of the way the absolute value function is defined.

**Algorithm 3**: This procedure computes quotients and remainders for the division of $p$ by $d$.

**Procedure division(integer $p$, integer $d$)**
$q := 0$ {$q$ *is the quotient*}
$r := p$ {$r$ *is the remainder*}
**while** $r \geq d$
$r := r - d$
$q := q + 1$

**return** $(q, r)$

*Solution:*

**The loop invariant:** $P(i)$ = "At the beginning of the $i$'th iteration $p = r_i + q_i * d$", where $r_i$ and $q_i$ designate the values of $r$ and $q$ respectively, at the beginning of the $i$'th iteration.

**Initialization:** before starting the loop, we have $q_1 = 0$ and $r_1 = p$. Therefore $p = r_1 + q_1 * d$.

**Maintenance:** Assume $P(i)$ = "At the beginning of the $i$'th iteration $i$, we have $p = r_i + q_i * d$". In the duration of the $i$'th iteration, we are updating the estimates on $r$ and $q$ as follows:
$r_{e+1} = r_i - d \rightarrow r_i = r_{i+1} + d$
$q_{i+1} = q_i + 1 \rightarrow q_{i+1} = q_i - 1$

By substituting the values in the induction hypothesis, we get:

$$p = (r_{i+1} + d) + (q_{i+1} - 1) * d$$

$$p = r_{i+1} + d + q_{i+1} * d - d$$

$$p = r_{i+1} + q_{i+1} * d$$

Therefore $p = r_{i+1} + q_{i+1} * d$ holds at the end of the $i$'th iteration (which is the beginning of the $i + 1$'st iteration – hence $P(i + 1)$ is established.

**Termination:** The loop terminates when $r < d$, since $d$ is a finite number and at every iteration $r$ is decremented by a constant amount, therefore $r$ will eventually become less than $d$ allowing termination. That is when $r$ is a valid remainder and $q$ is the final quotient.

# Extra Practice (Not To Be Submitted)

From the textbook "Discrete Mathematics and Its applications $7^{\text{th}}$ Edition"
Sec. 3.1 (exercises $1 \rightarrow 36$, $38 \rightarrow 43$)
Sec. 5.1 (exercises3 $\rightarrow 7, 9, 31 \rightarrow 35$)
Sec. 5.2 (exercises 3, 4, 8)