Name: Kevin Zhang

Problem 1.

(a) Given a function $f : X \to Y$ where X and Y are finite sets, and $|X| = |Y|$. Suppose that f is injective. This means for every element $y \in Y$, there is at most one arrow pointing in. But, f is a function, which means for every unique $x \in X$, there must be an arrow going from x to y. Since no y can have more than one arrow pointing in, and $|X| = |Y|$, then the number of arrows pointing in must be exactly 1 for every y. This implies that f is bijective, which then implies that f must also be surjective as well.

(b) Given a fucntion $f : X \to Y$ where X and Y are finite sets, and $|X| = |Y|$. Suppose that f is surjective. This means that for every element $y \in Y$, there is at least one arrow pointing in. But f is a function, which means there cannot be more than one arrow pointing out for every $x \in X$. Since $|X| = |Y|$, it is not possible for there to be any y with more than one arrow pointing in, which means there is exactly 1 arrow pointing in for every y. This implies that f is bijective, which then implies that f must also be injective as well.

Problem 2.

(a) A function that produces every n^{th} prime number. Some numbers are not prime, and will have no arrows pointing into them, making the function not surjective by default.

(b) A function that produces the smallest factor of n. Multiple numbers will have the same smallest factor (eg. 2), making this surjective, but not injective.

Problem 3.

3^6	3^5	3^4	3^3	3^2	3^1	3^0
b	c	a	c	b	a	b

In b-adic ordering, $a = 1$, $b = 2$, and $c = 3$, so the total would be

$$(3^6 \times 2) + (3^5 \times 3) + (3^4 \times 1) + (3^3 \times 3) + (3^2 \times 2) + (3^1 \times 1) + (3^0 \times 2) = \mathbf{2372}$$

Problem 4.

Number 909 can be broken down as follows:

$$\begin{aligned} 909 &= 3 \times 302 + \mathbf{3} \\ 302 &= 3 \times 100 + \mathbf{2} \\ 100 &= 3 \times 33 + \mathbf{1} \\ 33 &= 3 \times 10 + \mathbf{3} \\ 10 &= 3 \times 3 + \mathbf{1} \\ 3 &= 3 \times 0 + \mathbf{3} \end{aligned}$$

With $a = 1$, $b = 2$, and $c = 3$, the string would be **cbacac**.

Problem 5.

If we assign a bit vector to each of the results from f, we would get:

$$\begin{aligned} f(a) &= \{a,b\} && \rightarrow 11000 \\ f(b) &= \{c,d,e\} && \rightarrow 00111 \\ f(c) &= \{b,c\} && \rightarrow 01100 \\ f(d) &= \varnothing && \rightarrow 00000 \\ f(e) &= \{a,c,d\} && \rightarrow 10110 \end{aligned}$$

We can then take the flipped diagonal, and we get the bit-vector 01011, which corresponds to the set $\{b,d,e\}$.

Problem 6.

For the sets in the sequence $S_0, S_1, S_2, ...$ of subsets of $\mathbb{N}$, we are only concerned with the even numbers in each set (since S can contain only evens), so we can form the diagram as such:

	0	2	4	6	...
S_0	1	0	1	0	...
S_1	0	1	0	1	...
S_2	1	1	1	1	...
S_3	0	0	0	1	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

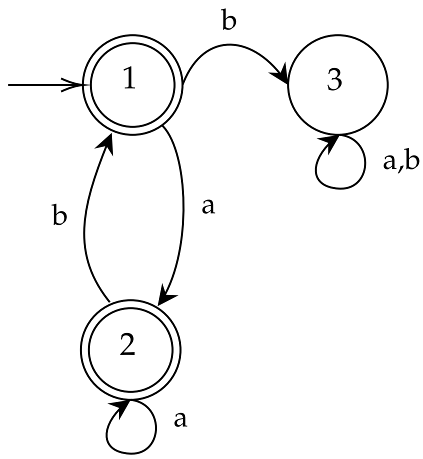
The bit-vectors for $S_0, S_1, S_2, ...$ are arbitrary here, simply for demonstration. Then, we can use diagonalization (by flipping the bits along the diagonal) to create a unique bit-vector for S, consisting of only even numbers, and where $S \neq S_0, S_1, S_2$.

Problem 7.

The tuple (Q, σ, δ, s, F) in order:
The states are $Q = q_1, q_2, q_3$
The alphabet is $\sigma = a, b$
The transition function (δ) is shown in the table below
The starting state is $s = 1$
The acceptable final states are $F = q_1, q_3$

δ	a	b
q_1	q_2	q_2
q_2	q_2	q_3
q_3	q_1	q_2

Problem 8.



Problem 9.

$L(M) = \{w \in \sigma * \mid \ w \ \text{ does not start with b and does not contain bb.}\}$

Problem 10.

(a)

δ	0	1
q_s	q_s	q_s

where $s = q_s$ and $F = \{\}$. q_s represents the starting state, and there is no final state in this case, because there is no word that belongs in $\emptyset$.

(b)

δ	0	1
q_s	q_e	q_e

where $s = q_s$ and $F = \{q_s\}$. q_s represents the starting state, and q_e represents a non-empty state. Since we are trying to get to ϵ, the only acceptable state is the starting state.

(c)

δ	0	1
q_s	q_0	q_1
q_0	q_n	q_f
q_1	q_f	q_n
q_n	q_n	q_n
q_f	q_n	q_n

where $s = q_s$ and $F = \{q_f\}$. q_0 and q_1 represents states in which the word starts with 0 or 1, respectively. From q_0, if a 1 is entered, then we enter q_f, which is the accepting state (similar is true for q_1). Any more letters after that (or if 0 is entered after q_0 or 1 after q_1), we enter q_n, which is simply a non-accepting state from which we can't exit.

(d)

δ	0	1
q_s	q_0	q_1
q_0	q_0	q_0
q_1	q_n	q_1
q_n	q_n	q_n

where $s = q_s$ and $F = \{q_1\}$. Because we have to start with 1, q_s and q_n are here for the situations of ϵ and when we start with 0, respectively. There is no exit from q_n. Then, for situations we do start with 1, we simply shuffle between q_1 and q_0, which represents what was the last digit read.

(e)

δ	0	1
q_s	q_0	q_1
q_0	q_n	q_1
q_1	q_0	q_n
q_n	q_n	q_n

where $s = q_s$ and $F = \{q_s, q_0, q_1\}$. Here, the only non-accepting state is q_n, which represents when 00 or 11 has been found. q_1 and q_0 represents the last digit read, and simply transitions between each other, except if there is a 00 or a 11.

(f)

δ	0	1
q_s	q_0	q_1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_2	q_1

where $s = q_s$ and $F = \{q_0\}$. The states q_0, q_1, q_2 represent the remainder when the binary string (so far) is divided by three. This logic works because each digit on a binary string is eventually added together, so whenever the remainder becomes 0, it can be treated as a new string. For when the remainder is 1, then we can examine 11 or 10 (for the next read). In the prior case, $11_2 = 3$, so the remainder is 0. In the latter case, $10_2 = 2$, so the remainder is 2. Finally, we can move into the cases when the remainder is 2. Remainder 2 in binary is 10, so we prepend that to the next digit and the cases become 101 or 100. The former gives remainder 2 (again), and the latter gives remainder 1.

(g)

δ	0	1
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_3
q_3	q_3	q_3

where $s = q_0$ and $F = \{q_3\}$. Since the strings x and y can be empty, then this simply becomes a search to see if the substring 111 is in this string. q_0, q_1, q_2, q_3 represents the number of consecutive 1's found, and once 3 are found, q_3 doubles as the accepting truthy state.

Problem 11.

The code is as follows:

```
# Accepts All binary strings in which all two successive ones
# are separated by an odd number of zeroes

start: truthy

accept-states: [truthy]

transitions:
- [truthy, 0, truthy]
- [truthy, 1, open_1]

- [open_1, 0, odd_zero]
- [open_1, 1, falsy]

- [odd_zero, 0, even_zero]
- [odd_zero, 1, truthy]

- [even_zero, 0, odd_zero]
- [even_zero, 1, falsy]

- [falsy, 0, falsy]
- [falsy, 1, falsy]
```