John Spicer
I pledge my honor that I have abided by the Stevens Honor System

4.1
a. Yes, 0100 is accepted by DFA M.
b. No, 011 is rejected by M.
c. No, the input is not in the correct form (only 1 comp)
d. No, the input is not in the correct form (first comp is not a regular exression)
e. No, the M's language is not empty.
f. Yes, M accepts the same language as itself.

4.2
Define the language as C = {<M, R> | M is a DFA and R is regular expression where L(M) = L(B) }

Turing Machine T decides C:

T = " on input <M,R> (M is DFA, R is reg. Expression):
1. Convert R into DFA D_R w | Kleene's theorem
2. Run TM decider F on input <M, D_R>
3. If F accepts, accept. If F rejects, reject.

4.3
$ALL_{DFA} = \{<A> \mid A$ is DFA and $L(A) = \Sigma^* \}$

Turing Machine M decides ALL_{DFA}:

M = "on input x
1. If x is DFA, denote the DFA_C D_A. Else reject x.
2. Make the complement DFA D_A
3. Run <$D_A{}^C$> on Turing Machine T that decides E_{DFA}
4. If T accepts <A^C>, accept x.
5. If T rejects, reject x.

4.7
Assume {0, 1} belongs to real number R and is countable.
Assumption must be contradicted to prove B is uncountable.

Countable real number:
$r_0, r_1, r_2, r_3, \ldots$

B lies in the range {0, 1}. This means that a number y between 0 and 1 has an integer part that is 0.

Suppose y_0 is a number < r_0. This ensures y_0 and r_0 are != .

$y_1 \leftrightarrow .d_{11}, d_{12} \ldots$

$y_2 \leftrightarrow .d_{21}, d_{22} \dots$
$y_3 \leftrightarrow .d_{31}, d_{32} \dots$
$y_n \leftrightarrow .d_{n1}, d_{n2} \dots$

Since $y = 0.\ y_0 y_1 y_2 y_3 \dots$ and y_n != d_{nn} for each N, then y_n != r_i .
This proves by contradiction that the value of y is different from the value of a rational number. Therefore, B is uncountable via diagonilization.

4.8

T = { (i, j, k) | i, j, k ∈ N }

Make a one-to-one and onto mapping from T to N.

i, j, k are natural numbers and are countable.

<i, j, k> = <i1, j1, k1> is a one-to-one function from S to N and an onto mapping from S to N.

Prove it is also a one-to-one and onto mapping from T to N.

Assume (h (<i, j>, k)) = (h (<i’, j’>, k’)).

Assume f (<i, j, k>) = f’ (<i’, j’, k’>).

h (<h (<h <i, j>, k>) >) = h (<h (<i’, j’>, k’)>)

Since h is one-to-one:

(h (<i, j>, k)) = (h (<i’, j’>, k’))

h (<i, j>) = h (<i’, j’>) and k = k’

<i, j> = <i’, j’>

h (<m, k>) = n, where m and k are natural numbers

h is onto, therefore f is also onto.

Therefore, f is one-to-one and onto. Hence, T is countable.

4.16

A = { <R> | R is a regular expression describing a language containing 1 string w that has 111 as a substring }

Assume regular expression: C = (0 ⋃ 1)* 111 (0 ⋃ 1)*

L(R) ∈ A, L(R) is regular

C ∩ L(R) has DFA M ∩ L(R), regular language is closed under intersection.

Following Turing Machine T decides A:

1. Convert R into DFA DR with Kleene's Theorem.
2. Make DFA DC∩L(R)
3. Run Turing Machine
4. Simulate TM that decides CFG, generate strings that contain 111 as a substring. If it accepts, accept. If it rejects, reject.

This proves language A is decidable.

4.21

S = { <M> | M is a DFA that accepts wR whenever it accepts w}

Consider Turing machine N.

Check if M is a proper DFA. If proper, accept. Otherwise, reject.

Make DFA MR that recognizes { w | wR }

Simulate Turing Machine N.

DFA MR is constructed by making NFA from M by reversing transitions as follows:

1. Swap initial state w/ accepting state

2. Make new initial state w/ ϵ transition to earlier accepting states

This proves M is decidable by S. Therefore, language S is decidable.

Turing Machine Code:

```
name: fibonacci
init: Init
accept: Accept

Init, _, _, _
Accept, _, _, _, -, -, -

Init, 0, _, _
Shift, 0, _, 0, -, -, -
```

Shift, 0, _, 0
Check, 0, _, 0, -, -, -

Shift, 0, 0, _
Shift, 0, 0, _, -, >, <

Shift, 0, 0, 0
Shift, 0, 0, 0, -, >, <

Shift, 0, _, _
Shift2, 0, _, _, -, <, -

Shift2, 0, 0, _
Shift2, 0, 0, 0, -, <, >

Shift2, 0, _, 0
Shift3, 0, _, 0, -, >, -

Shift3, 0, _, 0
Shift3, 0, 0, 0, -, >, >

Shift3, 0, 0, 0
Shift3, 0, 0, 0, -, >, >

Shift3, 0, _, _
Shift4, 0, _, _, -, <, <

Shift4, 0, 0, 0
Shift4, 0, 0, 0, -, <, <

Shift4, 0, _, 0

Shift4, 0, _, 0, -, -, <

Shift4, 0, 0, _
Shift4, 0, 0, _, -, <, -

Shift4, 0, _, _
Check, 0, _, _, -, >, >

Check, 0, 0, 0
Check, 0, 0, 0, >, >, -

Check, 0, _, 0
Check, 0, _, 0, >, -, >

Check, _, _, _
Accept, _, _, _, -, -, -

Check, 0, _, _
Reset, 0, _, _, -, <, <

Reset, 0, 0, 0
Reset, 0, 0, 0, -, <, <

Reset, 0, _, 0
Reset, 0, _, 0, -, -, <

Reset, 0, 0, _
Reset, 0, 0, _, -, <, -

Reset, 0, _, _
Reset, 0, _, _, <, -, -

Reset, _, _, _
Reset2, _, _, _, >, >, >

Reset2, 0, _, 0
Check, 0, 0, 0, -, -, -

Reset2, 0, 0, 0
Shift, 0, 0, 0, -, -, -