QUESTION 1. [10 MARKS]

Consider the (familiar, Ihope) recursive function:

$$\forall m \in \mathbb{N}, 0 \leq n \leq m \qquad PT(m,n) = \begin{cases} 1, & \text{if } n = 0 \\ 1, & \text{if } n = m \\ PT(m-1,n-1) + PT(m-1,n), & \text{if } 0 < n < m \end{cases}$$

Use induction to prove that for all $m \in \mathbb{N} - \{0\}$ (positive natural numbers):

$$\left(\sum_{n=0}^{m} (-1)^n PT(m,n)\right) = 0.$$

CLAIM: Let $P(m)$ be "$(\sum_{n=0}^{m}(-1)^n PT(m,n)) = 0$." Then $P(m)$ is true for all $m \in \mathbb{N} - \{0\}$.

PROOF (INDUCTION ON m): If $m = 1$, then $PT(1,0) = PT(1,1) = 1$, so $P(m)$ claims that $1 + (-1) = 0$, which is true, so the base case holds.

INDUCTION STEP: Suppose $P(m)$ holds for some arbitrary $m \in \mathbb{N} - \{0\}$, in other words I assume that $(\sum_{n=0}^{m}(-1)^n PT(m,n)) = 0$. I want to show that this implies $P(m+1)$. I can break up the summation and use the IH:

$$\begin{aligned}
\left(\sum_{n=0}^{m+1} (-1)^n PT(m+1,n)\right) &= 1 + \left(\sum_{n=1}^{m} (-1)^n PT(m+1,n)\right) + (-1)^{m+1} \\
[\text{defn of } PT(m+1,n)] &= (-1)^0 + \left(\sum_{n=1}^{m} (-1)^n [PT(m,n-1) + PT(m,n)]\right) + (-1)^{m+1} \\
[\text{defn } PT(m,0),\ PT(m,m)] &= \left((-1)\sum_{n-1=0}^{m} (-1)^{n-1} PT(m,n-1)\right) + \left(\sum_{n=0}^{m} (-1)^n PT(m,n)\right) \\
[\text{IH, twice}] &= -0 + 0 = 0.
\end{aligned}$$

Thus, $P(m) \Rightarrow P(m+1)$, as wanted.

I conclude that $P(m)$ is true for all $m \in \mathbb{N} - \{0\}$. QED.

QUESTION 2. [10 MARKS]

Let $PV = \{x_1, \ldots, x_{10}$ be a set of propositional variables. Define the set of propositional formulas over PV as F_{PV} by

BASIS: Every $x_i \in PV$ is in F_{PV}.

INDUCTION STEP: If f_1 and f_2 are in F_{PV}, then so are $(f_1 \wedge f_2)$, $(f_1 \vee f_2)$, $(f_1 \rightarrow f_2)$, $(f_1 \leftrightarrow f_2)$, and $\neg f_1$.

For $f, f_1, f_2 \in F_{PV}$, and $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ is a binary connector, define subformula(f) as the number of sub-formulas that occur as substrings in f, and define height(f) recursively

$$\text{height}(f) = \begin{cases} 1, & \text{if } f \in PV \\ 1 + \text{height}(f_1), & \text{if } f = \neg f_1 \\ 1 + \max\{\text{height}(f_1), \text{height}(f_2)\}, & \text{if } f = (f_1 * f_2) \end{cases}$$

Use structural induction to prove that for all $f \in F_{PV}$:

$$\text{height}(f) \leq \text{subformula}(f) \leq 2^{\text{height}(f)} - 1$$

Notice that, by definition, all formulas and subformulas are non-empty.

CLAIM: Let $P(f)$ be "height(f) $\leq$ subformula(f) $\leq 2^{\text{height}(f)} - 1$." Then $P(f)$ is true for all $f \in F_{PV}$.

PROOF (INDUCTION ON f): Suppose $f \in PV$. Then f has one subformula (itself), and has height 1 (by definition). This means that height(f) $= 1 =$ subformula(f) $= 2^1 - 1 = 2^{\text{height}(f)} - 1$, so $P(f)$ holds for the basis.

INDUCTION STEP: Assume that $P(f_1)$ and $P(f_2)$ are true. There are two cases to consider for f derived from f_1 and f_2. If $f = \neg f_1$, then f has one more sub-formula, and has height one greater than f_1, so

CASE 1: If $f = \neg f_1$, then

$$\begin{array}{rrcl} [\text{by } P(f_1)] & \text{height}(f) = \text{height}(f_1) + 1 & \leq & \text{subformula}(f_1) + 1 = \text{subformula}(f) \\ & [\text{by } P(f_1)] & \leq & 2^{\text{height}(f_1)} - 1 + 1 \\ & 2^{\text{height}(f)} \geq 2^{\text{height}(f_1)} + 1 & \leq & 2^{\text{height}(f)} - 1 \end{array}$$

So, in this case $P(f_1) \Rightarrow P(f)$.

CASE 2: If $f = (f_1 * f_2)$, where $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ is a binary connector, then f has one more sub-formula than the sum of those in f_1 and f_2, and so

$$\begin{array}{rrcl} \text{height}(f) = 1 + \max\{\text{height}(f_1), \text{height}(f_2)\} & \leq & 1 + \text{height}(f_1) + \text{height}(f_2) \\ [\text{By } P(f_1),\ P(f_2)] & \leq & 1 + \text{subformula}(f_1) + \text{subformula}(f_2) \\ & = & \text{subformula}(f) \\ [\text{By } P(f_1),\ P(f_2)] & \leq & 1 + 2^{\text{height}(f_1)} - 1 + 2^{\text{height}(f_2)} - 1 \\ & = & 2^{\text{height}(f_1)} + 2^{\text{height}(f_2)} - 1 \end{array}$$

$$
\begin{aligned}
&\leq 2 \times 2^{\max\{\text{height}(f_1),\text{height}(f_2)\}} - 1 \\
&= 2^{1+\max\{\text{height}(f_1),\text{height}(f_2)\}} - 1 \\
[\text{defn of height}(f)] \qquad &= 2^{\text{height}(f)} - 1
\end{aligned}
$$

So, in this case also, $P(f_1)$ and $P(f_2)$ imply $P(f)$, as wanted.

I conclude that $P(f)$ holds for all $f \in F_{PV}$. QED.

QUESTION 3. [10 MARKS]

Consider `subDiv(a,b)` and its precondition/post condition pair:

```
/**
  * Precondition: a, b are natural numbers, b != 0
  * Postcondition: return int[] divMod such that
  *                    a = b * divMod[0] + divMod[1] &&
  *                    0 <= divMod[1] < b
  *
  */
public static int[] subDiv(int a, int b) {
  int[] divMod = {0,a};
  while (divMod[1] >= b) {
    divMod[1] = divMod[1] - b;
    divMod[0] = divMod[0] + 1;
  }
  return divMod;
}
```

PART (A) [5 MARKS]

Prove that if the precondition is satisfied, then `subDiv(a,b)` terminates.

CLAIM: $\texttt{divMod[1]}_i$ is a natural number and the sequence $\langle \texttt{divMod[1]}_i \rangle$ is decreasing.

PROOF: By the precondition, $\texttt{divMod[1]}_0$ is initialized to a, a natural number, and modified by subtracting integer b, so $\texttt{divMod[1]}_i$ is certainly an integer. Furthermore, the while loop only executes (and subtracts b from `divMod[b]`) if `divMod[b]` $>= b$, so (by a very slight induction) after the ith iteration $\texttt{divMod[b]}_i$ is a natural number. If there is an $(i+1)$th iteration, then (by the exit condition) $\texttt{divMod[1]}_i \geq b$, so $\texttt{divMod[1]}_{i+1} = \texttt{divMod[1]}_i - b$, and (since b is a non-zero natural number, by the precondition) $\texttt{divMod[1]}_i > \texttt{divMod[1]}_{i+1}$, so the sequence is decreasing.

CLAIM: If the precondition is satisfied, then `subDiv(a,b)` terminates.

PROOF: In the previous claim we showed that $\langle \texttt{divMod[1]}_i \rangle$ is a decreasing sequence of natural numbers, and hence (PWO) finite. Thus it has a last element, say $\texttt{divMod[1]}_k$, and hence (since each element of the sequence corresponds to an iteration of the loop) the loop iterates only k times. Since all other statements are finite, `subDiv(a,b)` terminates. QED.

PART (B) [5 MARKS]

Prove that if the precondition is satisfied and `subDiv(a,b)` terminates, then when it does the postcondition is satisfied.

CLAIM: Let $P(i)$ be "If there is an ith iteration of the loop then $a = b * \texttt{divMod[0]}_i + \texttt{divMod[1]}_i$." Then $P(i)$ is true for all $i \in \mathbb{N}$.

PROOF (INDUCTION ON i): If $i = 0$, then $P(i)$ simply states that $a = b * 0 + a$, which is certainly true, so the claim holds for the base case.

INDUCTION STEP: Assume that $P(i)$ holds for some arbitrary natural number i. If there is no $(i+1)$th iteration of the loop, then $P(i+1)$ is vacuously true. Otherwise, $\texttt{divMod[1]}_{i+1} = \texttt{divMod[1]}_i - b$, and $\texttt{divMod[0]}_{i+1} = \texttt{divMod[0]}_i + 1$, and (using the IH):

$$\begin{aligned} [\text{by } P(i)] \quad a &= \texttt{divMod[0]}_i \times b + \texttt{divMod[1]}_i \\ [\text{by program}] &= \texttt{divMod[0]}_{i+1} \times b - b + \texttt{divMod[1]}_{i+1} + b \\ &= \texttt{divMod[0]}_{i+1} \times b + \texttt{divMod[1]}_{i+1} \end{aligned}$$

So $P(i) \Rightarrow P(i+1)$, as wanted.

I conclude that $P(i)$ is true for all $i \in \mathbb{N}$. QED.

CLAIM: If the precondition is satisfied and `subDiv(a,b)` terminates, then when it does the postcondition is satisfied.

PROOF: If the precondition is satisfied, then (proved above) `subDiv(a,b)` terminates. When it does, suppose after iteration k, then $0 \leq \texttt{divMod[1]}_k < b$ (by the exit condition, and the fact established in proof of termination). Together with $P(k)$, this means that

$$a = \texttt{divMod[0]}_k \times b + \texttt{divMod[1]}_k \qquad 0 \leq \texttt{divMod[1]}_k < b.$$

This is the postcondition. QED.

QUESTION 4. [10 MARKS]

Prove that program $g(n)$ is correct with respect to the following precondition/postcondition pair:

- precondition: n is a natural number.
- Postcondition: return $2^{n+1} + (-1)^n$

```
public static int g(int n) {
  if (n == 0) return 3;
  if (n == 1) return 3;
  else return g(n-1) + (2 * g(n-2));
}
```

CLAIM: Let $P(n)$ be "$g(n)$ returns $2^{n+1} + (-1)^n$." Then $P(n)$ is true for all $n \in \mathbb{N}$.

PROOF (COMPLETE INDUCTION ON n: Suppose $n = 0$. Then the "if $(n == 0)$" branch is executed, and $g(n)$ returns $3 = 2^{0+1} + (-1)^0$, so $P(0)$ holds. If $n = 1$), then the "if $(n == 1)$" branch is executed, and $g(n)$ returns $3 = 2^{1+1} + (-1)^1$, so $P(1)$ holds. These two verify the bases cases.

INDUCTION STEP: Suppose n is an arbitrary integer greater than 1, and assume that $P(\{0, \ldots, n-1\})$ holds. Then the "else" branch executes, and (since $P(n-1)$ and $P(n-2)$ are assumed, given $0 \leq n-1, n-2 < n$) $g(n)$ returns

$$\begin{aligned}
[\text{By } P(n-1),\ P(n-2)] \qquad g(n-1) + 2g(n-2) &= 2^{n-1} + (-1)^{n-1} + 2 \times 2^{n-2} + 2 \times (-1)^{n-2} \\
&= 2 \times 2^{n-1} + (-1)^{n-1} + (-1)^{n-2} + (-1)^{n-2} \\
[(-1)^{n-2} + (-1)^{n-1} = 0] \qquad &= 2^n + (-1)^{n-2} \\
[(-1)^n = (-1)^{n-2}] \qquad &= 2^n + (-1)^n
\end{aligned}$$

Thus, $P(\{0, \ldots, n-1\})$ implies $P(n)$, as wanted.

I conclude that $P(n)$ is true for all $n \in \mathbb{N}$. QED.

QUESTION 5. [10 MARKS]

State whether each formula is valid or not. Prove your claim.

PART (A) [5 MARKS]

$$\forall x \exists y (\forall y \exists z F(x,y,z) \to \exists x \forall z M(x,y,z)) \leftrightarrow \exists y \forall x (\forall y \exists z F(x,y,z) \to \exists x \forall z M(x,y,z))$$

CLAIM The formula is valid.

PROOF: Let $LF = \forall x \exists y (\forall y \exists z F(x,y,z) \to \exists x \forall z M(x,y,z))$ and $RF = \exists y \forall x (\forall y \exists z F(x,y,z) \to \exists x \forall z M(x,y,z))$. I will show that LF LEQV RF, which means (by definition) that $LF \leftrightarrow RF$ is valid. By the renaming rule I have:

$$\begin{aligned}
\forall x \exists y (\forall y \exists z F(x,y,z) \to \exists x \forall z M(x,y,z)) \quad &\text{LEQV} \quad \forall x \exists y (\forall u \exists v F(x,u,v) \to \exists w \forall t M(w,y,t)) \\
\text{[factoring over } \to\text{]} \quad &\text{LEQV} \quad (\exists x \forall u \exists v F(x,u,v) \to \exists y \exists w \forall t M(w,y,t)) \\
\text{[factoring over } \to \text{ again]} \quad &\text{LEQV} \quad \exists y \forall x (\forall u \exists v F(x,u,v) \to \exists w \forall t M(w,y,t)) \\
\text{[renaming rule again]} \quad &\text{LEQV} \quad \exists y \forall x (\forall y \exists z F(x,y,z) \to \exists x \forall z M(x,y,z))
\end{aligned}$$

Thus LF LEQV RF, so $LF \leftrightarrow RF$ is valid. QED.

PART (B) [5 MARKS]

$$(\forall x \exists z F(x,y,z) \vee \forall x \forall z M(y,x,z)) \leftrightarrow \forall x (\exists z F(x,y,z) \vee \forall z M(y,x,z))$$

CLAIM: The formula is not valid.

PROOF: Let $D = \mathbb{N}$, $F(x,y,z)$ have the interpretation $x + z \leq y$, $M(y,x,z)$ have the interpretation $x + z > y$, and $\sigma(y) = 3$. In this interpretation the left formula becomes $\forall x \exists z F(x,3,z) \vee \forall x \forall z M(3,x,z)$, which is false, since there are x that are greater than 3, and there are pairs $x + y$ whose sum is no more than 3. On the other hand, in this same interpretation, the right formula becomes $\forall x (\exists z F(x,3,z) \vee \forall z M(3,x,z))$, which is true, since for each x, either $x \leq 3$, so we can find a natural number z so that $x + z \leq 3$, or else $x > 3$, so for every $z, x + z > 3$. Since this interpretation falsifies LF but satisfies RF, the biconditional $LF \leftrightarrow RF$ is false. QED.

QUESTION 6. [10 MARKS]

An electrician is designing two-way switches so that the stairway lights can be turned on or off from the first, second, or third floor. During the first part of the design phase she selects the set of propositional variables $\{F, S, T\}$ with the intended meanings:

- F means the first-floor switch is up, $\neg F$ means the first-floor switch is down.
- S means the second-floor switch is up, $\neg S$ means the second floor switch is down.
- T means the third-floor switch is up, $\neg T$ means the third-floor switch is down.

Write down a DNF and a CNF formula for the electrician that are true exactly when when an even number of switches are up.

SOLUTION: List all 8 configurations of the propositional variables (1 for up, 0 for down), with the corresponding minterm or maxterm.

F	S	T	Even number up	minterm	maxterm
0	0	0	1	$\neg F \wedge \neg S \wedge \neg T$	
0	0	1	0		$F \vee S \vee \neg T$
0	1	0	0		$F \vee \neg S \vee T$
0	1	1	1	$\neg F \wedge S \wedge T$	
1	0	0	0		$\neg F \vee S \vee T$
1	0	1	1	$F \wedge \neg S \wedge T$	
1	1	0	1	$F \wedge S \wedge \neg T$	
1	1	1	0		$\neg F \vee \neg S \vee \neg T$

The DNF formula is the disjunction of the minterms:

$$(\neg F \wedge \neg S \wedge \neg T) \vee (\neg F \wedge S \wedge T) \vee (F \wedge \neg S \wedge T) \vee (F \wedge S \wedge \neg T)$$

The CNF formula is the conjunction of the maxterms:

$$(F \vee S \vee \neg T) \wedge (F \vee \neg S \vee T) \wedge (\neg F \vee S \vee T) \wedge (\neg F \vee \neg S \vee \neg T)$$

QUESTION 7. [10 MARKS]

The electrician from the previous questions is continuing to design two-way switches so that the stairway lights can be turned on or off from the first, second, or third floor. She selects an alphabet $\Sigma = \{F, f, S, s, T, T\}$ with the intended meaning:

- F means the first-floor switch is flipped up, f means the first-floor switch is flipped down.
- S means the second-floor switch is flipped up, s means the second-floor switch is flipped down.
- T means the third-floor switch is flipped up, t means the third-floor switch is flipped down.

Strings over Σ represent a sequence of switch flips, for example $FSfT$ means the first-floor switch is flipped up, then the second-floor switch is flipped up, then the first-floor switch is flipped down, then the third-floor switch is flipped up.

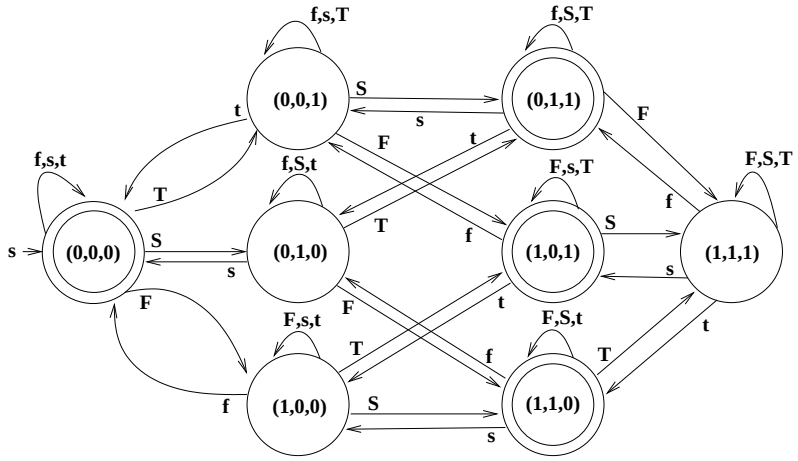
Our electrician is interested in those sequences of events that leave an even number of switches up. $FSfT$ leaves an even number of switches up, since only the second- and third-floor switches are up when the sequence finishes. On the other hand, $FSfTs$ leaves an odd number of switches up, since only the third-floor switch is up when this sequence finishes. There are even peculiar sequences such as FFS, corresponding to a frustrated individual who flips the first-floor switch up twice in a row (not noticing that it is already up), and then someone flips the second-floor switch up. Finally, the electrician assumes that the empty sequence, ε, leaves 0 switches up, and she's interested in the language, L:

$$L = \{x \in \Sigma^* : x \text{leaves an even number of switches up}\}.$$

PART (A) [5 MARKS]

Draw a DFSA, M, that accepts L.

SOLUTION: Keep track of which light switches are up with the triple (first,second,third), where a 1 indicates that the corresponding switch is up, and a 0 indicates that the corresponding switch is not up.



PART (B) [5 MARKS]

Write down a state invariant that helps show that M, indeed, accepts L. You are NOT required to provide a full proof that $L(M) = M$.

SOLUTION:

$$\delta^*(s,x) = \begin{cases} (0,0,0), & \text{if no switches are up} \\ (0,0,1), & \text{if only the third-floor switch is up} \\ (0,1,0), & \text{if only the second-floor switch is up} \\ (0,1,1), & \text{if only the second- and third-floor switches are up} \\ (1,0,0), & \text{if only the first-floor switch is up} \\ (1,0,1), & \text{if only the first- and third-floor switches are up} \\ (1,1,0), & \text{if only the first- and second-floor switches are up} \\ (1,1,1), & \text{if all three switches are up} \end{cases}$$

QUESTION 8. [10 MARKS]

The electrician from the previous question decides it will be easier to discuss her design by email if she denotes L by a regular expression. She is annoyed by the amount of work needed to translate her DFSA directly into a regular expression (using the approach from her CSC236 Course Notes), so she decides to solve a smaller problem first, without using a DFSA. In this problem she only worries about the first two floors, so $\Sigma = \{F, f, S, s\}$, and L is defined

$$L = \{x \in \Sigma^* : \text{x leaves an even number of switches up}\}$$

PART (A) [5 MARKS]

Write a regular expression that denotes L. Justify your answer (no need to translate from a DFSA).

SOLUTION: Break up the required strings into three sets:

- Strings that leave two switches up either leave the first-floor switch up and then the second, or else the second-floor switch up and then the first. These strings have a prefix that is an arbitrary string in Σ^*, followed by a sequence that leaves the two switches up in one of the two orders:

$$R_2 = (F + f + S + s)^*(F(s + S)^*S + S(f + F)^*F)$$

- Strings that leave zero switches up and explicitly flip (and leave) the first-floor switch down, and then the second, or else explicitly flip (and leave) the second floor switch down and then the first. These strings have a prefix that is an arbitrary string in Σ^*, followed by a sequence that flips (and leaves) the two switches down in one of the two orders:

$$R_{0,2} = (F + f + S + s)^*(f(S + s)^*s + s(F + f)^*f)$$

- Strings that leaved zero switches up but never explicitly flip down at least one of the two switches (and consequently, have never flipped that switch up). These strings contain characters from either $\{S, s\}$ or $\{F, f\}$, but not both, and (if they are non-empty) end with a flip down, so they can be written as zero or more blocks terminated by a down-flip:

$$R_{0,F} = (f^*F^*f)^* \qquad R_{0,S} = (s^*S^*s)^*$$

These cases exhaust the possibilities, and all strings denoted by these cases are in L, so

$$L = L(R_2 + R_{0,2} + R_{0,F} + R_{0,S}).$$

PART (B) [5 MARKS]

Now let $\Sigma = \{F, f, S, s, T, t\}$, and solve the original problem. Let L be defined:

$$L = \{x \in \Sigma^* : x \text{ leaves an even number of switches up}\}$$

Write a regular expression that denotes L. Justify your answer. WARNING: This is a lot of work for 5 marks. HINT: It is probably easiest to come up with the regular expression WITHOUT designing a DFSA.

SOLUTION: Break the strings up into cases:

- Strings that leave two switches up and omit up-flips on the other floor. The regular expressions that denote these strings indicate the last up-flip on each of the switches that are left up:

$$\begin{aligned} R_{2,\neg F} &= (f+S+s+T+t)^*(T(f+S+s)^*Sf^* + S(f+T+t)^*Tf^*) \\ R_{2,\neg S} &= (F+f+s+T+t)^*(F(T+t+s)^*Ts^* + T(T+t+s)^*Fs^*) \\ R_{2,\neg T} &= (F+f+S+s+t)^*(F(S+s+t)^*St^* + S(F+f+t)^*Ft^*) \end{aligned}$$

- Strings that leave two switches up and explicitly down-flip the switch for the other floor. These strings are a superset of the strings that leave two switches up and include up-flips for all three floors. The regular expressions for these strings must indicate where the last down-flip for floor with its switch left down occurs, relative to the last up-flips for the switches left up:

$$\begin{aligned} R_{FS} &= (F+f+S+s+T+t)^* \\ &\quad (\quad t(F+f+S+s)^*(F(S+s)^*S + S(F+f)^*F) \\ &\qquad + (F(S+s+T+t)^*t(S+s)^*S + S(F+f+T+t)^*t(F+f)^*F) \\ &\qquad + (F(S+s+T+t)^*S(T+t)^*t + F(S+s+T+t)^*S(T+t)^*t) \quad) \\ R_{FT} &= (F+f+S+s+T+t)^* \\ &\quad (\quad s(F+f+T+t)^*(F(T+t)^*T + T(F+f)^*F) \\ &\qquad + (F(T+t+S+s)^*s(T+t)^*T + T(F+f+S+s)^*s(F+f)^*F) \\ &\qquad + (F(T+t+S+s)^*T(S+s)^*s + T(F+f+S+s)^*F(S+s)^*s) \quad) \\ R_{ST} &= (F+f+S+s+T+t)^* \\ &\quad (\quad f(S+s+T+t)^*(S(T+t)^*T + T(S+s)^*S) \\ &\qquad + (S(T+t+F+f)^*f(T+t)^*T + T(T+t+F+f)^*f(S+s)^*S) \\ &\qquad + (S(T+t+F+f)^*T(F+f)^*f + T(T+t+F+f)^*S(F+f)^*f) \quad) \end{aligned}$$

- Strings that leave zero switches up but explicitly down-flip at most one switch (and hence never up-flip the other two):

$$R_{0,F} = (f^*F^*f)^* \qquad R_{0,S} = (s^*S^*s)^* \qquad R_{0,T} = (t^*T^*t)^*$$

- Strings that leave zero switches up and explicitly down-flip two (but not three) switches:

$$\begin{aligned} R_{0,2} &= (F+f+S+s)^*(f(S+s)^*s + s(F+f)^*f) + (F+f+T+t)^*(f(T+t)^*t + t(F+f)^*f) \\ &\quad + (S+s+T+t)^*(s(T+t)^*t + t(S+s)^*s) \end{aligned}$$

- Strings that leave zero switches up and explicitly down-flip three switches. These regular expressions specify the order of the last down-flip on each switch.

$$\begin{aligned} R0,3 \;=\; & (F+f+S+s+T+t)^* \\ & (\quad f(S+s+T+t)^*s(T+t)^*t + f(S+s+T+T)^*t(S+s)^*s \\ & \quad + s(F+f+T+T)^*f(T+t)^*t + s(T+t+F+t)^*t(F+f)^*f \\ & \quad + t(F+f+S+s)^*f(S+s)^*s + t(S+s+F+f)^*s(F+f)^*f \quad) \end{aligned}$$

These cases exhaust the possibilities, so L is the union of the languages they denote:

$$L \;=\; L(R_{\neg F} + R_{\neg T} + R_{\neg S} + R_{FS} + R_{FT} + R_{ST} + R_{0,S} + R_{0,F} + R_{0,T} + R_{0,2} + R_{0,3})$$

QUESTION 9. [10 MARKS]

Which of the following claims are true, which are false? Justify your answers.

PART (A) [2 MARKS]

Let prefix$(L) = \{x \in \Sigma^* : xy \in L$, for some $y \in \Sigma^*\}$. Claim: if prefix(L) is regular, then L is regular.

SOLUTION: The claim is false. Let L be the language of binary strings with an equal number of 0s and 1s. The set of prefixes for this language is $(0+1)^*$, since any binary string can have a suffix added to balance the number of 0s and 1s. But L (from class) is not regular, whereas its prefix language $(0+1)^*$ is regular.

PART (B) [2 MARKS]

Suppose R, S, and T are regular expressions. Claim: If $R(S+1) \equiv T(S+1)$, then $R \equiv S$.

SOLUTION: The claim is false. Let $S = (0+1)^*$ (all binary strings), so $(S+1) = S$. Let $R = (0+1+\varepsilon)$ and $T = \varepsilon$. Then $R(S+1) = (0+1+\varepsilon)(0+1)^* = (0+1)^*$. On the other hand, $T(S+1) = (0+1)^*$. Thus $R(S+1) \equiv T(S+1)$, but $R \not\equiv T$.

PART (C) [2 MARKS]

Let $L = \{x \in \{0,1\}^* : \text{each 0 in } x \text{ is followed by exactly two 1s}\}$. Claim: L is regular.

SOLUTION: The claim is true, since $L = L(1^*(011)^*)$.

PART (D) [2 MARKS]

Let $L = \{x \in \{0,1\}^* : x \text{ contains exactly twice as many 1s as 0s}\}$. Claim: L is regular.

SOLUTION: The claim is false. If L were regular, then (for pumping length p) choose $x = 0^p 1^{2p}$. Then $x \in L$, and $x = uvw$, where $v = 0^j$, some $1 \leq j \leq p$, but $uv^0w \notin L$. Contradiction.

PART (E) [2 MARKS]

Let $L = \{x \in \{0,1\}^* : x \text{ contains more 1s than 0s and } |x| < 50\}$. Claim: x is regular.

SOLUTION: Presumably, that should be "L is regular." The claim is true, since L has finitely many strings (certainly no more than 2^{51}). A regular expression composed of all these strings denotes L.

[This page left nearly blank for answers that didn't fit elsewhere]

Total Marks = 90