1. Consider the following code for checking if a 3 character password is entered correctly:

```c
check1(key, input) {
    for (int i = 0; i < 3; i++)
        if (key[i] != input[i])
            return 0;
    grant_access();
    return 1;
}
check2(key, input) {
    if (key == input) {
        grant_access();
        return 1;
    }
    else
        return 0;
}
```

Assume that each password character is a lowercase alphabet character. (a) Assuming that the symbolic execution framework tracks the number of instructions executed using a variable $O$, generate the path constraints for the above procedures. (b) Use model counting the determine the probabilities for each path constraint and compute the information leakage using entropy assuming that the value of $O$ is observable.

2. Consider the following procedure where $S$ is a secret value and $I$ is a public input.

```c
P(S, I) {
    if (S > I)
        sleep(1);
    else
        sleep(2);
}
```

Assume that $S$ and $I$ are both 4 bit values. Generate the path constraints for this procedure. Represent the amount of information leakage in terms of entropy for a single execution of $P$ as a function of $I$ (first write a function of $I$ that represents the number of models for a given $I$). Then, show the value of $I$ that maximizes the information leakage.
3. Consider the following procedures where H and L are high and low security variables, respectively.

P1(H) {
  if (H)
    L = 0;
  else
    L = 1;
}

P2(H) {
  if (H)
    X = 0;
  else
    X = 1;
  L = X & S;
}

Use self-composition to transform these procedures so that the non-interference property can be expressed as a safety property. State the non-interference property for the transformed procedures. Do these procedures satisfy the non-interference property?

4. Consider the following procedure:

P(H) {
  if (H > 0)
    result = 1;
  else
    result = 0;
  return result;
}

Use the quantitative information flow analysis based on bounded model checking to check if the channel capacity for this procedure is 1. First, show the template code calling this procedure (which should include an assertion that limits the channel capacity to 1). Show the formula that bounded model checking technique will generate based on the template code. Is the generated formula satisfiable and what can you conclude from the satisfiability of the generated formula?

5. (a) Write an HyperLTL property that represents non-interference. Show traces that satisfy the property and traces that violate the property. (b) Write an HyperLTL property that limits the channel capacity of the leakage to 2 bits. Show traces that satisfy the property and traces that violate the property.