Question 1. [8 marks]

Part (a) [5 marks]

Here is a recursive definition:

$$T(n) = \begin{cases} c_1, & \text{if } n = 1 \\ c_2, & \text{if } n = 2 \\ T(n-2) + c_3, & \text{if } n > 2 \end{cases}$$

Find a closed form for this recursive definition. Carry out the first four steps of repeated substitution. As a reminder, these four steps are: substitute a few times, guess the recurrence, find appropriate value(s) for k, and substitute for k.

Part (b) [3 marks]

Prove using induction that the closed form you came up with is correct.

Question 2. [6 MARKS]

Figure out the precondition and postcondition of the following recursive algorithm, and then prove its correctness:

```python
def do_something(a, x):
    if x == 0:
        return a
    else:
        return do_something(a+1, x-1)
```

Question 3. [8 MARKS]

Figure out the precondition and postcondition of the following recursive algorithm, and then prove its correctness (find loop invariant, prove loop invariant, find variant and prove termination):

```python
def f(b, x):
    r = 0
    count = x
    while count > 0:
        r += b
        count -= 1
    return r
```

Question 4. [5 MARKS]

For function `peak` below, you are given a list that ascends up to some maximum value and then descends from there until the end of the list. For example, the list `[2, 5, 8, 12, 15, 7, 4]` is allowed but `[5, 9, 14, 7, 6, 10]` is not.

Develop a recursive algorithm that finds the maximum element in such a list. Full marks will be awarded for giving a correct algorithm that is the tightest asymptotic bound. No sorting or modifying the list is permitted.

```
def peak(lst, low, high):
  '''
  Pre: lst is a list of integers that ascends and then descends,
       0 <= low <= high <= len(lst) - 1.
  Post: return the maximum element in lst.
  '''


  # ... YOUR CODE HERE ...
```

Question 5. [3 MARKS]

Give the recurrence relation in terms of n where n is the size of the list for the above D and C algorithm. The first call to this would be peak(A, 0, len(A)-1). Then, use Master theorem and give a Big O bound.

Question 6. [5 MARKS]

Recall from tutorial that the *reversal* of a string s is a string s^R that reverses the order of the characters of s. For example, $(aabc)^R = cbaa$, and $\epsilon^R = \epsilon$.

Now we define a *reversal operator Rev* that takes as input a regular language and outputs a language containing the set of all reversals of strings in the original language:

$$Rev(L) = \{w^R \mid w \in L\}.$$

Now, recall from lecture that the set of all regular languages over an alphabet Σ is recursively defined as follows:

- $\{\}$, the empty set, is a regular language.
- $\{\epsilon\}$, the language consisting of only the empty string, is a regular language.
- For any symbol $a \in \Sigma$, $\{a\}$ is a regular language.
- If L, M are regular languages, then so are $L \cup M$, LM, and L^*.

Using **structural induction**, prove that the following claim is true for all regular languages (as defined above):

If L is a regular language, then so is $Rev(L)$.

Question 7. [9 MARKS]

Part (a) [6 MARKS]

Design a 5-state DFA that accepts $L = \{w \in \{0,1\}^* \mid |w| \geq 2 \text{ and the first two symbols of } w \text{ are not the same}\}$. Then, prove that your DFA is correct by giving and proving state invariants.

Part (b) [3 MARKS]

Prove that this DFA must have a minimum of 5 states.