You may work with one other person on this assignment. If you do so, turn in one copy with both your names on it.

There will be a third problem added shortly.

(1) Consider the following sorting algorithm:

```python
def bsort(L):
    # precondition: L is a non-empty list
    result = L[:]  # copy of L
    for k in range(1, len(L)):
        for i in range(0, len(L) - k):
            if result[i] > result[i + 1]:
                temp = result[i + 1]
                result[i + 1] = result[i]
                result[i] = temp
    return result
```

# postcondition: returns an array with the same elements as L,
# in non-decreasing order

Prove correctness of the algorithm:
(a) State an appropriate invariant for the outer loop.
(b) State an appropriate precondition and postcondition for the inner loop.
(c) Prove the inner loop is correct for this precondition and postcondition.
(d) Prove the method is correct.
(2) Consider the merge sort algorithm, which breaks the list in half, recursively sorts each half, and then merges the two halves together:

```python
def merge_sort(L):
    '''
    Given a list, returns the list in non-descreeasing order.
    '''
    if len(L) == 0 or len(L) == 1:
        return L
    else:
        middle = ((len(L) - 1) / 2) + 1
        left = L[:middle]
        right = L[middle:]
        left = merge_sort(left)
        right = merge_sort(right)
        return merge(left, right)

def merge(L1, L2):
    '''
    Given two sorted lists, return a new list of all their elements again in sorted order.
    '''
    i = 0
    j = 0
    result = []
    while i < len(L1) and j < len(L2):
        if L1[i] < L2[j]:
            result.append(L1[i])
            i += 1
        else:
            result.append(L2[j])
            j += 1
    while i < len(L1):
        result.append(L1[i])
        i += 1
    while j < len(L2):
        result.append(L2[j])
        j += 1
    return result
```

For each positive $m \in \mathbb{N}$, let $\phi_m(n)$ be the number whose $m$-bit binary representation is the reverse of the one for $n$.

For example, since $5 = 101_2 = 000101_2$, $\phi_6(5) = 101000_2 = 40_{10}$. For $m = 3$, $\phi_3(0) = 000_2$, $\phi_3(1) = 001_2$, $\phi_3(2) = 010_2$, $\phi_3(3) = 011_2$, $\phi_3(4) = 100_2$, $\phi_3(5) = 101_2$, $\phi_3(6) = 110_2$, $\phi_3(7) = 111_2$.

Prove that for each positive $m \in \mathbb{N}$, $\text{merge-sort}$ of $\phi_m(0), \phi_m(1), \ldots, \phi_m(2^m-1)$ requires the maximum number of comparisons out of all sequences of $2^m$ elements.

A comparison is when an element of the list is compared ($<$, $>$, etc.) to another element of the list.
(3) For $n \in \mathbb{N}$, define $PF$ as the recursive datatype of Propositional Formulas:

- Literal: $T \in PF$ and $F \in PF$.
- Propositional Variable: $X_i \in PF$ for each $i \in \mathbb{N}$.
- Negation: for each $p \in PF$, $\neg p \in PF$.
- Conjunction, disjunction, implication and biconditional.
  - For each $p, q \in PF$: $p \land q$, $p \lor q$, $p \rightarrow q$ and $p \leftrightarrow q$ are in $PF$.

Prove that each $p \in PF$ is logically equivalent to a $q \in PF$ where:

If $\neg r$ occurs somewhere in $q$, then $r$ is a Propositional Variable.

Do this by giving a recursive algorithm to produce such a $q$ from $p$, and prove it correct.

Direct Structural Induction will not be enough: instead induct on the recursive/nesting depth — or height, depending on one’s point of view — of the expression tree for the formula (or some other measure of formula size if your algorithm makes recursive calls on formulas of the same or greater depth).