Name: Key

1. (6 points) The algorithm MINIMUM returns the minimum value in a non-empty, numeric array A of length $A.length$. Prove that the algorithm is correct using the following loop invariant: *At the beginning of an iteration, min is equal to the smallest value in* $A[1..(j-1)]$.

```
MINIMUM(A)
1  min = A[1]
2  for j = 2 to A.length
3      if min ≥ A[j]
4          min = A[j]
5  return min
```

Initialization: At the beginning of the first iteration, $j=2$, and the invariant is "min is equal to the smallest value in $A[1..(2-1)] = A[1]$. This is true since min is set equal to $A[1]$ on line 1.

Maintenance: Suppose the invariant is true at the start of an iteration, so min is equal to the smallest value in $A[1..(j-1)]$. On lines 3 and 4, if $A[j] \leq min$, then $A[j]$ is equal to the smallest value in $A[1..j]$, and min is set equal to $A[j]$, so min is now equal to the smallest value in $A[1..j]$. If $A[j] > min$, min is unchanged, so min is equal to the smallest value in $A[1..j]$. This is the correct statement of the invariant for the next iteration.

Termination: The loop terminates with $j = A.length + 1$, so the invariant gives that min is equal to the smallest value in $A[1..(A.length + 1 - 1)] = A[1..A.length]$.

(continued on other side)

2. (4 points) The algorithm PREFIX-SUM computes the prefix sums of a non-empty, numeric array A. That is, for each $1 \le j \le A.length$, it computes $\sum_{i=1}^{j} A[i]$. State a loop invariant for the for-loop in PREFIX-SUM and show that when the loop terminates, the invariant gives a useful property that helps show that the algorithm is correct.

```
PREFIX-SUM(A)
  B is a new array of length A.length
  B[1] = A[1]
  for j = 2 to A.length
      B[j] = A[j] + B[j - 1]
  return B
```

Invariant: At the beginning of an iteration,

$$B[k] = A[1] + A[2] + \cdots + A[k]$$

for $1 \le k < j$.

Termination: The loop terminates with $j = A.length + 1$, so the invariant states that

$$B[k] = A[1] + A[2] + \cdots + A[k]$$

for $1 \le k < A.length$, or, equivalently,

$$B[k] = \sum_{i=1}^{k} A[i] \quad \text{for } 1 \le k \le A.length$$

So the algorithm correctly computes the prefix sums of A.

Name: $\boxed{\text{Key}}$

1. (6 points) The algorithm SUM returns the sum of the values in a non-empty, numeric array A of length $A.length$. Prove that the algorithm is correct using the following loop invariant: *At the beginning of an iteration, the variable sum is equal to the sum of the values in* $A[1\,..\,(j-1)]$.

```
SUM(A)
1  sum = A[1]
2  for j = 2 to A.length
3      sum = sum + A[j]
4  return sum
```

Initialization: At initialization, $j=2$, and the invariant states that sum is equal to the sum of the values in $A[1..(j-1)] = A[1]$. Since sum is set equal to $A[1]$ on line 1, this is true.

Maintenance: At the start of an iteration, sum is equal to the sum of the values in $A[1..(j-1)]$, or

$$\text{sum} = A[1] + A[2] + \cdots + A[j-1]$$

Line 3 updates sum to

$$\text{sum} = \text{sum} + A[j] = A[1] + A[2] + \cdots + A[j-1] + A[j]$$

That is, sum is now the sum of the values in $A[1..j]$ which is the correct statement of the invariant for the next iteration.

Termination: The loop terminates with $j = A.length + 1$, and the invariant states that sum is the sum of the values in $A[1..(A.length+1)-1] = A[1..A.length]$. Therefore, the algorithm correctly computes the sum.

(continued on other side)

2. (4 points) The algorithm ARRAY-SUM computes the element-wise sum of two non-empty, numeric arrays A and B of the same length. That is, for each $1 \leq j \leq A.length$, it computes $C[j] = A[j] + B[j]$. State a loop invariant for the for-loop in ARRAY-SUM and show that when the loop terminates, the invariant gives a useful property that helps show that the algorithm is correct.

```
ARRAY-SUM(A, B)
C is a new array of length A.length
for j = 1 to A.length
    C[j] = A[j] + B[j]
return C
```

Invariant: At the start of an iteration, $C[k] = A[k] + B[k]$ for $1 \leq k < j$.

Termination: At termination, $j = A.length + 1$, and the invariant states that

$$C[k] = A[k] + B[k]$$

for $1 \leq k < A.length + 1$, or, equivalently, for $1 \leq k \leq A.length$. We can conclude immediately that the algorithm correctly computes the sum of the arrays.