1. The following algorithm takes as input a positive int \( n \), an array of \( n \) ints \( A \), and an int \( val \). It returns the number of times that \( val \) occurs in \( A \).

   ```c
   int Count_matches(int A[], int n, int val) {
       int matches = 0;
       for (i = 0; i < n; i++)
           if (A[i] == val) matches++;
       return matches;
   }
   ```

   In order to prove the algorithm is correct, we would define a loop invariant: a predicate \( P(i) \) that depends on the iteration \( i \). We choose the invariant so that

   - It is true after each iteration
     
     Test \( i < n; \)
     
     if (A[i] == val) matches++;
     
     i++;

   - And if it is true after iteration \( i = n-1 \), then the for statement will have correctly counted the number of times \( val \) occurs in \( A \).

   What is the loop invariant \( P(i) \)?

   Note that you do not need to prove that the algorithm is correct.

   After iteration \( i \) is complete \( \text{matches} \) is the number of occurrences of \( val \) in the list \( A[0], A[1], \ldots, A[i] \).

2. Recall the sorting algorithm selection sort.
```c
void Sel_sort(int A[], int n) {
    int i, j, min_loc, tmp;

    for (i = 0; i < n-1; i++) {
        min_loc = i;
        for (j = i+1; j < n; j++)
            if (A[j] < A[min_loc]) min_loc = j;
        tmp = A[i];
        A[i] = A[min_loc];
        A[min_loc] = tmp;
    }
} /* Sel_sort */
```

It takes as input a positive int $n$ and an array $A$ of $n$ ints. It returns in $A$ the same ints sorted into increasing order.

(a) What is a loop invariant $Q(j)$ for the inner loop?

(b) What is a loop invariant $R(i)$ for the outer loop?

You do not have to prove that the algorithm is correct.

(a) After iteration $j$ is complete, $\text{min \_loc}$ stores the subscript of the smallest element in the list $A[i]$, $A[i+1]$, ..., $A[n-1]$.

(b) After iteration $i$, is complete the list $A[0]$, $A[1]$, ..., $A[i]$ contains the smallest $i+1$ elements in the input list, and these elements are sorted in increasing order.

There are other possibilities here. For example, $A[i]$ is is the $(i+1)$st smallest element in $A$, and the elements $A[0]$, $A[1]$, ..., $A[i-1]$ are unchanged by the iteration.