CSC236H Tutorial 7

Problem Set

1. Let L be a list of integers.
   Let $L[p:q]$ be an nonempty slice, i.e., $0 \leq p < q \leq len(L)$.
   We say that $L[p:q]$ is *unimodal* iff there is a natural number m such that

   (a) $p \leq m < q$;

   (b) $L[p:m+1]$ strictly increasing;

   (c) $L[m:q]$ is strictly decreasing.

   Furthermore, such a number m, if it exists, is call the *mode* of $L[p:q]$.
   Since L is the same as $L[0:len(L)]$, we also say that L is unimodal if $L[0:len(L)]$ is unimodal.

   Give a proof of correctness for the program below with respect to the given specification.
   Precondition: A is a list of integers, $0 \leq p < q \leq len(A)$, $A[p:q]$ is unimodal.
   Postcondition: Return the maximum element of $A[p:q]$.

   ```
   def Max(A, p, q):
       if p + 1 == q:
           return A[p]
       else:
           m = ⌊(p+q)/2⌋
           if A[m − 1] < A[m]:
               return Max(A, m, q)
           else:
               return Max(A, p, m)
   ```

   Hint 1: Notice that the maximum element of a unimodal slice occurs at its mode. Hint 2: For any integers a, b such that $a + 1 < b$,

   $$a < \lfloor \frac{a+b}{2} \rfloor < b.$$

2. For each of the following programs:

   - give an appropriate loop invariant (LI) for the loop;
   - Use your LI and the loop exit condition to prove partial correctness;
   - Define an appropriate loop measure m which you can use for proving the termination of the loop.

   (a) **Precondition:** x is a non-empty string.
   Postcondition: returns x except with all the characters in reverse order

   ```
   def backwards_string(x):
       i = 0; word = ''
       while i < len(x):
   ```

```
        word = x[i] + word
        i = i + 1
    return word
```

(b) **Precondition:** *word* is a non-empty string of small alphabetic characters.
Postcondition: returns true iff *word* is palindrome.

```
def Is_palindrome(word) :
    start_idx = 0; end_idx = len(word) − 1; result = True;
    while start_idx < end_idx:
        if word[start_idx] != word[end_idx] :
            result = False
        start_idx = start_idx + 1
        end_idx = end_idx − 1
    return result
```

3. A *majority element* in a list is an element that appears in (strictly) more than half of the list locations. Consider the following algorithm that finds a majority element in an array, if one exists (if one doesn't exist the algorithm returns an arbitrary wrong answer). Write a detailed proof that the algorithm is partially correct.

Precondition: A is a list of integers and $1 \leq len(A)$.
Postcondition: If A has an element that appears in more than half of the cells of A, then that element is equal to m at return.

```
def Majority(A):
    c = 1
    m = A[0]
    i = 1
    while i < len(A):
        if c == 0:
            m = A[i]
            c = 1
        else if A[i] == m:
            c = c + 1
        else:
            c = c − 1
        i = i + 1
    return m
```