```
"""Joshua Cole  Problem 1-3a: Unimodal Search

Give an algorithm to compute the maximum element of a unimodal input
array in O(lg(n)) time. Prove the correctness of your algorithm, and prove
the bound on its running time.

Algorithm accepts a unimodal array for its input.

  Step 1: Calculate the middle of the list.

  Step 2: If the middle element is the unimodal item, the algorithm
  returns it.

  Step 3: If the middle element is in the decreasing section, give
  the array a[0: middle] to step 1.

  Step 4: If te middle element is in the increasing section, give
  the array a[middle:end] to step 1.

In all cases the algorithm either gets smaller or ends assuming it has been
given valid input. Therefore, the algorithm will terminate. If this algorithm
was converted into a loop (which it can be trivially (see below for the code))
then the algorithm would also have a loop invariant in which all discarded
elements are not the maximum. This proves the correctness of the algorithm.

It does not prove the correctness of the code.

The algorithm is halving its input each time while doing a constant amount of
work. This is similar to binary search, so we can expect to have a running
time of lg(n).

Proof by master method of the bound:

T(n) = T(n/2) + Theta(1)
a = 1, b = 2 c = Theta(1)
 log_2(1)
n         =  1 vs Theta(1)
The two are equal. Therefore case two of the master method is used:
 log_2(1)
n        lgn = lgn
Therefore, the algorithm is lgn.
"""
def in_increasing_section(s, m):
    return s[m-1] < s[m] < s[m+1]

def in_decreasing_section(s, m):
    return s[m-1] > s[m] > s[m+1]

def on_unimodal(s, m):
    return s[m] > s[m-1] and s[m] > s[m+1]

def max_unimodal(s):
    """Returns the index of the maximum value in the unimodal sequence.

    Args:
      s: A unimodal sequence.

    Returns:
      A unimodal sequence is a sequence in which:

        a[i] < a[i + 1] for all indexes below and including m
        a[i] > a[i - 1] for all indexes above and including m

      This function returns m.

    Raises:
      TypeError: List was empty.
      TypeError: List was not unimodal.
    """
    lower_bound = LOWEST = 0
```

```
upper_bound = HIGHEST = len(s) - 1
if lower_bound > upper_bound:
    raise TypeError("List was empty.")
elif not upper_bound:
    return 0
while lower_bound < upper_bound:
    middle = upper_bound - (upper_bound - lower_bound) / 2
    if middle == LOWEST:
        return middle if s[middle] > s[middle + 1] else middle + 1
    elif middle == HIGHEST:
        return middle if s[middle] > s[middle - 1] else middle - 1
    elif on_unimodal(s, middle):
        return middle
    elif in_increasing_section(s, middle):
        lower_bound = middle
    elif in_decreasing_section(s, middle):
        upper_bound = middle
raise TypeError("The list was not unimodal.")
```