CP2410 Practical 03 - Recursion and Array-Based Sequences

1. (R-4.1) Describe a recursive algorithm for finding the maximum element in a sequence, S, of n elements. What is your running time and space usage?

2. (R-4.2) Draw the recursion trace for the computation of power(2,5), using the traditional function implemented below:

```
def power(x, n):
    """Compute the value x**n for integer n."""
    if n == 0:
        return 1
    else:
        return x * power(x, n - 1)
```

3. (R-4.3) Draw the recursion trace for the computation of power(2,18), using the repeated squaring algorithm, as implemented below:

```
def power(x, n):
    """Compute the value x**n for integer n."""
    if n == 0:
        return 1
    else:
        partial = power(x, n // 2)  # rely on truncated division
        result = partial * partial
        if n % 2 == 1:  # if n odd, include extra factor of x
            result *= x
        return result
```

4. (C-4.12) Give a recursive algorithm to compute the product of two positive integers, m and n, using only addition and subtraction.

5. Modify ch05/experiment_list_append.py to investigate the time taken by append operations for DynamicArray (ch05/dynamic_array.py).

6. Create a modified version of DynamicArray (ch05/dyanmic_array.py) that takes a parameter, resize_factor, which it uses to determine the new size (rather than doubling in the original code - `self._resize(2 * self._capacity)`). Using different values of resize_factor, examine if and how the average time to append changes.