Question 1. Find the $O$-notation for the worst-case running time of the following algorithm. You do not need to, nor should you, give the leading constant or the $n_0$ value. Assume the first listed function is called with a sorted vector.

```cpp
int binarySearch(const std::vector<int> & numbers, int target);
int binarySearch(const std::vector<int> & numbers, int target, int low, int high);

int binarySearch(const std::vector<int> & numbers, int target)
{
    return binarySearch(numbers, target, 0, numbers.size() - 1);
}

int binarySearch(const std::vector<int> & numbers, int target, int low, int high)
{
    if( low > high )
    {
        throw ElementNotFoundException("Element not found by binary search.");
    }
    int mid = (low + high) / 2;
    int e = numbers[mid];
    if( e == target )
        return mid;
    else if (target < e)
        return binarySearch(numbers, target, low, mid-1);
    else
        return binarySearch(numbers, target, mid+1, high);
}
```
Question 2. We saw that $O$ notation provides an upper bound on the growth rate of a function. What are $\Omega$ and $\Theta$ used to describe?

Question 3. $f(n) = \log_{10} n$ and $g(n) = \log_2 n$. How do they relate?

Question 4. $f(n) = \log n$. What base do I mean?

Suppose we modify the array-based Stack and/or Queue in the following fashion. When adding to the structure (push or enqueue), if it is full, instead of throwing an exception, we do the following instead:

- Create a new object of twice the size of the current elements.
- Copy everything from elements to this new array. Note: if you elect to implement this, and I encourage you to give this a try, be careful when doing this for the queue. Copy to the new array’s element 0 starting at front, not at 0, and be sure to set front and rear appropriately when done.
- Delete the old elements
- Set the array pointer to point to this new one.

Previously, push/enqueue took $O(1)$ time. Now they take $O(1)$ most of the time, but sometimes they take $O(n)$, where $n$ is the number of elements in the structure.

Question 5. Is it correct to say this new procedure takes $O(n)$?

Question 6. Is it correct to say this new procedure takes $\Theta(n)$?

Question 7. What is an accurate way to describe the running time of the new procedure?