Question 1

1. Develop well-documented pseudo code that finds all the elements of a given array (of any size n) that are multiple of x. The code must display the indices and the values of these elements. For instance, given an array A :
    - (22, 61,-10, 21, 0, 9, 50, 17, 35, 81,-46, 19, 5, 77) with x as 5, your code should find and display something similar to the following (notice that this is just an example. Your solution must not refer to this particular example):
        - The elements of the array A that are multiple of 5 are:
        - Index 2 with value -10
        - Index 6 with value 50
        - Index 8 with value 35
        - Index 12 with value 5

```
Algorithm MultiplesOfX (A, x)
    Input: Array<int>, int
    Output: void

    for n in array A
        if A[n] mod x is equal to 0 then
            print "A at index n is: x";
```

a. Briefly justify the motive(s) behind your design.

> I wrote my algorithm this way because it's very simple. Each element in the array must be iterated over, and so I kept it simple by just looping over everything and checking if it was a multiple of x.

b. What is the Big-O complexity of your solution? Explain clearly how you obtained such complexity.

> The Big-O of my solution is $O(n)$. I loop through every item in my array once every time.

c. What is the Big-Ω complexity of your solution? Explain clearly how you obtained such complexity.

> The Big-Ω of my algorithm is

d. What is the Big-O space complexity of your solution?

> The space complexity of my algorithm is only $O(1)$. It never needs to create new values in the loops, it will only ever output the values that are multiples of x.

Question Removed From Assignment

2. ~~Develop a well-documented pseudo that solves the problem stated in part 1), using either a stack S or a queue Q to perform what is needed.~~

```
Algorithm MultiplesOfX_Stack(S, x)
    Input: Stack<int>, int
    Output: void

    let i = 0;
    let n = S.pop
    while S is not empty loop
        if n mod x is equal to 0
            print "The ith value in the stack is n"
        n = S.pop
        i = i + 1
```

a) ~~Briefly justify the motive(s) behind your design. Similarly to before, you need to iterate over every element in your array.~~
b) ~~What is the Big-O complexity of your solution? Explain clearly how you obtained such complexity.~~
c) ~~What is the Big-Ω complexity of your solution? Explain clearly how you obtained such complexity.~~
d) ~~What is the Big-O space complexity of the utilized stack or queue? Explain your answer.~~

Question 2

Consider the algorithm MySolution below:

```
Algorithm MySolution (A, n)
 Input: Array A of integer containing n elements
 Output: Array B of integer containing n elements
    for i = 0 to n - 1 do
        Res[i] = 0
    end for
    for i = 0 to n - 2 do
        for j = i + 1 to n - 1 do
            if A[i] <= A[j] then
                Res [j] = Res [j] + 1
            else
                Res[i] = Res[i] + 1
            end if
        end for
    end for
    for i = 0 to n - 1 do
        B[Res [i]] = A[i]
    end for

    return B
```

1. What is the big-O $\left(O()\right)$ and big-Omega $\left(\Omega()\right)$ time complexity for algorithm MySolution in terms of n? Show all necessary steps.

   > The code has a for loop, followed by a second for with a nested loop, followed by a third loop. So it'll be:
   > $O(n + n^2 + n) = O(2n + n^2) = O(n^2)$

2. Trace (hand-run) MySolution for an array $A = \{88, 12, 94, 17, 2, 36, 69\}$. What is the resulting B?

```
Res[0] = 0;
Res[1] = 0;
//...
Res[n-1] = 0;
Res == [0,0,0,0,0,0,0]

i = 0, j = 1
    A[0] <= A[1] == false
        Res[0] = Res[0] + 1
Res == [1,0,0,0,0,0,0]

i = 0, j = 2
    A[0] <= A[2] == true
        Res[2] = Res[2] + 1
Res == [1,0,1,0,0,0,0]

i = 0, j = 3
    A[0] <= A[3] == false
        Res[0] = Res[0] + 1
Res == [2,0,1,0,0,0,0]

...

i = 0, j = 6
    A[0] <= A[6] == false
        Res[0] = Res[0] + 1
Res == [5,0,1,0,0,0,0]

i = 1, j = 2
    A[1] <= A[2] == true
        Res[2] = Res[2] + 1
Res == [5,0,2,0,0,0,0]

i = 1, j = 3
    A[1] <= A[3] == true
        Res[3] = Res[3] + 1
Res == [5,0,2,1,0,0,0]

//... Not going to write out every loop through this...

Res = [5,1,6,2,0,3,4]

B = []
q = 0
    B[Res[0]] = A[0] //B[5] = 88
q = 1
    B[Res[1]] = A[1] //B[1] = 12
//...
q = 6
    B[Res[6]] = A[6] //B[4] = 69

B == [2,12,17,36,69,88,94]
```

> $\therefore$ the final value of B is $\{2, 12, 17, 36, 69, 88, 94\}$

3. What does MySolution do? Explain that clearly and briefly given any arbitrary array A of n integers?

> This is a sorting algorithm. It will take an arbitrary array of integers and sort them from smallest to largest.

4. Can the runtime of MySolution be improved easily? Explain how (i.e. re-write another solution(s) that does exactly what MySolution is doing more efficiently)?

> Well, since we can see from the 3rd outer for loop, this pseudo-code language allows us to insert into arrays that don't yet exist at any index. So that means that we could cut out the first for loop entirely, and just use the second for-loop to be inserting into that position in the array. Doing this we'd of course need to have a check to see if the value exists, since you can't add 1 to some undefined value. So you'd check if it's undefined, then set it to one if it is, otherwise increment.
>
> You could also drop the final loop by instead, at the end of the secound outer for loop, inserting into array B.
>
> While the time-complexity of the algorithm will still be $O(n^2)$, the *actual* runtime of the program will be improved by cutting out two loops.
>
> You could of course re-write a sorting algorithm entirely that runs in $O(nlog(n))$, but you can never write one in less than $O(n)$ because you always need to loop through at least once to check if the array is sorted.

```
Algorithm myImplementation(A) {
    Input: Array A of integers.
    Output: Sorted Array of integers.

    Res = [];
    B = [];
    n = A.length;

    for q = 0 to n - 1
        if Res[q] is undefined
            Res[q] = 0;
        end if
        for k = q + 1 to n - 1
            if A[q] <= A[k]
                if Res[k]
                    Res[k] = Res[k] + 1
                else
                    Res[k] = 1
                end if
            else
                if Res[q]
                    Res[q] = Res[q] + 1;
                else
                    Res[q] = 1
                end if
            end if
        end for

        B[Res[q]] = A[q];
    end for

    return B;
```

5. Can the space complexity of MySolution be improved? Explain how?

> This algorithm's space complexity could be further improved. You could make it so that the inner for loop starts back at the beginning of the array so that it checks everything in the array, and increments only a single value instead of that index in an array. This would slightly increase the runtime, but decrease the space complexity.

```
Algorithm myImplementation2(A) {
    Input: Array A of integers.
    Output: Sorted Array of integers.

    Res = 0;
    B = [];

    for q = 0 to A.length - 1
        for k = 0 to A.length - 1
            if A[q] > A[k]
                Res = Res + 1
            end if
        end for

        B[Res] = A[q];
        Res = 0
    end for

    return B;
```

NOTE: Code examples of all of these implementations are available in EcmaScript 6 and Python 2.7 in the assignment folder.

Question 3

For each of the following pairs of functions, either $f(n)$ is $O(g(n))$, $f(n)$ is $\Omega(g(n))$, or $f(n)$ is $\theta(g(n))$. For eachnpair, determine which relationship is correct. Justify your answer.

The symbol $\underline{\underline{\text{H.R}}}$ indicates that Hopital's Rule is being applied to calculate the limit.

i) $f(n) = \log^3 n;\ \ g(n) = \sqrt{n}.$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\log^3 n}{\sqrt{n}} = \frac{\log^3 \infty}{\sqrt{\infty}} = \frac{\infty}{\infty}$$

$$\underline{\underline{\text{H.R}}} \lim_{n\to\infty} \frac{f'(x)}{g'(x)} = \lim_{n\to\infty} -\frac{3\left(\ln\left(x\right)-4\right)\ln\left(x\right)}{\ln^3\left(10\right)x^{\frac{3}{2}}} = -\frac{3\left(\ln\left(\infty\right)-4\right)\ln\left(\infty\right)}{\ln^3\left(10\right)\infty^{\frac{3}{2}}} = \frac{\infty}{\infty}$$

$$\underline{\underline{\text{H.R}}} \lim_{n\to\infty} \frac{f''(x)}{g''(x)} = \lim_{n\to\infty} -\frac{2\left(6\ln\left(x\right)-12\right)}{3\ln^3\left(10\right)x^{\frac{3}{2}}} = -\frac{2\left(6\ln\left(\infty\right)-12\right)}{3\ln^3\left(10\right)\infty^{\frac{3}{2}}} = \frac{\infty}{\infty}$$

$$\underline{\underline{\text{H.R}}} \lim_{n\to\infty} \frac{f'''(x)}{g'''(x)} = \lim_{n\to\infty} -\frac{2}{3\ln^3\left(10\right)x^{\frac{3}{2}}} = -\frac{2}{3\ln^3\left(10\right)\infty^{\frac{3}{2}}} = -\frac{2}{\infty} = 0$$

$\therefore$ Since $\lim_{n\to\infty} \frac{f'''(n)}{g'''(n)} = 0$, this relationship belongs to $f(n) \in \mathrm{O}(g(n))$

ii) $f(n) = n\sqrt{n} + \log n;\ \ g(n) = \log n^2$.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n\sqrt{n} + \log n}{\log n^2} = \frac{\infty\sqrt{\infty} + \log\infty}{\log\infty^2} = \frac{\infty+\infty}{\infty} = \frac{\infty}{\infty}$$

$$\underline{\underline{\text{H.R}}} \lim_{n\to\infty} \frac{f'(x)}{g'(x)} = \lim_{n\to\infty} \frac{\frac{3\sqrt{n}}{2} + \frac{1}{\ln(10)n}}{\frac{2}{\ln(10)n}} = \frac{\frac{3\sqrt{\infty}}{2} + \frac{1}{\ln(10)\infty}}{\frac{2}{\ln(10)\infty}} = \frac{\frac{\infty}{2} + \frac{1}{\infty}}{\frac{2}{\infty}} = \frac{\infty + 0^+}{0^+} = \infty$$

$\therefore$ Since $\lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \infty$, this relationship belongs to $f(n) \in \Omega(g(n))$

iii) $f(n) = n;\ \ g(n) = \log^2 n$.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n}{(\log_{10} n)^2} = \frac{\infty}{(\log_{10}\infty)^2} = \frac{\infty}{\infty}$$

$$\underline{\underline{\text{H.R}}} \lim_{n\to\infty} \frac{f'(x)}{g'(x)} = \lim_{n\to\infty} \frac{1}{\frac{2\ln(n)}{\ln^2(10)n}} = \lim_{n\to\infty} \frac{\ln^2(10)n}{2\ln(n)} = \frac{\ln^2(10)\infty}{2\ln(\infty)} = \frac{\infty}{\infty}$$

$$\underline{\underline{\text{H.R}}} \lim_{n\to\infty} \frac{f''(x)}{g''(x)} = \lim_{n\to\infty} \frac{\ln^2(10)}{\frac{2}{n}} = \frac{\ln^2(10)}{\frac{2}{\infty}} = \frac{C}{0^+} = \infty$$

$\therefore$ Since $\lim_{n\to\infty} \frac{f''(n)}{g''(n)} = \infty$, this relationship belongs to $f(n) \in \Omega(g(n))$

iv) $f(n) = \sqrt{n};\ \ g(n) = 2^{\sqrt{\log n}}$.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\sqrt{n}}{2^{\sqrt{\log n}}} = \frac{\sqrt{\infty}}{2^{\sqrt{\log\infty}}} = \frac{\infty}{2^{\infty}} = \frac{\infty}{\infty}$$

$$\underline{\underline{\text{H.R}}} \lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{\sqrt{\ln(10)}\sqrt{x}\sqrt{\ln(x)}}{\ln(2)\cdot 2^{\frac{\sqrt{\ln(x)}}{\sqrt{\ln(10)}}}} = \frac{\sqrt{\ln(10)}\sqrt{\infty}\sqrt{\ln(\infty)}}{\ln(2)\cdot 2^{\frac{\sqrt{\ln(\infty)}}{\sqrt{\ln(10)}}}} = \frac{C * \infty * \infty}{\infty} = \frac{\infty}{\infty}$$

$$\underline{\underline{\text{H.R}}} \lim_{n\to\infty} \frac{f''(n)}{g''(n)} = \infty$$

At this point the function gets far too large to write the whole thing out in Markdown, but ultimately reduces to ∞.

$\therefore$ Since $\lim_{n\to\infty} \frac{f''(n)}{g''(n)} = \infty$, this relationship belongs to $f(n) \in \Omega(g(n))$

***v)** $f(n) = 2^{n!};\ \ g(n) = 3^n$.

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\lim_{n\to\infty}\frac{2^{n!}}{3^n}=\frac{2^{\infty!}}{3^\infty}=\infty$$

$\therefore$ Since $\lim_{n\to\infty}\frac{f(n)}{g(n)}=\infty$, this relationship belongs to $f(n)\in\Omega(g(n))$

*__vi)__ $f(n)=2^{10n};\ \ g(n)=n^n.$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\lim_{n\to\infty}\frac{2^{10n}}{n^n}=\lim_{n\to\infty}2^{10n}*n^{-n}=0$$

$\therefore$ Since $\lim_{n\to\infty}\frac{f(n)}{g(n)}=0$, this relationship belongs to $f(n)\in\mathrm{O}(g(n))$

*__vii)__ $f(n)=(n^n)^5;\ \ g(n)=n^{(n^5)}.$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\lim_{n\to\infty}\frac{(n^n)^5}{n^{(n^5)}}=\lim_{n\to\infty}(n^n)^5\cdot n^{-(n^5)}=\lim_{n\to\infty}n^{5n-n^5}=0$$

$\therefore$ Since $\lim_{n\to\infty}\frac{f(n)}{g(n)}=0$, this relationship belongs to $f(n)\in\mathrm{O}(g(n))$

*__NOTE:__ For those of these that were much more difficult to compute by hand, an online limit calculator was used. Those that I used WolframAlpha for include the last 3 questions, since neither manipulating the equation, nor using Hopital's rule to calculate derivatives of the functions (when the limit is $\frac{\infty}{\infty}$) provided a decent chance at calculating this. Question 4 was calculated out, but ended up being far too long to input in Markdown. As such, I cut it short.

Programming Questions

In these programming questions you will evaluate two implementations of *List* interface in terms of their performance for different operations.

List[1] interface is an ordered collection (also known as a *sequence*). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

The List interface provides four methods for index **access** to list elements, two methods to **search** for a specific object, and two methods to efficiently **insert** and **remove** multiple elements at an arbitrary location in the list. Note that the speed of these operations may depend on the implementation (e.g. Array or LinkedList).

You are required to write two implementations of `List` interface, one that uses array, and one that uses doubly-linked list. Then, you will have to test the performance of several operations when using your implementations.

Question 1

Implement the following methods in the two implementations (called `MyArrayList` and `MyLinkedList`) of `List` interface:

```
Boolean add(E e);               // Appends the specified element to the end of this list
void add(int index,E element);  // Inserts the specified element at the specified position in this list
void clear();                   // Removes all of the elements from this list
E remove(int index);            // Removes the element at the specified position in this list
Boolean remove(Object o);       // Removes the first occurrence of the specified element from this list
String toString();              // Returns a string representation of this list
int size();                     // Returns the number of elements in this list
```

Define your classes to be generics. The array implementation should have dynamic resizing (double the size when growing and halve the size when less than 25 % of the capacity is used); and the linked list implementation should use doubly linked list. Also, the behavior of these methods should be equivalent to that of Java Standard Library's classes `ArrayList` or `LinkedList`. Please refer to the corresponding descriptions online[2/3].

For the rest of the methods of the List interface, you may just throw an exception:

```
public type someUnneededMethod() {
    throw new UnsupportedOperationException();
}
```

Question 2

Write your driver class `ListTester`. Use both of your list implementations and compare them to the corresponding Java library implementations (`ArrayList` and `LinkedList`) For numbers $N = \{10, 100, 1000, 10000, 100000, 1000000\}$

1. Starting with *empty* lists of Number-s; measure how long it takes to insert N integer numbers (`int`, or `Integer`) with random values ranging from 0 to 2N into the lists, when inserting them at the *beginning*, at the *end*, and into a *random location* of the list (use indices to indicate where to do the insertion (e.g., `list.add (randomLocation, number)`).
2. Starting with non-empty lists of N items (e.g., from part a), measure how long it takes to remove N numbers from the lists when removing them from the beginning, from the end, and from a random location of the list (use indices to indicate the location).
3. Starting with non-empty lists of N items (same as part b), measure how long it takes to remove N random numbers (with values between 0 and 2N) from the four lists (some values might not exist in the list!).

- Produce the following table (the timing values below are just an illustration and do not relate to any real measurements):

N = 10	Insert@start(ms)	Insert@end (ms)	Insert@random (ms)
MyArrayList	15	201	603
ArrayList	15	201	603
MyLinkedList	15	201	603

N = 10	Insert@start(ms)	Insert@end (ms)	Insert@random (ms)
LinkedList	15	201	603

N = 10	Remove@start(ms)	Remove@end(ms)	Remove@random(ms)	Remove byvalue (ms)
MyArrayList	15	201	603	121212
ArrayList	15	201	603	121212
MyLinkedList	15	201	603	121212
LinkedList	15	201	603	121212

- Repeat for all values of $N = 100$; $N = 1000$; ...etc.

Save the result of your program execution in a file `testrun.txt` and submit it together with your other files.

Important Requirements:

1. Make sure you reset the timer (or save the intermediate time before the next measurement); i.e., make sure your measured time contains only the time to perform one set of operations that was supposed to be timed.
2. In case the operations for big N numbers take too long (e.g., more than 50s) you may reduce the number to a smaller one or eliminate it (so that you will have a range from, say, 1 to 100000).
3. Do not use any java abstract data type or packages when writing `MyArraylist` and `MyLinkedList` in these programming questions of your assignment.
4. Your code should handle boundary cases and error conditions. It is also imperative that you test your classes.
5. For these programming questions, you are required to submit the commented Java source files, the compiled files (.class files), and the test run text files.