Assignment 1
Due: June 1, 2023 at 11:59PM

1 Assignment 1 Specification

1.1 Implementation

The skeleton code at `https://classroom.github.com/a/ebe1_XNQ` is the start of this week's assignment. Follow the setup and import directions to access the code. Write and test the following functions. Use interactive mode and run the provided unit test cases to check your solutions. Feel free to write more tests to confirm that edge cases behave the way you expect.

For this and future F# assignments, you should use **no library calls** besides `.Head` and `.Tail`. There are some libraries that have our desired functions already, but you should practice implementing functions from simple language constructs.

1.1.1 Binomial coefficient

In mathematics, the binomial coefficient $C(n, k)$ is the number of ways of picking k unordered outcomes from n possibilities. It is given by the formula:

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

Write a version that calls your factorial function from Lab 1. Copy the factorial function definition to your Assignment 1 project in order to use it.

```
> C 20 5;;
val it : System.Numerics.BigInteger = 15504 {...}
```

State a runtime complexity (e.g. $O(1), O(n), O(n^2)$, etc.) using the `CComplexity` function. Pick from among the possible `Complexity` type values (defined at the top of the code) to replace `NoAns`. (The `Complexity` type is an F# version of an enum.)

1.1.2 Biggest

Biggest returns the largest `int` from a given list of ints. Return -1 if the list is empty.

1.1.3 Positive

Positive returns a list consisting of the positive ints in a given list of ints.

1.1.4 Intersect

Intersect returns the intersection of two lists. Assume that neither input list has duplicates and that the lists are the same type. Hint: Use isMember as a helper function:

```
> let rec isMember a L =
    match L with
      | [] -> false
      | h::t -> if a=h then true else isMember a t

> isMember "ab" ["t";"k";"ab"];;
val it : bool = true
```

1.1.5 Insert

Insert takes an `int` value and `int list` and returns an ordered int list with the int element inserted in the proper place. Assume the int list argument is in order already.

State a time complexity for insert (e.g. $O(1), O(n), O(n^2)$, etc.) in the `insertComplexity` function.

1.1.6 Sort

Implement selection sort on an int list. Hint: Use insert repeatedly.

Just for comparison, below is a Java version of the insertion sort.

```
static LinkedList insert(Integer a, LinkedList L) {
  if (L.size() == 0) L.add(a);
  else if (a.intValue() > ((Integer) L.getFirst()).intValue()) {
    Integer first = (Integer) L.removeFirst();
    insert(a, L).addFirst(first);
  }
  else L.addFirst(a);
  return L;
}

static LinkedList sort(LinkedList L) {
  if (L.size() == 0) return L;
  Integer first = (Integer) L.removeFirst();
  return insert(first, sort(L));
}
```

State a time complexity (e.g. $O(1), O(n), O(n^2)$, etc.) using the `sortComplexity` function.

1.1.7 Vector add

`vecadd` adds two integer lists, element by element. Assume the two int lists contain the same number of elements.

```
> vecadd [1;2;3] [4;5;6];;
val it : int list = [5;7;9]

> vecadd [1; 2; -3; 4] [4; -5; 6; 7];;
val it : int list = [5; -3; 3; 11]
```

1.1.8 Matrix add

Add two integer matrices using `vecadd` function from the previous question. Assume the two matrices have the same dimensions.

```
> matadd [ [1;2]; [3;4] ] [ [5;6]; [7;8] ];;
val it = int list list = [ [6;8]; [10;12] ]

> matadd [ [1;2]; [3;4]; [5;6] ] [ [1;2]; [3;4]; [5;6] ];;
val it : int list list = [ [2;4] [6;8]; [10;12] ]
```

2 Testing

Unit testing is supplied for this assignment. These are not exhaustive tests and you should try running the functions in interactive mode with other valid arguments as well.

All of your function signatures (the parameter and return type) must be correct in order for the tests to run. Do not include any ';;' statement endings in your source code.

Mac directions:

1. Go to **View→Tests** and a panel should open
2. Click the 'Run all tests' button and you should get unit-by-unit feedback on test results.

Windows directions:

1. Go to the **Solution explorer** window.
2. Right-click on the top line in the window: **Solution 'comp3350_a1'** file→**Build Solution**.
3. Go to **Test→Run All Tests**.
4. The test names should appear in the **Test Explorer** window, and you can run them by clicking **Run All** in the **Test Explorer** window.

3 Grading

Be professional. Make results easy to understand and grade. Include only those parts to be graded. Leave comments where necessary, especially if it aids in grading.

Each of the 8 functions is worth $\frac{1}{8}$ of the assignment grade.