Peter Parianos

Project 3: CIS424



**Description of the Code:**

**Problem 1**: The ML code uses two built in functions call *hd*() and *tl*(), where *hd* returns the first element of a list, and tl returns the the list without the head. If you use *tl* then *hd*, this returns the second elements, given a list >= 2.

**Problem 2**: nothird returns a list without the third element, given that the list is >= 3. If you use the const function twice on the tail of the rest of the list, the third element is deleted.

**Problem 3**: This is probably the most difficult question. I use a swap-sort algorithm and recursively call the *largesmall*() method until the parameters are sorted from least to greatest. At the most, three comparisons would be made recursively. The return value is a 2-tuple containing the largest and smallest parameters, in that order. This is basically bubble sort.

**Problem 4**: Since there will be no bounds to the size of the list to be input into this program, we need use test cases where a null list or a singleton list is entered. This is because if the list >1, then we will just make one swap. We will define flipL three times. flipL wil we return the singleton if the rest of the list is nil when const.

**Problem 5(ML and Python):** There is a built in map function that comes with ML, but I defined it again in order to gain more knowledge on how it works. In Map, function *f* is applied to each element of the list. We needed to use an anonymous function (no name) to modify each of the elements of the list. We use fn to take a strictly real number, x, and then change it to 0.0 if < 0.0. This was put directly into the Map() params. negLreal is list of reals.

In the python example, I used the built in function Map. You need to use lambda instead of fn. Also it return an iterator which will need to be converted to a list before a print to console.

**Problem 6 (ML and Python):** This is the same case where I could have used a built in function but chose not to do so, because an example in the class was given. We throw an error if a nul list is entered, we return element a if it is a singleton list, and we use function *f* using the first element and then the second element, which will do an operation two bot, then reduce. We use an anonymous function where we return the smallest element of the list.

In the python example, reduce is located in the functools library, and I used a lambda function instead.

**Problem 7 (ML and Python):** I decided to use the built-in function which comes with ML in the library called List. The built in function uses currying, which uses a space as a delimiter instead of a comma. You need to place the anonymous function in parentheses for this case. I used the operator andalso in order to bound param x to be 2.0 <= x <= 3.0. I used type inference to make sure the param is a real. Filter returns a list where the function return true on each of the elements of the list.

Debugging and issues:

Problem 3: My first idea to create this function failed to succeed with all of the unit tests. Then I used recursion instead. I unfortunately did not take screenshots. Now it passes all of the tests.