

Project 4

Krishaun Bartlet 2696351

Ryan Reynolds 2693018

Login: ryreynol

**Code Description:**

SML:

Problem 1 takes in any list and returns the second element of the list. It does this by calling splitting the list with x::xr x representing the first element of the list and xr representing the remaining elements of the list. Then calling the head function hd to return the 2nd element of the list. An exeception ListTooShort was declared and thrown if the list is empty or only contains 2 elements. The exceptions were raised to remove the sml warning, even though the question assumes the list is at least 2 elements long.

Problem 2 takes in any list and removes the third element from the list. We chain the cons function to isolate the 1st, 2nd , 3rd, and remaining elements when the function is called. Then appending the 1st, 2nd, and remaining elements effectively removing the third element from the list. We use the same exception used in problem one. The exception is raised if the list is empty or the list is shorter than three elements. The exceptions were raised to remove the sml warning, even though the question assumes the list is at least 2 elements long.

Problem 3 takes in 3 integers and returns the max and min of the three using 3 comparisons. In order to achieve the 6 possible combinations with 3 comparisons, recursive calls are used to swap the positions of each element until the max is in position b and the min is in position c.

Problem 4 flips alternating elements in a list. If the list is empty nil is returned. If the list contains a single element that element is returned (this is used to handle odd lists). Else the function uses cons chaining to get the first two elements of the list and the tail of the list. The first element is then swapped with the second element by appending the 1st element to the 2nd element. Then the alternate function is called and appended to the 1st and 2nd element to iterate through the entire list.

Problem 5 uses the map function to apply an anonymous function to each element of the list. The map function takes in a function and a list. Map then calls the function on the first element calls cons then recursively calls the map function to continue the iteration through the list. The anonymous function checks if the element x is positive if it’s not it is changed in the list to 0.0 else the value is unchanged.

Problem 6 defines a reduce function used with an anonymous function to return the min in the list. Reduce takes in a function and a list then returns the result of the application of the function over all of the elements of the list. In this case we apply a simple min function. The anonymous min function reduces the list size until there is only one element remaining. The first two elements are chosen then compared and the min remains as the head comparison. Then the function is applied to the remaining list until the result is the only element left in the lsit.

Problem 7 defines a filter function to iterate through a list and apply a Boolean and return the elements within the list that meet the Boolean condition. In our case the elements in the list are returned if they are between 2.0 and 3.0. Our filter function checks the result of the function for the element x if it is true it is added to the list and the next iteration of the function is called. If the function is false then the next iteration is called and the previous element is not added to the list.

Python:

The project also required us to implement the map, reduce and filter functions in python using

anonymous functions to carry out the specified task. The anonymous function was

implemented using the lambda function or lambda operator which is a simple way to create a

small functions without a function name. The general implementation of a lambda function is

lambda arg\_list : expression. If there are more than one argument in the arg\_list they are

separated by a comma. The lambda operator is often used with the map(), reduce() and filter()

functions.

How the map function handles this is that it takes both the anonymous function and the list as parameters and applies the function to all the elements of the list. It then returns a new list with the elements changed by what the function does. In this case replace negative real

numbers with 0.0.

Both the reduce and filter function, just like the map function takes a function and a list as parameters, but unlike the map function the reduce function as the name suggests is a very

useful function. When implemented, the function passed is continuously applied to all the

values of the list, to be more specific, the function is called with the first two elements of the

list and the function is applied to them then the function will then be applied to the result of

the previous function implementation and third element of the list until just one element

remains and then that element is returned as the result of the reduce function. The reduce

function is defined in functools for python 3, so the code from functools import reduce must be called prior to execution.

**Debugging:**

SML:

We had some minor debugging with problems 1 and 2. These were mostly syntax errors as we had forgotten the proper syntax from lecture for the cons operator.

Majority of debugging occurred for problems 3, 4, and 6. For problem 3 we had significant logic errors when reducing our comparisons down to a max of 3. Initially we were not providing the correct position swaps and reported the middle and the max instead of max, min. For problem 4 we encountered both syntax and logic errors. The syntax errors came because we initially attempted to pass hd(x)::hd(tl(x))::tl(x) to our function instead of simply passing the remaining list xr. We also encountered an error when trying to handle a list with an odd number of elements. To combat this case we included a separate alternate call to be used when there is only 1 element left in the list. Problem 6 we encountered logic errors when defining the anonymous function. We were returning the max instead of the min.

Additionally, We tested each function individually and encountered some minor syntax errors when combing them into a single file (typically accidentally omitting a ; at the end of a function).

Python:

Very little debugging was required for the anonymous functions because we had already implemented them in sml and were easily able to transfer the anonymous functions logic into python. Similarly there was not much debugging for the reduce, map, and filter functions because python defines them already. The only issue we ran into was running the reduce function without calling functools.reduce(). To solve this issue we directly imported the reduce function so that it did not need to be called as functools.reduce().

**Testing:**

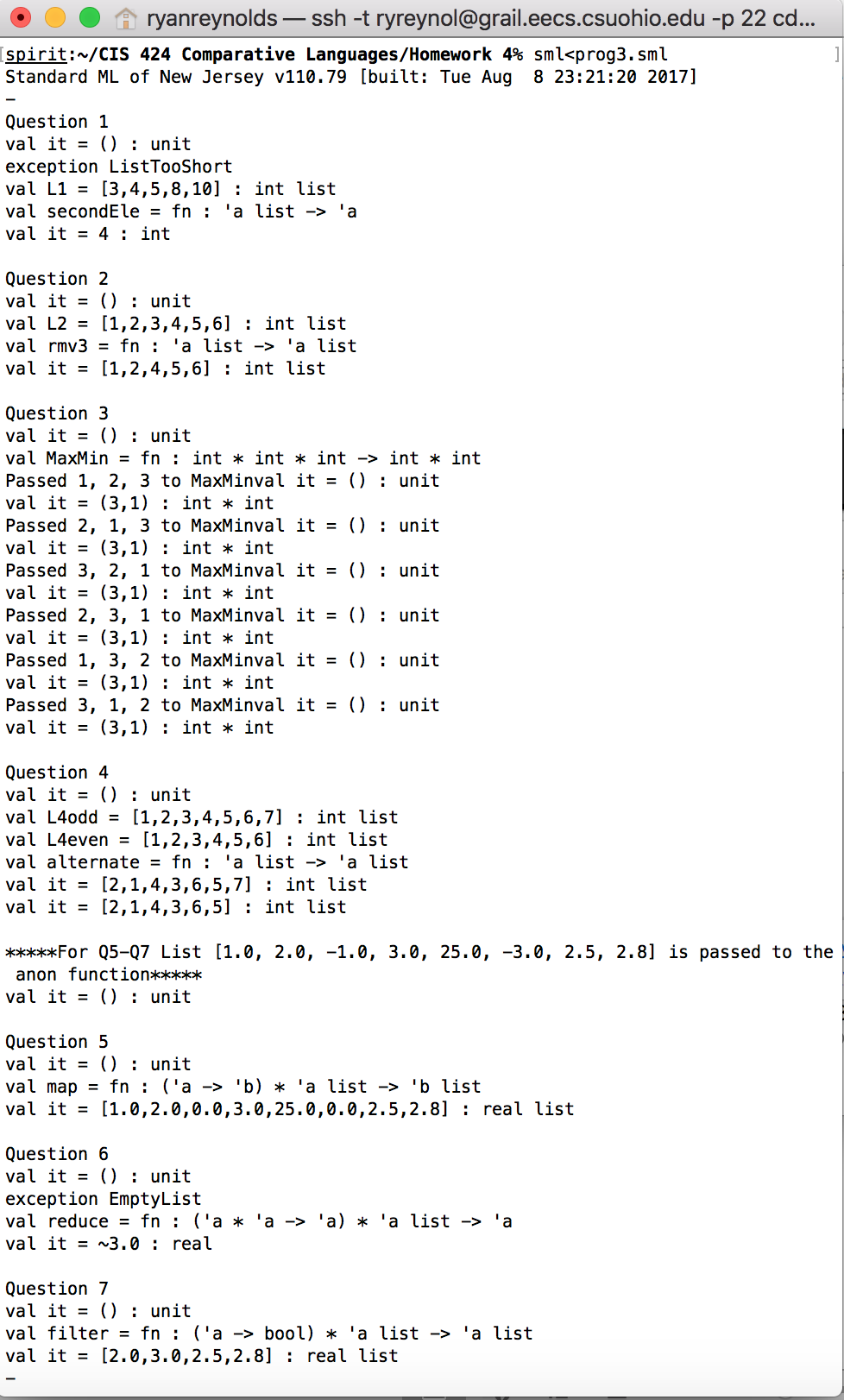
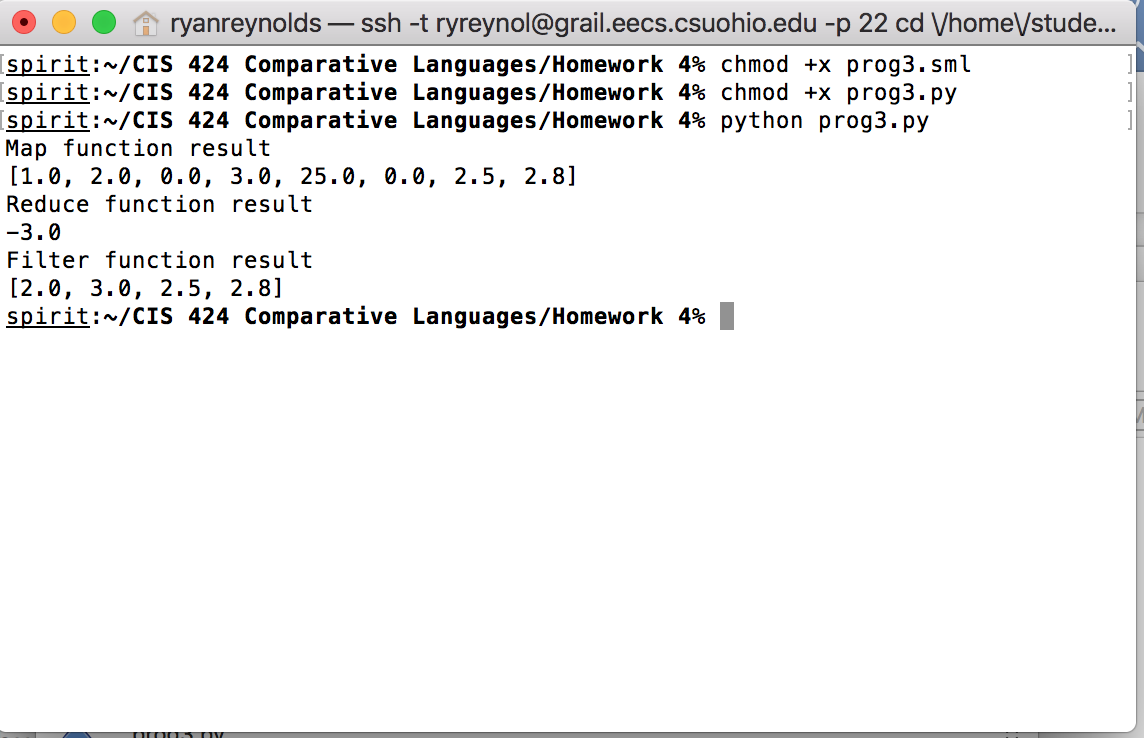
Each problem is given a separate test list or parameters to test all the conditions required in project prompt. The testing of the program in CSU’s linux environment and proof of submission is provided below.

Fig 1: The above screenshot shows the sml test cases being run on spirit

Fig 2: Shows the python file being run on spirit. Note it is not possible to run the python program as an executable with the shebang line on spirit for this project. We provided the shebang line for the python program and modified both the sml and python files with executable permissions just in case.

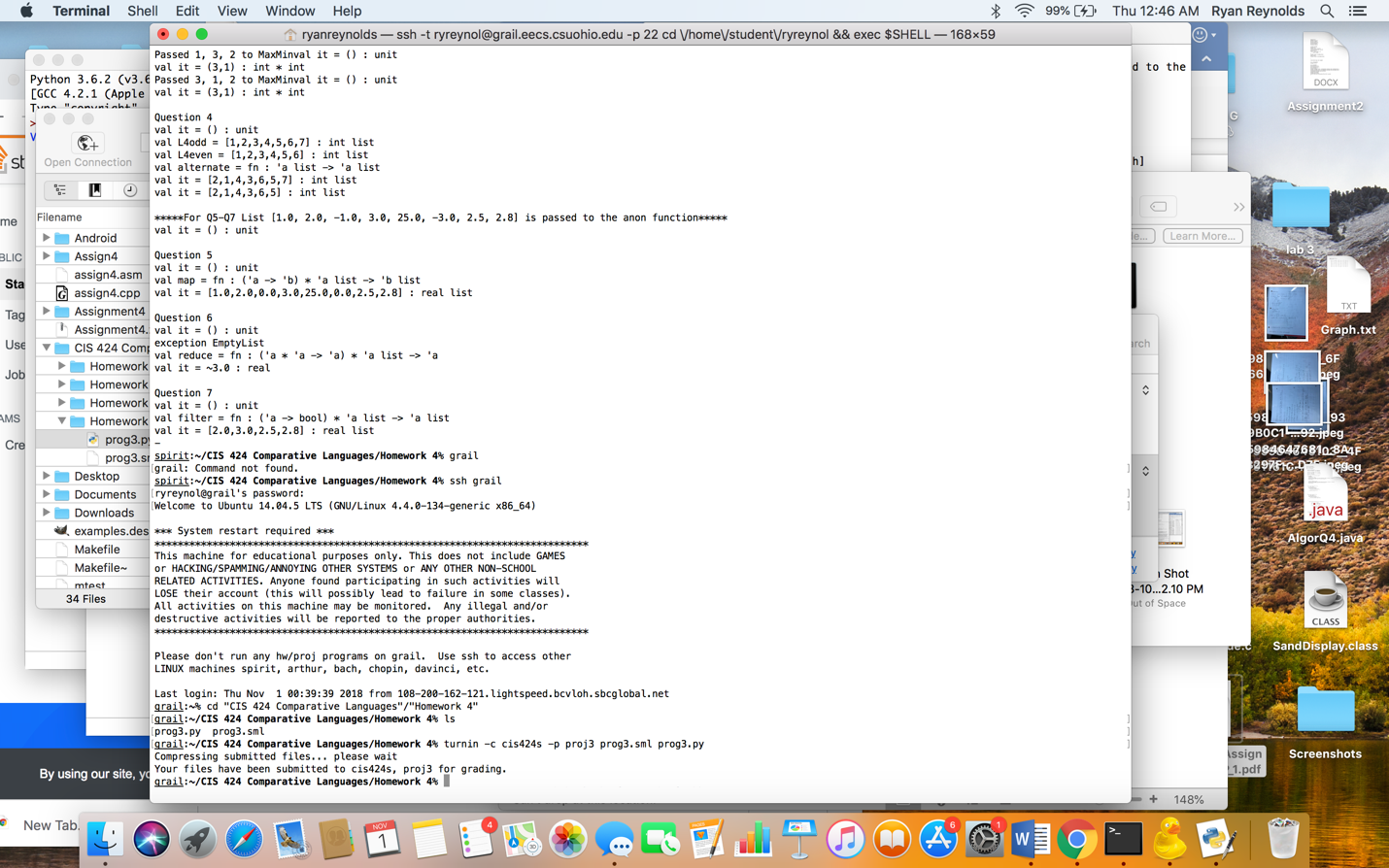


Fig 3: Shows proof of submission of both the successfully run programs.