Project 1 Report

Kyle Peppe

January 20, 2020

Abstract

This project built on our initial learning of the ML programming language and some more complicated problems that we had to solve. This project finished up with us performing some basic HOL instructions. Then I put the findings, code and output into Latex to further enhance my knowledge on making these reports. I did the exercises from the PDF book from the class 4-6-3, 4-6-4, 5.3.4, 5.3.5, and 6.2.1. Below are the sections that are in this report for each problem (some sections have Execution Transcripts too:

- Problem statement
- Relevant code
- Test results
- Execution Transcripts

For each problem or exercise-oriented chapter in the main body of the report is a corresponding chapter in the Appendix containing the source code in ML. This source code is not pasted into the Appendix. Rather, it is input directly from the source code file itself.

Acknowledgments: I would like to acknowledge the 2 professors, Professor Chin and Professor Hamner, that have helped me begin to understand this new ML programming language. Also to Syracuse University for accepting me to this Masters program in Cybersecurity.

Contents

1 Executive Summary **4**

2 Exercise 4.6.3 **5**
2.1 Problem Statement for Exercise 4.6.3 5
2.2 Relevant Code for Exercise 4.6.3 5
2.3 Test cases for Exercise 4.6.3 5
2.4 Execution Transcripts for Exercise 4.6.3 5

3 Exercise 4.6.4 **8**
3.1 Problem Statement for Exercise 4.6.4 8
3.2 Relevant Code for Exercise 4.6.4 8
3.3 Test cases for Exercise 4.6.4 8
3.4 Execution Transcripts for Exercise 4.6.4 8

4 Exercise 5.3.4 **9**
4.1 Problem Statement for Exercise 5.3.4 9
4.2 Relevant Code for Exercise 5.3.4 9
4.3 Test cases for Exercise 5.3.4 9
4.4 Execution Transcripts for Exercise 5.3.4 9

5 Exercise 5.3.5 **10**
5.1 Problem Statement for Exercise 5.3.5 10
5.2 Relevant Code for Exercise 5.3.5 10
5.3 Test cases for Exercise 5.3.5 10
5.4 Execution Transcripts for Exercise 5.3.5 10

6 Exercise 6.2.1 **11**
6.1 Problem Statement for Exercise 6.2.1 11
6.2 Relevant Code for Exercise 6.2.1 11
6.3 Test cases for Exercise 6.2.1 11

7 Relevant Information **12**
7.1 Specific Test Cases 12
7.1.1 Test Cases for Exercise 4.6.3 12
7.1.2 Test Cases for Exercise 4.6.4 12
7.1.3 Test Cases for Exercise 5.3.4 12
7.1.4 Test Cases for Exercise 5.3.5 12
7.1.5 Test Cases for Exercise 6.2.1 12

A Source Code for Exercise 4.6.3 **13**

B Source Code for Exercise 4.6.4 **14**

C Source Code for Exercise 5.3.4 **15**

D Source Code for Exercise 5.3.5 **16**

E Source Code for Exercise 6.2.1 **17**

Chapter 1

Executive Summary

All requirements for this project are satisfied. Specifically,

Report Contents
The report has the following content:

Chapter 1: Executive Summary

Chapter 2: Exercise 4.6.3

- Section 2.1: Problem statement
- Section 2.2: Relevant code
- Section 2.3: Test results
- Section 2.4: Execution Transcripts

Chapter 3: Exercise 4.6.4

- Section 3.1: Problem statement
- Section 3.2: Relevant code
- Section 3.3: Test results
- Section 3.4: Execution Transcripts

Chapter 4: Exercise 5.3.4

- Section 4.1: Problem statement
- Section 4.2: Relevant code
- Section 4.3: Test results
- Section 4.4: Execution Transcripts

Chapter 5: Exercise 5.3.5

- Section 5.1: Problem statement
- Section 5.2: Relevant code
- Section 5.3: Test results
- Section 5.4: Execution Transcripts

Chapter 6: Exercise 6.2.1

- Section 6.1: Problem statement
- Section 6.2: Relevant code
- Section 6.3: Test results

Chapter A: Source Code for Exercise 4.6.3

Chapter B: Source Code for Exercise 4.6.4

Chapter C: Source Code for Exercise 5.3.4

Chapter D: Source Code for Exercise 5.3.5

Chapter E: Source Code for Exercise 6.2.1

Reproducibility in ML and LaTeX
Our ML and LaTeX source files compile with no errors.

Chapter 2

Exercise 4.6.3

2.1 Problem Statement for Exercise 4.6.3

For this problem I solved the Exercise 4.6.3 from the PDF book. This problem was to create fucntions for the 5 probelms from Exercise 4.6.2. I had to create the functions using val and then using fun.

2.2 Relevant Code for Exercise 4.6.3

The relevant code wil be in the corresponding Appendix at the end of the report.

2.3 Test cases for Exercise 4.6.3

Below are the test cases for this exercise that were requested based of the provided handout. I did change variable names where needed.

Part A: Test List A = [(1,2,3), (4,5,6), (7,8,9)] Part B: Test List B = [(0,0), (1,2), (4,3)] Part C: Test List C = [(Hi, there!), (Oh , no!), (What, the)] Part D: Test List D1 = [([0,1], [2,3,4]), ([], [0,1])] Part D: Test List D2 = [([true, true], [])] Part E: Test List E = [(2,1), (5,5), (5,10)]

2.4 Execution Transcripts for Exercise 4.6.3

Below are the results from running the test cases:

The following is output from *ex-4-6-3.sml*

```
        HOL-4 [Kananaskis 11 (stdknl, built Sat Aug 19 09:30:06 2017)]

        For introductory HOL help, type: help "hol";
        To exit type <Control>-D
```

```
> > > > val funA1 = (fn x => (fn y => (fn z => x + y + x)));
val funA1 = fn: int -> int -> 'a -> int
> val funA1 = (fn x => (fn y => (fn z => x + y + x)1)2)3;
val funA1 = 8: int
> val funA1 = (fn x => (fn y => (fn z => x + y + x)4)5)6;
val funA1 = 17: int
> val funA1 = (fn x => (fn y => (fn z => x + y + x)7)8)9;
val funA1 = 26: int
> fun funA2 (x,y,z) = x + y + z;
val funA2 = fn: int * int * int -> int
> funA2 (1,2,3);
val it = 6: int
> funA2 (4,5,6);
val it = 15: int
```

```
> funA3 (7,8,9);
poly: : error: Value or constructor (funA3) has not been declared Found near funA3 (7, 8, 9
Static Errors
> funA2 (7,8,9);
val it = 24: int
> val funB1 = (fn x => (fn y => x < y));
val funB1 = fn: int -> int -> bool
> val funB1 = (fn x => (fn y => x < y)0)0;
val funB1 = false: bool
> val funB1 = (fn x => (fn y => x < y)1)2;
val funB1 = false: bool
> val funB1 = (fn x => (fn y => x < y)4)3;
val funB1 = true: bool
> fun funB2 (x,y) = x < y;
val funB2 = fn: int * int -> bool
> funB2 (0,0);
val it = false: bool
> funB2 (1,2);
val it = true: bool
> funB2 (4,3);
val it = false: bool
> val funC1 = (fn s1 => (fn s2 => s1 ^ s2));
val funC1 = fn: string -> string -> string
> val funC1 = (fn s1 => (fn s2 => s1 ^ s2)"hi")"there!";
val funC1 = "there!hi": string
> val funC1 = (fn s1 => (fn s2 => s1 ^ s2)"Oh")"no!";
val funC1 = "no!Oh": string
> val funC1 = (fn s1 => (fn s2 => s1 ^ s2)"What")"the...";
val funC1 = "the...What": string
> fun funC2 (s1, s2) = s1 ^ s2;
val funC2 = fn: string * string -> string
> funC2 ("Hi","there!");
val it = "Hithere!": string
> funC2 ("Oh", "no!");
val it = "Ohno!": string
> func2 ("What", "the...");
poly: : error: Value or constructor (func2) has not been declared
Found near func2 ("What", "the...")
Static Errors
> funC2 ("What", "the..");
val it = "Whatthe..": string
> val funD1 = (fn list1 => (fn list2 => list1 @ list2));
val funD1 = fn: 'a list -> 'a list -> 'a list
> val funD1 = (fn list1 => (fn list2 => list1 @ list2)[0,1])[2,3,4];
val funD1 = [2, 3, 4, 0, 1]: int list
> val funD1 = (fn list1 => (fn list2 => list1 @ list2)[])[0,1];
val funD1 = [0, 1]: int list
> fun funD2 (list1, list2) = list1 @ list2;
val funD2 = fn: 'a list * 'a list -> 'a list
> funD2 ([0,1],[2,3,4]);
val it = [0, 1, 2, 3, 4]: int list
> funD2 ([],[0,1]);
val it = [0, 1]: int list
```

```
> val funE1 = (fn x => (fn y => if x > y then x else y));
val funE1 = fn: int -> int -> int
> val funE1 = (fn x => (fn y => if x > y then x else y)2)1;
val funE1 = 2: int
> val funE1 = (fn x => (fn y => if x > y then x else y)5)5;
val funE1 = 5: int
> val funE1 = (fn x => (fn y => if x > y then x else y)5)10;
val funE1 = 10: int
> fun funE2 (x,y) = if x > y then x else y;
val funE2 = fn: int * int -> int
> funE2 (2,1);
val it = 2: int
> funE2 (5,5);
val it = 5: int
> funE2 (5,10);
val it = 10: int
>
```

Chapter 3

Exercise 4.6.4

3.1 Problem Statement for Exercise 4.6.4

For this problem I solved the Exercise 4.6.4 from the PDF book. This problem was to creat a funtion called listSquares that would take in a list of values and return the square of each value in the list.

3.2 Relevant Code for Exercise 4.6.4

The relevant code wil be in the corresponding Appendix at the end of the report.

3.3 Test cases for Exercise 4.6.4

Below are the test case that was used and provided by the professor.
Test List = [1,2,3,4,5]

3.4 Execution Transcripts for Exercise 4.6.4

Below are the results from running the test cases:
The following is output from *ex-4-6-4.sml*

```
        HOL-4 [Kananaskis 11 (stdknl, built Sat Aug 19 09:30:06 2017)]

        For introductory HOL help, type: help "hol";
        To exit type <Control>-D
```

```
> > > > fun listSquares intList =
let
        fun square x = x * x;
in
        (map square intList)
end;
# # # # # val listSquares = fn: int list -> int list
> val testList = [1,2,3,4,5];
val testList = [1, 2, 3, 4, 5]: int list
> val testResults = listSquares testList;
val testResults = [1, 4, 9, 16, 25]: int list
>
```

Chapter 4

Exercise 5.3.4

4.1 Problem Statement for Exercise 5.3.4

For this problem I solved the Exercise 5.3.4 from the PDF book. This problem was to set up a filter using the rules from the previous Exercise 5.3.3.

4.2 Relevant Code for Exercise 5.3.4

The relevant code wil be in the corresponding Appendix at the end of the report.

4.3 Test cases for Exercise 5.3.4

Below are the test case that was used and provided by the professor.
testResults = Filter(fn x =¿ x ¡5) [1,2,3,4,5,6,7,8,9]

4.4 Execution Transcripts for Exercise 5.3.4

Below are the results from running the test cases:
The following is output from *ex-5-3-4.sml*

```
        HOL-4 [Kananaskis 11 (stdknl, built Sat Aug 19 09:30:06 2017)]

        For introductory HOL help, type: help "hol";
        To exit type <Control>-D
```

```
> > > > filter (fn x => x < 5);
val it = fn: int list -> int list
> val testResults = filter (fn x => x < 5) [1,2,3,4,5,6,7,8,9];
val testResults = [1, 2, 3, 4]: int list
>
Process HOL killed
```

Chapter 5

Exercise 5.3.5

5.1 Problem Statement for Exercise 5.3.5

For this problem I solved the Exercise 5.3.5 from the PDF book. The problem was to take in a list of integers and an integer and then based off that integer return the list with only values that are greater than the solo integer.

5.2 Relevant Code for Exercise 5.3.5

The relevant code wil be in the corresponding Appendix at the end of the report.

5.3 Test cases for Exercise 5.3.5

Below are the test case that was used and provided by the professor.
addPairsGreaterThan 0 [(0,1), (2,0), (2,3), (4,5)];

5.4 Execution Transcripts for Exercise 5.3.5

Below are the results from running the test cases:
The following is output from *ex-5-3-5.sml*

```
        HOL-4 [Kananaskis 11 (stdknl, built Sat Aug 19 09:30:06 2017)]

        For introductory HOL help, type: help "hol";
        To exit type <Control>-D
```

```
> > > > fun addpair (x,y) = x + y;
val addpair = fn: int * int -> int
> fun Filter n xlist = filter(fn (x,y) => y>n andalso x>n) xlist;
val Filter = fn: int -> (int * int) list -> (int * int) list
> fun addlist xlist = map addpair xlist;
val addlist = fn: (int * int) list -> int list
> fun addPairsGreaterThan y xlist = (addlist (Filter y xlist));
val addPairsGreaterThan = fn: int -> (int * int) list -> int list
> addPairsGreaterThan 0 [(0,1),(2,0),(2,3),(4,5)];
val it = [5, 9]: int list
>
Process HOL killed
```

Chapter 6

Exercise 6.2.1

6.1 Problem Statement for Exercise 6.2.1

For this problem I solved the Exercise 6.2.1 from the PDF book. This problem was to solve 7 different HOL based problems. I made sure to show types and disable Unicode so the printing of the output was as expected/wanted.

6.2 Relevant Code for Exercise 6.2.1

The relevant code wil be in the corresponding Appendix at the end of the report.

6.3 Test cases for Exercise 6.2.1

In this section we did not necessarily run any tests we just got the output from running the HOL commands.

Chapter 7

Relevant Information

7.1 Specific Test Cases

7.1.1 Test Cases for Exercise 4.6.3

Below are the test cases for this exercise that were requested based of the provided handout. I did change variable names where needed.

Part A: Test List A = [(1,2,3), (4,5,6), (7,8,9)] Part B: Test List B = [(0,0), (1,2), (4,3)] Part C: Test List C = [(Hi, there!), (Oh , no!), (What, the)] Part D: Test List D1 = [([0,1], [2,3,4]), ([], [0,1])] Part D: Test List D2 = [([true, true], [])] Part E: Test List E = [(2,1), (5,5), (5,10)]

7.1.2 Test Cases for Exercise 4.6.4

Below are the test case that was used and provided by the professor.

Test List = [1,2,3,4,5]

7.1.3 Test Cases for Exercise 5.3.4

Below are the test case that was used and provided by the professor.

testResults = Filter(fn x =¿ x ¡5) [1,2,3,4,5,6,7,8,9]

7.1.4 Test Cases for Exercise 5.3.5

Below are the test case that was used and provided by the professor.

addPairsGreaterThan 0 [(0,1), (2,0), (2,3), (4,5)];

7.1.5 Test Cases for Exercise 6.2.1

In this section we did not necessarily run any tests we just got the output from running the HOL commands.

Appendix A

Source Code for Exercise 4.6.3

The following code is from *Ex-4-6-3.sml*

```
(*************************************************************************)
(* Authoer: Kyle Peppe                                                   *)
(* Exercise 4.6.3                                                        *)
(* Date: 1/14/20                                                         *)
(*************************************************************************)

(* Part A                       *)
(* Using val and fn             *)
val funA1 = (fn x => (fn y => (fn z => x + y + x)));

(* Using fun                    *)
fun funA2 (x,y,z) = x + y + z;

(* Part B                       *)
(* Using val and fn             *)
val funB1 = (fn x => (fn y => x < y));

(* Using fun                    *)
fun funB2 (x,y) = x < y;

(* Part C                       *)
(* Using val and fn             *)
val funC1 = (fn s1 => (fn s2 => s1 ^ s2));

(* Using fun                    *)
fun funC2 (s1, s2) = s1 ^ s2;

(* Part D                       *)
(* Using val and fn             *)
val funD1 = (fn list1 => (fn list2 => list1 @ list2));

(* Using fun                    *)
fun funD2 (list1, list2) = list1 @ list2;

(* Part E                       *)
(Using val and fn               *)
val funE1 = (fn x => (fn y => if x > y then x else y));

(* Using fun                    *)
fun funE2 (x,y) = if x > y then x else y;
```

Appendix B

Source Code for Exercise 4.6.4

The following code is from *Ex-4-6-4.sml*

```
(*****************************************************************************)
(* Author: Kyle Peppe                                                        *)
(* Exercise 4.6.4                                                            *)
(* Date: 1/16/20                                                             *)
(*****************************************************************************)

(* Create a function that takes a list and find the square value of it       *)
fun listSquares intList =
let
        fun square x = x * x;
in
        (map square intList)
end;

(* Creating the test list from the handout                                   *)
val testList = [1,2,3,4,5];

(* Creating a new list that will contain the result of the above testList    *)
val testResults = listSquares testList;
```

Appendix C

Source Code for Exercise 5.3.4

The following code is from *Ex-5-3-4.sml*

```
(*************************************************************************)
(* Author: Kyle Peppe                                                    *)
(* Exercise 5.3.4                                                        *)
(* Date: 1/18/20                                                         *)
(*************************************************************************)

(* Setting up the Filter                                                 *)
filter (fn x => x < 5);

(* Creating variable where the results from Test will be returned        *)
(* Returning the results from the filter                                 *)
val testResults = filter (fn x => x < 5) [1,2,3,4,5,6,7,8,9];
```

Appendix D

Source Code for Exercise 5.3.5

The following code is from *Ex-5-3-5.sml*

```
(*****************************************************************************)
(* Author: Kyle Peppe                                                        *)
(* Exercise 5.3.5                                                            *)
(* Date: 1/19/20                                                             *)
(*****************************************************************************)


fun addpair (x,y) = x + y;
fun Filter n xlist = filter(fn (x,y) => y>n andalso x>n) xlist;
fun addlist xlist = map addpair xlist;
fun addPairsGreaterThan y xlist = (addlist (Filter y xlist));

(* Running the new Fun using the provided Test Case                          *)
addPairsGreaterThan 0 [(0,1),(2,0),(2,3),(4,5)];
```

Appendix E

Source Code for Exercise 6.2.1

The following code is from *Ex-6-2-1.sml*

```
(*****************************************************************************)
(* Author: Kyle Peppe                                                        *)
(* Exercise 6.2.1                                                            *)
(* Date: 1/20/20                                                             *)
(*****************************************************************************)

(* Part 1                 *)
``x``;
``y``;
``P x``;
``Q y``;
``P x ==> Q y``;

(* Part 2                 *)
``x:num``;
``y:bool``;
``P x:num``;
``Q y:bool``;
``P (x:num) ==> Q (y:bool)``;

(* Part 3                 *)
``x``;
``y``;
``P x``;
``Q y``;
``!x. !y. P(x) ==> Q(y)``;

(* Part 4                 *)
``x:num``;
``R(x:num)``;
``?x. R(x:num)``;

(* Part 5                 *)
```~!x.(P(x) \/ Q(x)) = ?x.(~P(x) /\ ~Q(x))``;

(* Part 6   *)
``!x.P(x) ==> M(x)``;

(* Part 7   *)
``!x.P(x) ==> ?x.Funny(x)``;
```
```