William George
CS 161
October 29, 2014
Assignment 5 Report

Understanding:

The idea for this week is to develop an understanding of recursion and passing values by reference. While we had a lecture on passing by reference last week, we did not practice it at all. We are doing several coding exercises this week in which these skills are needed, including a few we attacked differently last week and the week before.

The first exercise randFun.cpp is one we worked on last week and one we will address differently this week. The concept will be the same, building a random number generator that will use some specified range but we will be using a function that accepts references rather than copies of values. It will accepted a referenced minimum and a referenced maximum then send the value of the third, randomly generated value to a third referenced value.

The second exercise retFun.cpp follows the same theme. We worked on a function last week that would accept to string values and evaluate if they were equal. This exercise will do the same but will reference two strings to check their equality.

The third exercise sortFun.cpp will use a function to takes three integers then pass them by reference to their assigned locations based on their value. In other words the function should sort the three values so that they are in order from smallest to largest.

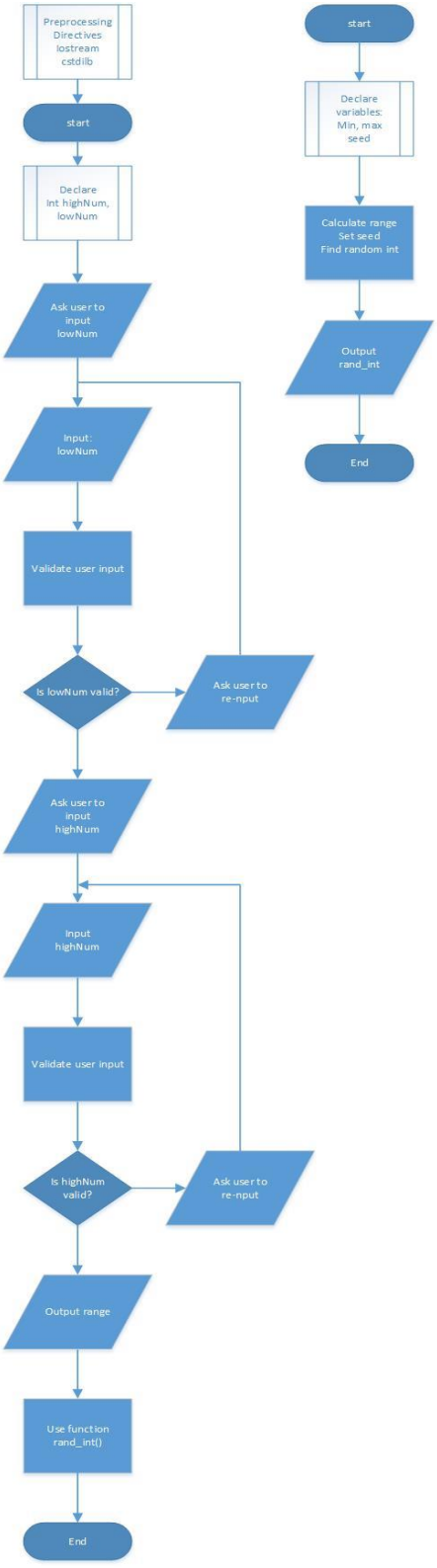
The fourth exercise, recFun.cpp, is the first one with recursion and has us creating a program that will take our integer then return the corresponding Fibonacci number based on that integer. So for example if we gave a 5 then the function would output the fifth number in the Fibonacci sequence or 8. This exercise is going to need recursion as conventional loops will not work.

The fifth and final exercise hailstone.cpp uses a recursive function as well, or a function that calls itself. With a given integer the function will either divide by two or multiply it by three and add one to get the next value in the sequence, based on whether the number is even or odd. The function should stop when it reaches the base condition or when the value is one.

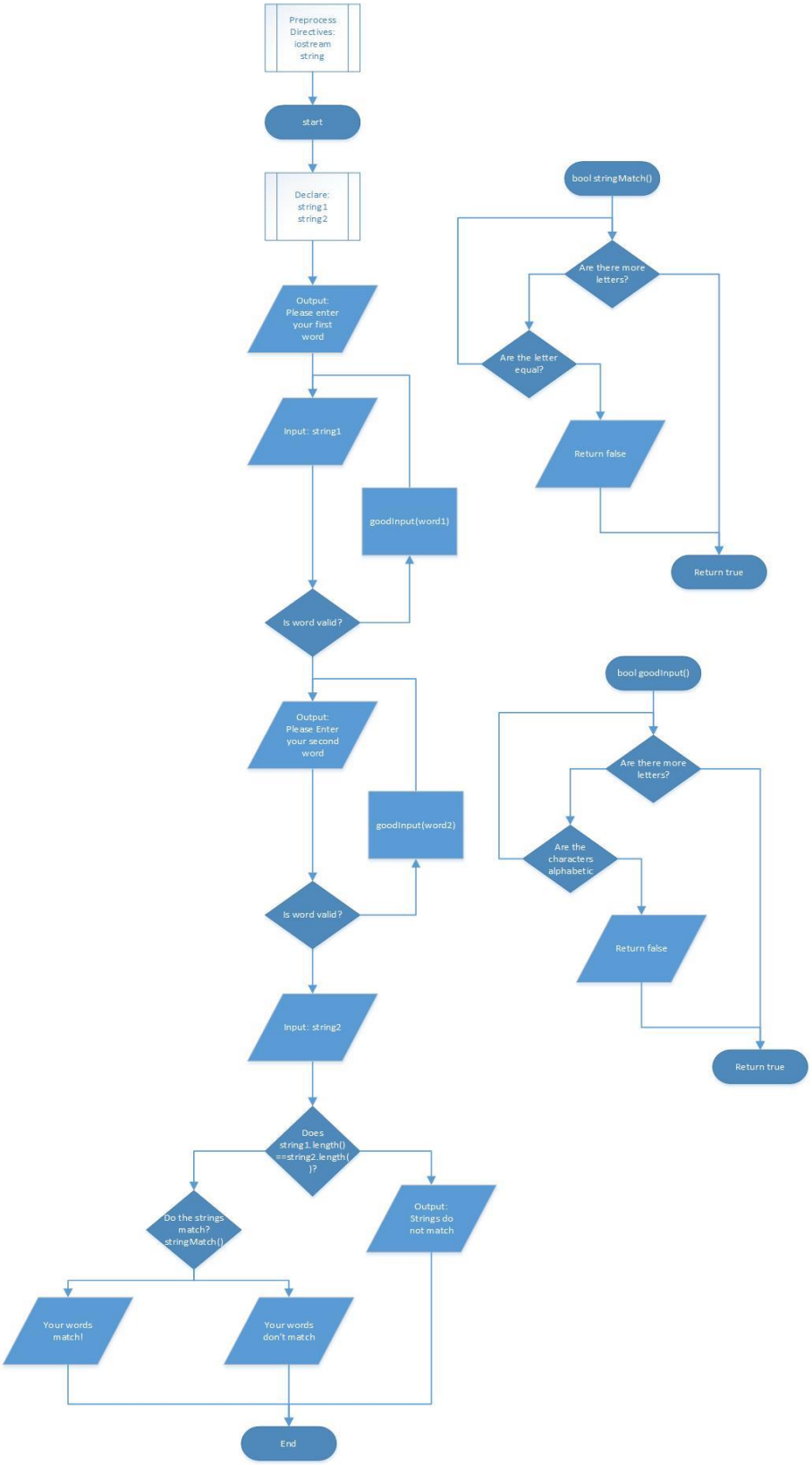
The project component this week called reconvert.cpp allows a user to call from a list of three things either convert a binary number to a decimal number, to convert a decimal number to binary or exit the program. The program then asks the user the number, checks if the number is a valid int and uses a recursive function to either convert the binary number to decimal or vice versa. One important element of this project is the input validation. Decimal numbers need to be validated as decimal integers and binary needs to be 0's and 1's.

Design:

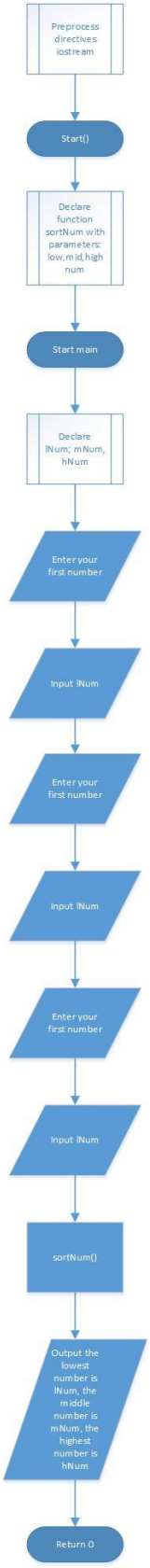
randFun.cpp

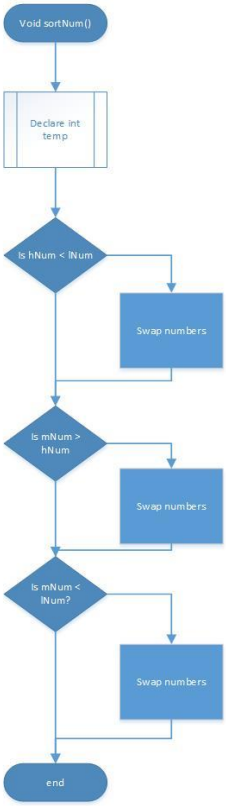


refFun.cpp

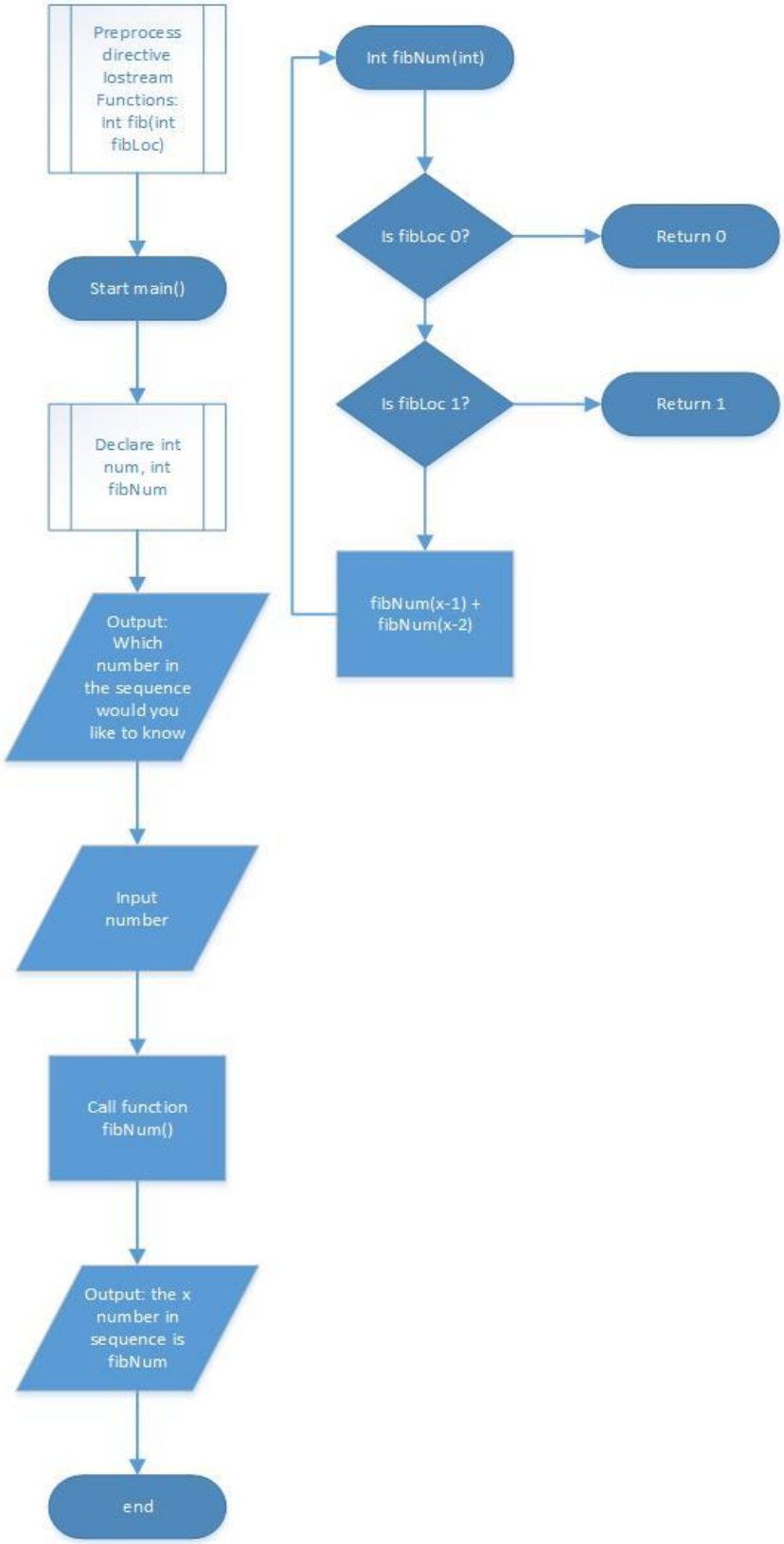

Preprocess Directives: iostream string
start
Declare: string1 string2
Output: Please enter your first word
Input: string1
goodInput(word1)
Is word valid?
Output: Please Enter your second word
goodInput(word2)
Is word valid?
Input: string2
Does string1.length() ==string2.length( )?
Do the strings match? stringMatch()
Output: Strings do not match
Your words match!
Your words don't match
End
bool stringMatch()
Are there more letters?
Are the letter equal?
Return false
Return true
bool goodInput()
Are there more letters?
Are the characters alphabetic
Return false
Return true

sortFun.cpp

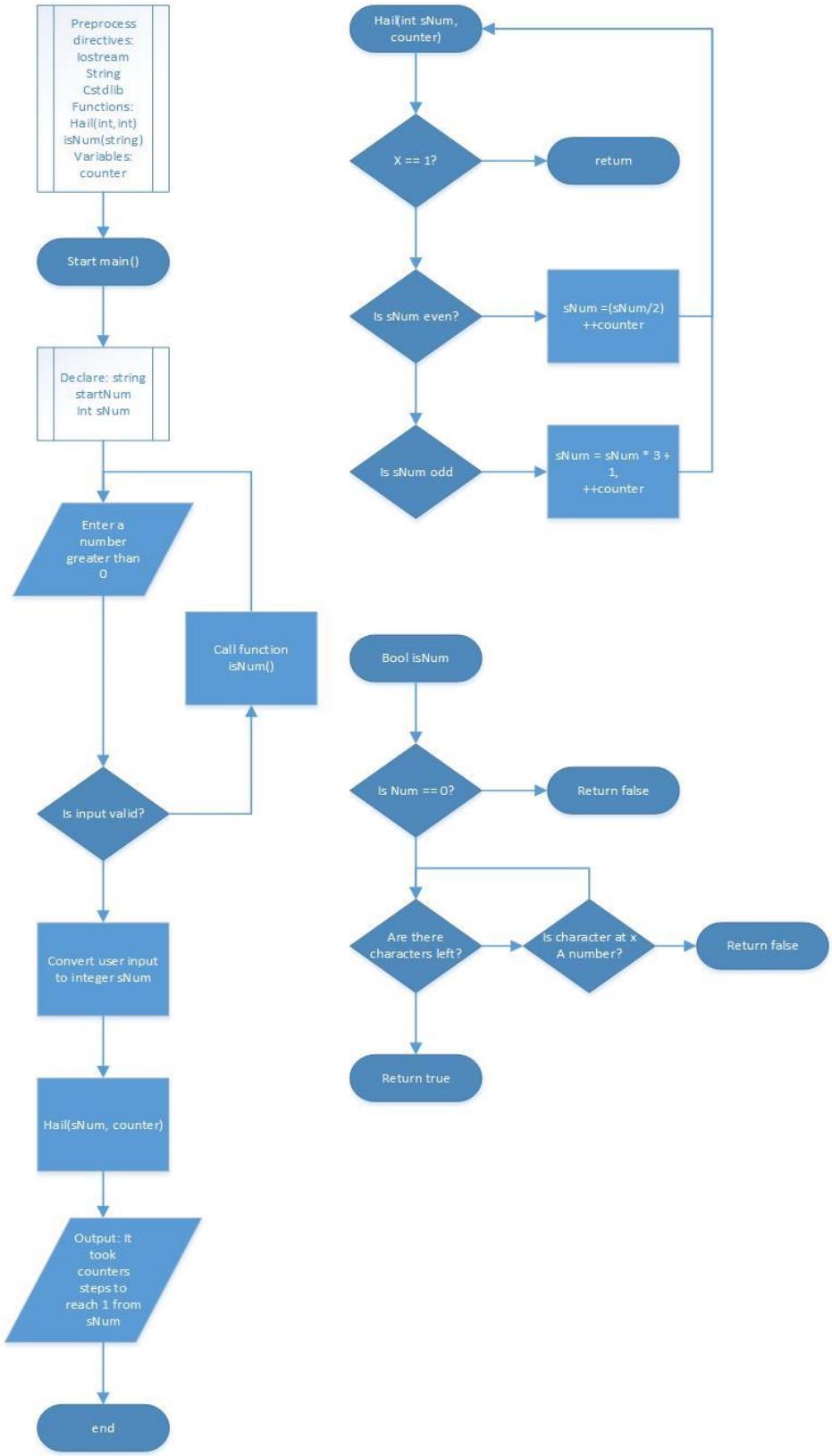

Preprocess directives iostream
Start()
Declare function sortNum with parameters: low, mid, high num
Start main
Declare lNum; mNum, hNum
Enter your first number
Input lNum
Enter your first number
Input lNum
Enter your first number
Input lNum
sortNum()
Output the lowest number is lNum, the middle number is mNum, the highest number is hNum
Return 0


Void sortNum()
Declare int temp
Is hNum < lNum
Swap numbers
Is mNum > hNum
Swap numbers
Is mNum < lNum?
Swap numbers
end

recFun.cpp


Preprocess directive Iostream Functions: Int fib(int fibLoc)
Start main()
Declare int num, int fibNum
Output: Which number in the sequence would you like to know
Input number
Call function fibNum()
Output: the x number in sequence is fibNum
end
Int fibNum(int)
Is fibLoc 0?
Return 0
Is fibLoc 1?
Return 1
fibNum(x-1) + fibNum(x-2)

hailstone.cpp



Project Design
reconvert.cpp
Pseudocode
Preprocess directives:
Iostream, string, cmath, cstdlib, algorithm

binaryToDecimal function;
decimalToBinary function;
menuFunction;
menuChoice function;
binaryInputValidationFunction;
decimalInputValidationFunction;

int decNum;
string binary;

main function()

int choice,dec
string decimal;

```
do
        menuFunction()
        choice = menuChoice()
        dec = 0
        decNum = 0
        binary = ‘ ‘
        switch
                case 1   output enter decimal number
                         input decimal
                         while inputValidationFunction is false
                                  re-input
                         convert decimal string to decNum int
                         decimalToBinary function
                         reverse string
                         output binary
                         break

                case 2 = output enter binary number
                         input binary
                         while binaryInputValidation is false
                                  re-input binary
                         binaryToDecimal function
                         output decimal
                         break
```

```
                case 3 = End Program
while menuChoice != 3;

return 0;

void displayMenu()
        clear screen
        output: Enter the corresponding number
        output: 1. Convert decimal to binary
        output: 2. Convert binary to decimal
        output: 3. Quit program

int getChoice()
        declare int choice
        input choice
        while choice is not between 1 and 3
                select a number between 1 and 3
        return choice

binaryToDecimal function(string biNum)
        int length
        length = biNum.length()
        if (length == 0)
                return decimal;
        if biNum.at(0) == 1
                decimal = decimal + pow(2, length-1);
        else
                decimal =decimal;
        biNum.erase(0,1);
        binaryToDecimal function(biNum);

decimalToBinary function (int decNum)
        int remainder
        if (num > 0)

                remainder = decNum % 2
                if remaider == 0
                        binary = binary + '0'
                if remainder == 1
                        binary = binary + '1'
                decNum = decNum / 2
                decimalToBinary function(decNum)
```

bool decimalInputValidation(string num)

```
For int i = 0 while i++ till length of num = i
        If num is not between 0 and 9
                Return false
Return true
```

bool binaryInputValidation(string num)

```
For int i = 0 while i++ till length of num = i
        If num is not 0 or 1
                Return false
Return true
```

Testing

randFun

Input	Input 2	Expected Output	Actual Output
15	17	Number is 15,16 or 17	Expected
32	15	Second input needs to be greater than first	Expected
-3	300	Enter a value greater than 0	Expected
15	S12	Invalid input	Expected
15S	17	Invalid Input	Expected
S15	17	Invalid input	Expected

refFun

Input 1	Input 2	Expected Output	Actual Output
String	String	Words match	expected
String	Strong	words don't match	expected
Str5ng	Strng	Invalid Entry	expected
55436	55436	Invalid Entry	expected

sortFun

Input 1	Input 2	Input 3	Expected result	Actual Result
1	2	3	1,2,3	expected
2	1	3	1,2,3	expected
3	2	1	1,2,3	expected
3	1	2	1,2,3	expected
1n, n1	2	3	Please re input	expected
1	2n, n2	3	Please re input	expected
1	2	3n, n3	Please re input	expected

recFun

Input	Expected Output	Actual Output
0	0	Expected
1	1	Expected
-1	Invalid input	Expected
3n	Invalid input	Expected
n	Invalid input	Expected
3	2	Expected

Hailstone

Input	Expected Output	Actual Output
0	Enter a valid number	Expected
1	0 steps	Expected
5	5 steps	Expected
5n	Enter a valid number	Expected
N	Enter a valid number	Expected
-1	Enter a valid number	Expected
12	9 steps	Expected

Reconvert

Input – Choice	Input – Binary	Input – Decimal	Expected Output	Actual Output
1			Please enter a decimal number	expected
3			Thank you good bye	Expected
1n			Please enter a valid input	Expected
N			Please enter a valid input	Expected
-1			Please enter a valid input	Expected
	1010		10	Expected
	1n1		Please enter a binary number	Expected
	5		Please enter a binary number	Expected
		5	101	Expected
		n	Please enter a valid number	Expected
		1n	Please enter a valid number	Expected
		N1	Please enter a valid number	Expected
		-1	Please enter a valid number	expected

Reflection:

Looking back this week had a directed focus on recursion and returning values by reference rather than by value as we have throughout the rest of the course. Each of the exercises had their own challenges. The refFun and randFun exercises were both focused on returning values by reference on functions we had already used to return by value. This really helped me gather an understanding of what is needed to do so and what it means to pass by reference. I changed my retFun program from a previous week so that it would return a Boolean input rather than the check on every word.

The sortFun exercise was challenging in trying to understand the logic of how to move each number around. It took me longer than I thought it would because I placed an inequality sign incorrectly. The recFun program using the Fibonacci sequence was difficult as was the hailstone program because the thought of answering a question inside out is not something we normally do. Being able to solve the problem backwards can be a good skill, especially since it's something we regularly do when programming.

Finally with the project for the week, reConvert.cpp, I had a few issues which took me a long time to figure out. The binary to decimal problem was easy for me as the program required I solve it the way I normally do, by breaking down each binary digit into its value based on its position in the string then adding them all together. The logic for the decimal to binary was difficult, because the way I solve those problems would have been difficult to code, so I had to learn a new means of doing so. Eventually I found an algorithm on wikiHow that worked well. My next issue was the string I was creating was printing backwards, luckily another student posted a function that worked to fix that issue. Finally I was unsure how to work the input validation. Eventually I went back to the lectures and saw there was a lecture that had an applicable input validation function and used that which worked well. There was a lot of frustration with this program, not because it was difficult but because I made some silly mistakes with misplaced semi-colons and other characters. However, I'm happy with how the program turned out.