Angie Chen
GE1501 18472 SEC 10
Professor Hertz
2 October 2016

Assignment 4 Report: Expanding Exponentially like some Recursive Virus

To accommodate the band's crowd's complex growing or decreasing proportion of fans, a program based on an equation and a given R constant, estimations can be made about the band's success or failure. The given equation (below) representing the population mentality, where x_{n+1} represents the popularity in the current week and x_n represents the popularity of the week before, accommodates two phenomena: growth of popularity "in proportion to the number of people who are currently fans", and the loss of some fans who have an aversion to the band's increasing popularity (2 Hertz).

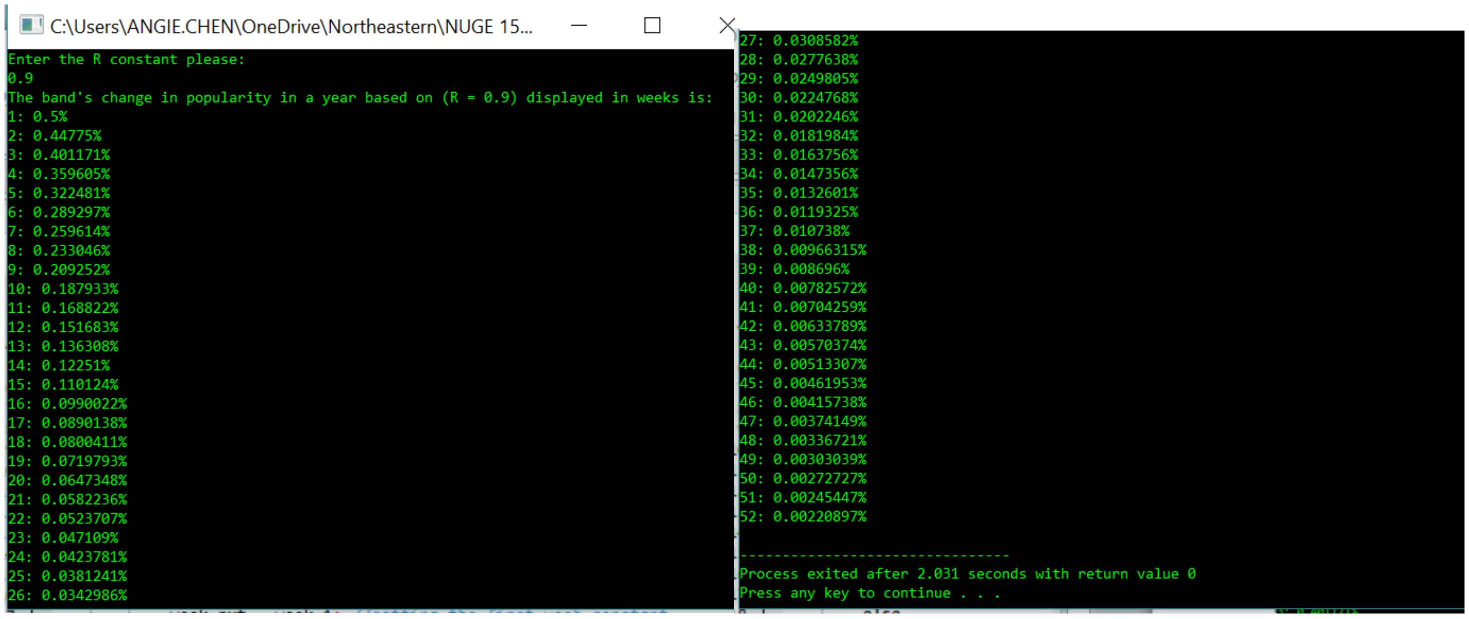
$$x_{n+1} = R * x_n * (1 - x_n)$$

The first week's percentage of popularity ($x_1 = 0.005$), and three sample R values (0.90, 1.50, 3.95) were given. To truly understand the outcome of the band's success or failure, weekly outputs reflecting the band's weekly changing popularity are displayed.

As usual, I brainstormed the general sequence of logic this program should follow. The idea of this program reminded me of how a program producing the Fibonacci sequence ran (shown in Appendix A and Fig. #) (ProgrammingSimplified). Following this structure, I composed my program after writing a series of pseudocode displayed below. In addition to pseudocode, the isolation and checking of the functionality of the equation itself is shown in Appendix B.

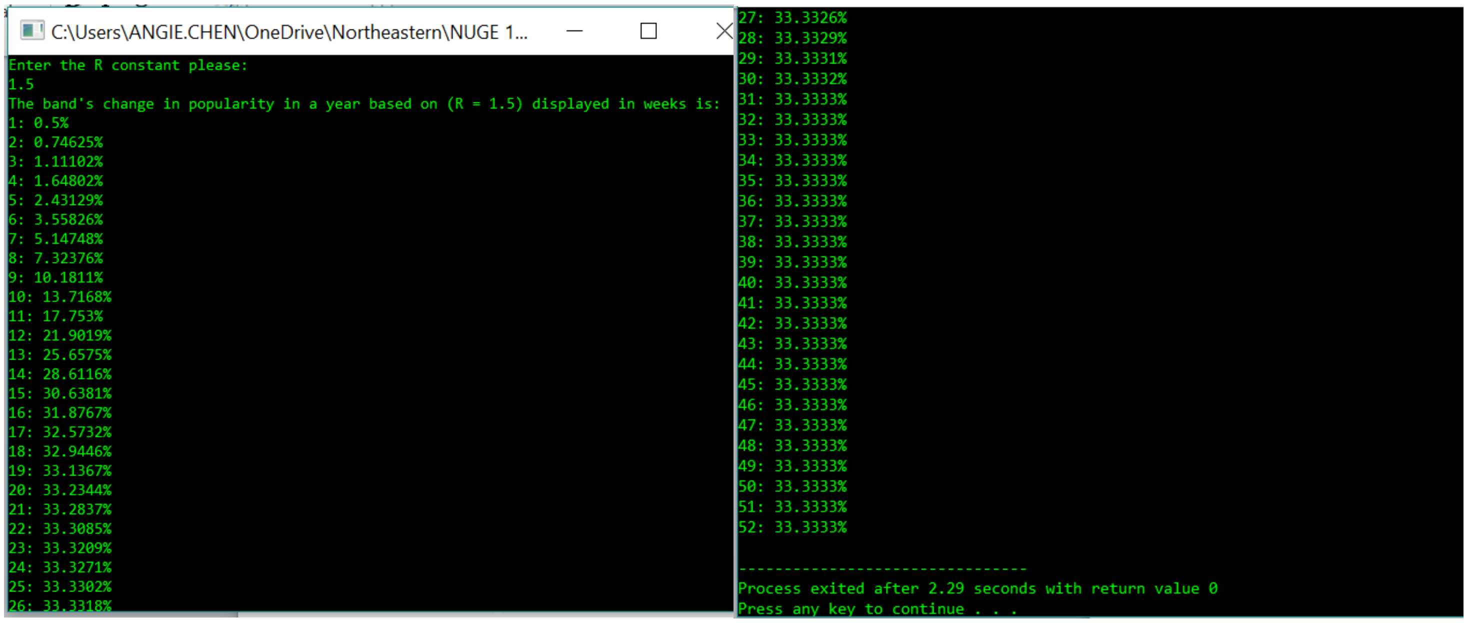
```
1-2     Include iostream, library for more complex arithmetic
4       Maintain standard versions of functions/variables
5       Function automatically applied to every line of code below
8       Ensuring that the all of the variables can store decimals
10-11   Asking the user for an R input to store
12      Display the answer sentence for proper run order
14      Initializing a more compact form of do/while, and
        increasing the week number by 1 each cycle up to 52 cycles
15-17   Declaring and defining the value for the first week
18-23   Calculating the value for the subsequent week and
        declaring the value as the initial value for the next cycle
27      End program
```

Below are screenshots and analysis of the success of the band predicted using the given R constants. Appendix B contains the code for the program.



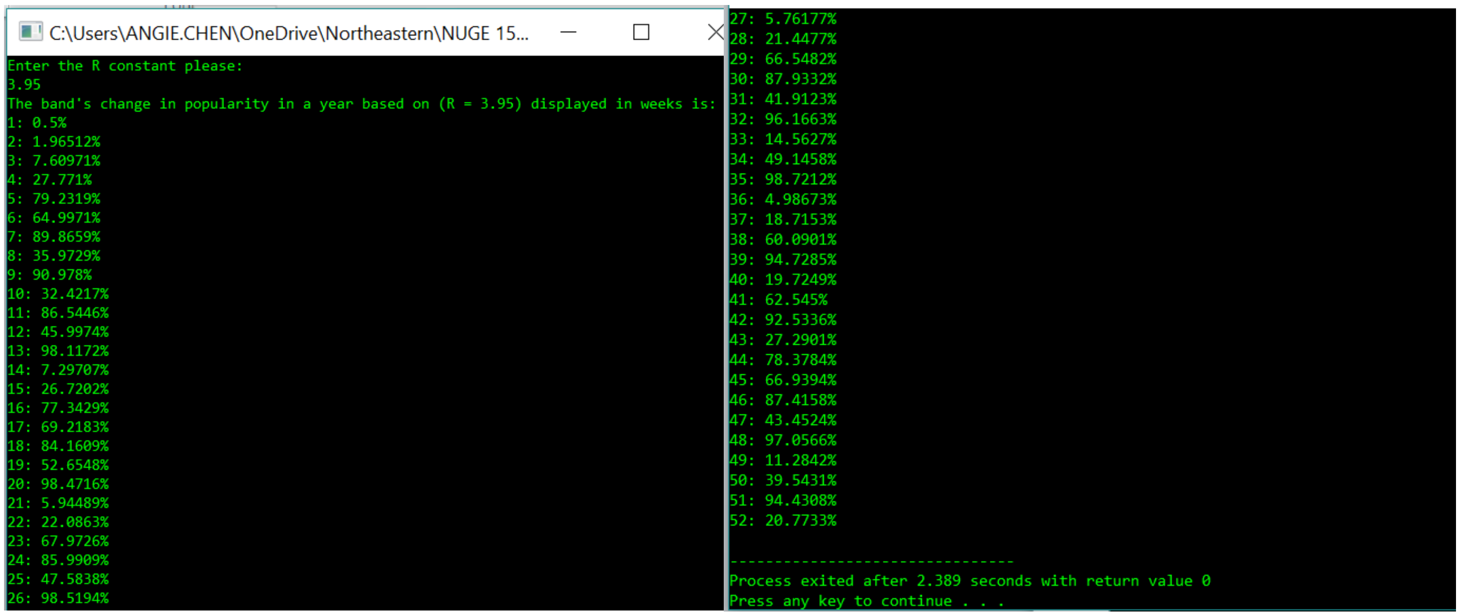
```
Enter the R constant please:
0.9
The band's change in popularity in a year based on (R = 0.9) displayed in weeks is:
1: 0.5%
2: 0.44775%
3: 0.401171%
4: 0.359605%
5: 0.322481%
6: 0.289297%
7: 0.259614%
8: 0.233046%
9: 0.209252%
10: 0.187933%
11: 0.168822%
12: 0.151683%
13: 0.136308%
14: 0.12251%
15: 0.110124%
16: 0.0990022%
17: 0.0890138%
18: 0.0800411%
19: 0.0719793%
20: 0.0647348%
21: 0.0582236%
22: 0.0523707%
23: 0.047109%
24: 0.0423781%
25: 0.0381241%
26: 0.0342986%
27: 0.0308582%
28: 0.0277638%
29: 0.0249805%
30: 0.0224768%
31: 0.0202246%
32: 0.0181984%
33: 0.0163756%
34: 0.0147356%
35: 0.0132601%
36: 0.0119325%
37: 0.010738%
38: 0.00966315%
39: 0.008696%
40: 0.00782572%
41: 0.00704259%
42: 0.00633789%
43: 0.00570374%
44: 0.00513307%
45: 0.00461953%
46: 0.00415738%
47: 0.00374149%
48: 0.00336721%
49: 0.00303039%
50: 0.00272727%
51: 0.00245447%
52: 0.00220897%

--------------------------------
Process exited after 2.031 seconds with return value 0
Press any key to continue . . .
```

When R is defined as 0.9, it can be seen that the band ends up steadily losing popularity.



```
Enter the R constant please:
1.5
The band's change in popularity in a year based on (R = 1.5) displayed in weeks is:
1: 0.5%
2: 0.74625%
3: 1.11102%
4: 1.64802%
5: 2.43129%
6: 3.55826%
7: 5.14748%
8: 7.32376%
9: 10.1811%
10: 13.7168%
11: 17.753%
12: 21.9019%
13: 25.6575%
14: 28.6116%
15: 30.6381%
16: 31.8767%
17: 32.5732%
18: 32.9446%
19: 33.1367%
20: 33.2344%
21: 33.2837%
22: 33.3085%
23: 33.3209%
24: 33.3271%
25: 33.3302%
26: 33.3318%
27: 33.3326%
28: 33.3329%
29: 33.3331%
30: 33.3332%
31: 33.3333%
32: 33.3333%
33: 33.3333%
34: 33.3333%
35: 33.3333%
36: 33.3333%
37: 33.3333%
38: 33.3333%
39: 33.3333%
40: 33.3333%
41: 33.3333%
42: 33.3333%
43: 33.3333%
44: 33.3333%
45: 33.3333%
46: 33.3333%
47: 33.3333%
48: 33.3333%
49: 33.3333%
50: 33.3333%
51: 33.3333%
52: 33.3333%

--------------------------------
Process exited after 2.29 seconds with return value 0
Press any key to continue . . .
```

When R is defined as 0.9, it can be seen that the band’s popularity evens out at about 33%.



C:\Users\ANGIE.CHEN\OneDrive\Northeastern\NUGE 15...

```
Enter the R constant please:
3.95
The band's change in popularity in a year based on (R = 3.95) displayed in weeks is:
1: 0.5%
2: 1.96512%
3: 7.60971%
4: 27.771%
5: 79.2319%
6: 64.9971%
7: 89.8659%
8: 35.9729%
9: 90.978%
10: 32.4217%
11: 86.5446%
12: 45.9974%
13: 98.1172%
14: 7.29707%
15: 26.7202%
16: 77.3429%
17: 69.2183%
18: 84.1609%
19: 52.6548%
20: 98.4716%
21: 5.94489%
22: 22.0863%
23: 67.9726%
24: 85.9909%
25: 47.5838%
26: 98.5194%
27: 5.76177%
28: 21.4477%
29: 66.5482%
30: 87.9332%
31: 41.9123%
32: 96.1663%
33: 14.5627%
34: 49.1458%
35: 98.7212%
36: 4.98673%
37: 18.7153%
38: 60.0901%
39: 94.7285%
40: 19.7249%
41: 62.545%
42: 92.5336%
43: 27.2901%
44: 78.3784%
45: 66.9394%
46: 87.4158%
47: 43.4524%
48: 97.0566%
49: 11.2842%
50: 39.5431%
51: 94.4308%
52: 20.7733%


--------------------------------
Process exited after 2.389 seconds with return value 0
Press any key to continue . . .
```

When R is defined as 0.9, the band’s change in popularity seems random, with an even amount of peaks of successes and dips of extreme uninterest.

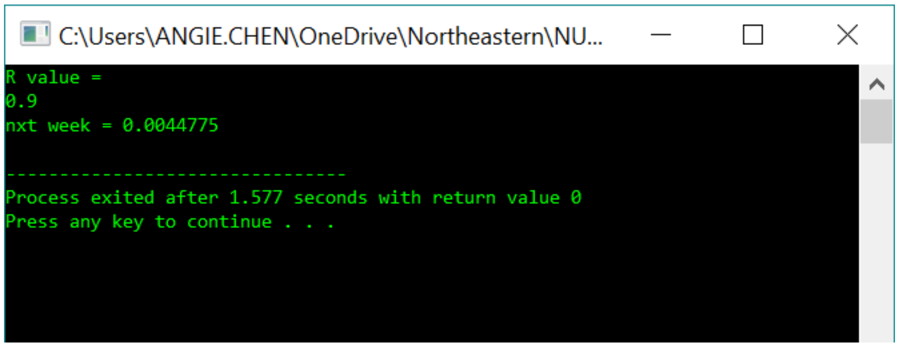
Works Cited

Hertz, Joshua. “A4: Expanding Exponentially like some recursive virus”. Blackboard.neu.edu. 2016. Web. 30 September 2016.

Programming Simplified. “C++ program to generate Fibonacci series”. Programming Simplified: Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. Web. 1 October 2016.

Appendix A

```
#include <cstdlib>

using namespace std;
int main()

{
        int n, c, first = 0, second = 1, next;

        cout << "Enter the number of terms of Fibonacci series you want" << endl;
        cin >> n;

        cout << "The first " << n << " terms of the Fibonacci series are : " << endl;

        for ( c = 1; c <= n; c++)
                {
                if ( c <= 1 )
                  next = c;
                else
                  {
                  next = first + second;
                  first = second;
                  second = next;
                  }
                cout << c << ": " << next << endl;
                }

        return 0;
}
```



```
R value =
0.9
nxt week = 0.0044775

--------------------------------
Process exited after 1.577 seconds with return value 0
Press any key to continue . . .
```

The output value is equivalent to that of true program where R = 0.90.

Appendix B

```
using namespace std;
int main()

{
        float R, week_1, week_nxt;

        cout << "R value = " << endl;
        cin >> R;

        week_1 = 0.005;

        week_nxt = (R * week_1 * (1-week_1));
        cout << "nxt week = " << week_nxt << endl;

        return 0;

}
```

Appendix C

```
#include <iostream>
#include <cstdlib>

using namespace std;
int main()
```

```
{
	float R, week_1 = 0.005, week_nxt, week_num; //ensuring that the code can account for
decimals

	cout << "Enter the R constant please: " << endl;
	cin >> R;
	cout << "The band's change in popularity in a year based on (R = " << R << ") displayed
in weeks is: " << endl;

	for(week_num = 1; week_num <= 52; week_num ++) //increasing the week number by 1
			{
			if( week_num <= 1)
			   week_nxt = week_1; //setting the first week constant
			else
			   {
			   week_nxt = R * week_1 * (1 - week_1); //defining the equation for the
change in popularity
			   week_1 = week_nxt; //declaring the new value for "week_1" in the next
cycle
			   }
			cout << week_num << ": " << week_nxt * 100 << "%" << endl; //printing
the week number and the popularity percentage
			}

	return 0;
}
```