Assignment 1
CPSC 331
Wayne Eberly
CPSC 331 (L01)

30041703

Nguyen, Leslie

30042040

Tran, Jordan

30054457

Yang, Jeffery

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

1.

Given the precondition, n is an integer input such that $n \geq 0$, then the function is a integer-valued function. Whenever the function is called again within itself, recursively on lines 10 or 11 (exclusively), the integer value given to those functions is decreased by at least one therefore the total value of the function is decreased by at least one. If the value $f(n) \leq 0$, then $n \leq 0$. However, given the precondition that $n \geq 0$, $0 \leq n \leq 0$, so $n = 0$. When the value of the integer, n, is zero the first line of the algorithm holds true and the second line is executed to return an output of 1 giving the n^{th} Grindelwald number, therefore never being called recursively thus terminating the program (Eberly, 2019).

2.

Note that the strong induction format used is from the lecture notes (Eberly, 2019).

Basis:

Let n = 0
Lines 1-2 executes, and 1 is returned
A Grindelwald number is returned as output and thus the postcondition is satisfied

Let n = 1
Line 1 fails, Lines 3-4 execute, and 2 is returned
A Grindelwald number is returned as output and thus the postcondition is satisfied

Let n = 2
Lines 1 and 3 fail, Lines 5-6 execute and 3 is returned
A Grindelwald number is returned as output and thus the postcondition is satisfied

Let n = 4
Lines 1, 3, and 5 fail, Lines 7-8 execute and 4 is returned
A Grindelwald number is returned as output and thus the postcondition is satisfied

Induction Step: Let k be an integer such that $k \geq 4$

> **Inductive Hypothesis:** Let n be an integer such that $0 \leq n \leq k$. Assume the algorithm is executed with the input n = k, and that the algorithm eventually ends with the k^{th} Grindelwald number = $G_k = G_n$ as the output.
>
> **Inductive Claim:** We want to prove that if the algorithm is executed with n = k+1, then the algorithm will end and output the G_{k+1} Grindelwald number.

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

Proof:

$n = k + 1$, and $k \geq 4$, so $n \geq 5$.

Case 1 - k+1 is even:

Assume k+1 is even, then Lines 1-8 fail, and Lines 9-10 execute

At Line 10:

sGrin(n-1) is called,

$n \geq 5$,

$n-1 \geq 4$,

$n-1 = (k+1) -1 = k$

$4 \leq k \leq k$, thus by the inductive hypothesis it will end and return the G_k Grindelwald number

sGrin(n-3) is called,

$n \geq 5$,

$n-3 \geq 2$,

$n-3 = (k+1) -3 = k - 2$

$2 \leq k-2 \leq k$, thus by the inductive hypothesis it will end and return the G_{k-2} Grindelwald number

sGrin(n-4) is called,

$n \geq 5$,

$n-4 \geq 1$,

$n-4 = (k+1) -4 = k - 3$

$1 \leq k-3 \leq k$, thus by the inductive hypothesis it will end and return the G_{k-3} Grindelwald number

$2 \times sGrin(n - 1) - 2 \times sGrin(n - 3) + sGrin(n - 4)$

$= 2*G_k - 2*G_{k-2} + G_{k-3} = G_{k+1}$ By the definition of the sequence

Case 2 - k+1 is odd:

Assume k+1 is odd, then Lines 1-10 fail, and Line 11 executes

sGrin(n-1) is called which was proved above,

sGrin(n-2) is called,

$n \geq 5$,

$n-2 \geq 3$,

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

n-2 = (k+1) -2 = k - 1
3 ≤ k-1 ≤ k, thus by the inductive hypothesis it will end and return the G_{k-1} Grindelwald number

sGrin(n-3) is called which was proved above
sGrin(n-4) is called which was proved above
= sGrin(n − 1) + 3 × sGrin(n − 2) − 5 × sGrin(n − 3) +2 × sGrin(n − 4)
= G_k +3*G_{k-1} -5G_{k-2} G_{k-3} = G_{k+1} By the definition of the sequence

Since the value of these 2 outputs are both G_{k+1} , this establishes the inductive claim and completes the proof.

3.

```java
package cpsc331.assignment1;
/**
30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

description: this program executes an algorithm to solve the Grindelwald sequence problem using a recursive function
**/

public class SGrindelwald {
	public static void main(String[] args) {
		int n;
		// attempt to read from command line
		try {
			if(args.length != 1) throw new Exception();
			n = Integer.parseInt(args[0]);
		}catch(Exception e) {
			// catch if an error occur
			System.out.println("Gadzooks! One integer input is required.");
			return;
		};

		try{
		System.out.println(sGrin(n));
	    }catch(IllegalArgumentException e){
	       System.out.println("Gadzooks! The integer input cannot be negative.");
	    }
	}

	protected static int sGrin (int n) {
		if(n < 0) {
			throw new IllegalArgumentException();

		}
```

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

```
        // Assertion: A non-negative integer n has been given as input.
                if (n == 0) {
                    return 1;
                } else if (n == 1) {
                        return 2;
                } else if (n == 2) {
                        return 3;
                } else if (n == 3) {
                        return 4;
                } else if ((n % 2) == 0) {
                        return 2 * sGrin(n - 1) - 2 * sGrin(n - 3) + sGrin(n - 4);
                } else {
                        return sGrin(n - 1) + 3 * sGrin(n - 2) - 5 * sGrin(n - 3) +2 * sGrin(n - 4);
                }
        }
}
```

4.

Note that recurrence format used is from the lecture notes (Eberly, 2019).

$$T_{sGrin}(n) = \begin{cases} 2 & n = 0 \\ 3 & n = 1 \\ 4 & n = 2 \\ 5 & n = 3 \\ T_{sGrin}(n-1) + T_{sGrin}(n-3) + T_{sGrin}(n-4) + 6 & n \geq 4 \end{cases}$$

5.

Note that the strong induction format used is from the lecture notes (Eberly, 2019).

Basis:

Let n = 0,

$T_{sGrin}(0) = 2$

$(3/2)^0 = 1$

$2 \geq 1$, therefore f(n) is true

Let n = 1,

$T_{sGrin}(1) = 3$

$(3/2)^1 = (3/2)$

$3 \geq (3/2)$, therefore f(n) is true

Let n = 2,

$T_{sGrin}(2) = 4$

$(3/2)^2 = (9/4)$

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

$4 \geq (9/4)$, therefore f(n) is true

Let n = 3,
$T_{sGrin}(3) = 5$
$(3/2)^3 = (27/8)$
$5 \geq (27/8)$, therefore f(n) is true

Inductive Step: Let k be an integer such that $k \geq 0$.
Inductive Hypothesis: Let n be an integer such that $0 \leq n \leq k$, and assume that $TsGrin(n) \geq (3/2)^n$

Inductive Claim:
We want to show that $T_sGrin(k+1) \geq (3/2)^{k+1}$

Proof:

Case 1: k+1 is even:
$T_{sGrin}(k+1) = 6 + sGrin(k+1-1) + sGrin(k+1-3) + sGrin(k+1-4)$
By the inductive hypothesis, states that since $n \leq k$, we can assume n = k
$T_{sGrin}(n+1) = 6 + sGrin(n+1-1) + sGrin(n+1-3) + sGrin(n+1-4)$
$T_{sGrin}(n+1) = 6 + sGrin(n) + sGrin(n-2) + sGrin(n-3)$
$= 6 + (3/2)^n + (3/2)^{n-2} + (3/2)^{n-3}$ By IH
$= 6 + (3/2)^{n-1}(3/2) + (3/2)^{n-1}(3/2)^{-1} + (3/2)^{n-1}(3/2)^{-2}$
$= 6 + (3/2)^{n+1}(3/2)^{-1} + (3/2)^{n+1}(3/2)^{-3} + (3/2)^{n+1}(3/2)^{-4}$
$= 6 + (3/2)^{n+1}((3/2)^{-1} + (3/2)^{-3} + (3/2)^{-4})$
$= 6 + (3/2)^{n+1}(94/81)$

$(94/81) \geq 1$, so
$T_{sGrin}(n+1) = 6 + (3/2)^{n+1}(94/81) \geq (3/2)^{n+1}$, thus
$T_{sGrin}(k+1) \geq (3/2)^{k+1}$ is true when k+1 is even.

Case 2: k+1 is odd:
$T_{sGrin}(k+1) = 6 + sGrin((k+1)-1) + sGrin((k+1)-2) + sGrin((k+1)-3) + sGrin((k+1)-4)$
By the inductive hypothesis, states that since $n \leq k$, we can assume n = k
$T_{sGrin}(n+1) = 6 + sGrin((n+1)-1) + sGrin((n+1)-2) + sGrin((n+1)-3) + sGrin((n+1)-4)$
$= 6 + sGrin(n) + sGrin(n-1) + sGrin(n-2) + sGrin(n-3)$
$= 6 + (3/2)^n + (3/2)^{n-1} + (3/2)^{n-2} + (3/2)^{n-3}$ By IH
$= 6 + (3/2)^{n+1}(3/2)^{-1} + (3/2)^{n+1}(3/2)^{-2} + (3/2)^{n+1}(3/2)^{-3} + (3/2)^{n+1}(3/2)^{-4}$
$= 6 + (3/2)^{n+1}((3/2)^{-1} + (3/2)^{-2} + (3/2)^{-3} + (3/2)^{-4})$
$= 6 + (3/2)^{n+1}(130/81)$

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

$(130/81) \geq 1$, so
$T_{sGrin}(n+1) = 6 + (3/2)^{n+1}(130/81) \geq (3/2)^{n+1}$, therefore
$T_{sGrin}(k+1) \geq (3/2)^{k+1}$ is true when k+1 is odd.

Since $T_{sGrin}(k+1)$ is true for both cases when k+1 is odd or when k+1 is even, this establishes the inductive claim and completes the proof.

6.

1. n is an input integer such that $n \geq 4$
2. i is an integer variable such that $4 \leq i \leq n+1$
3. $G[k] = G_k$ for $0 \leq k < i$

7.

Proof 1:

Using loop theorem #1 we will prove that the first assertion stated in question 6 is a loop invariant for fGrin on steps 15-19 (Eberly, 2019). The loop test on step 15 is only comparing integers and does not change the value of any inputs, variables, or global and therefore has no side effects or changes to the system state. For the algorithm to be executed n must meet the precondition of being a non-negative integer and for the loop to be reached steps 1-7 must not hold true, meaning that $n \geq 4$. Therefore, the first assertion from the previous question holds true. From analysing the loop body, it can be seen that the value of n is never changed or modified, therefore when the assertion of $n \geq 4$ holds at the beginning of the body and before the loop it holds at the end of the loop and after the while loop ends. Therefore the assertion holds.

Proof 2:

By inspection of the code it does not change the value of any inputs, variables, or global and therefore has no side effects or changes to the system state .

On line 14 i is initialized and defined as integer such that i = 4. Thus, $i \geq 4$ is true. By the first assertion we know that when the while loop is reached $n \geq 4$ is true so the loop test on line 15 will pass the first time. Therefore $1 \leq i \leq n+1$ is true at the beginning of every while loop execution.

By the loop test on line 15, it will only fail if $i > n$. If the loop test passes we know that $i \leq n$ is true. Thus it holds at the beginning of every body loop execution. On step 19 we can see that i is being incremented by 1 every execution of the while loop body. If the

loop test fails then we know $i + 1 > n$, since the test loop body ran for the last execution. That means that $i-1 = n$. Thus, $i = n +1$, and $1 \leq i \leq n+1$ is true at the end of the while loop body execution and the end of every execution of the while loop. Therefore the assertion holds.

Proof 3:
By inspection of the loop test, it is an integer comparison and it does not modify any variables, inputs, or global data. Therefore, there are no side-effects.

If the precondition is satisfied and the while loop is reached, when the execution of the loop begins, $i = 4$. Then there are 4 possible values of k for this i value such that $0 \leq k < i$, $k = 0$, $k = 1$, $k = 2$ or $k = 3$,

If $k = 0$, then
$G[k] = G_k$,
$G[0] = 1$ As defined at line 10
$G_0 = 1$
$G[0] = 1 = 1 = G_0$ so it is true when $k = 0$,

If $k = 1$, then
$G[k] = G_k$,
$G[1] = 2$ As defined at line 11
$G_1 = 2$
$G[1] = 2 = 2 = G_1$ so it is true when $k = 1$,

If $k = 2$, then
$G[k] = G_k$,
$G[2] = 3$ As defined at line 12
$G_2 = 3$
$G[2] = 3 = 3 = G_2$ so it is true when $k = 2$,

If $k = 3$, then
$G[k] = G_k$,
$G[3] = 4$ As defined at line 13
$G_3 = 4$
$G[3] = 4 = 4 = G_3$ so it is true when $k = 3$,

Therefore the assertion is satisfied when the loop is reached.

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

Before the while loop is initialized, the first 4 terms of the sequence is defined within the array, thus at the start of the while loop,G[k] = G_k , for all k where $0 \leq k < i$, where 'i' is initialized as 4 on step 14.

Since G[k] = k holds for the beginning of the execution then,
G[k] = 2 × G[k − 1] − 2 × G[k − 3] + G[k − 4]
$G_k = 2*G_{k-1} - 2*G_{k-3} + G_{k-4}$, which by definition of the sequence is the k^{th} (G_k) Grindelwald number when k is even.
G[k] = G[k − 1] + 3 × G[k − 2] − 5 × G[k − 3] + 2 × G[k − 4]
$G[k] = G_{k-1} + 3*G_{k-2} - 5*G_{k-3} + 2*G_{k-4}$ which by definition of the sequence is the k^{th} (G_k) Grindelwald number when k is odd.

Since k < i, implies k+1 < i+1, therefore G[k+1] = G_{k+1} holds true for the execution of the beginning and end of the body loop and after the while loop end execution.

Since all three assertions for Loop Theorem 1 holds (Eberly, 2019), it implies that assertions that were listed above hold as loop invariances for the while loop.

8.

By the precondition, we know that any input n is an integer such that $n \geq 0$.
If n = 0, then lines 1-2 will execute and return 1 and ends with the postcondition satisfied
If n = 1, then line 1 will fail to execute, lines 3-4 will execute and return 2 and ends with the postcondition satisfied
If n = 2, then lines 1 and 3 will fail, lines 5-6 will execute and return 3 and ends with the postcondition satisfied
f n = 3, then lines 1, 3, and 5 will fail, lines 7-8 will execute, and return 1 and ends with the postcondition satisfied

By assertions 1 and 2 we know that n is an integer input and i is a variable integer such that
$4 \leq i \leq n+1$ when the loop is reached.
If the loop test fails then, by evaluation of the loop test we know that, i > n. Since $0 \leq n < i$, it follows by assertion 3 that when G[n] is returned on step 20 it will return the G_n Grindelwald number and satisfy the postcondition.
Therefore it is partially correct.

9.

Let the bound function be f(n,i) = n - i + 1

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

By inspection of the code, we can see that the integer variable ‘i’ is being initialized at step 14 being assigned the value 4. Since the first 4 test cases failed implies that n is greater than or equal to 4, and is fixed to that assigned value. The variable ‘i’ is shown to be incremented by 1 at step 19, thus shows that the functions-value is decreasing by at least one per iteration. As shown by taking the change-in ‘i’

$$\begin{aligned} f(n, \Delta i) = f(n, i) - f(n, i+1) &= (n - i + 1) - (n - (i+1) + 1) \\ &= (n - i + 1) - (n - i - 1 + 1) \\ &= n - i + 1 - n + i \\ &= 1 \end{aligned}$$

Once the bound function is 0, it implies that i = n+1, the loop’s test case will fail (step 15) since n+1 = i is strictly greater than n (i = n+1 > n). Therefore proves that the program will terminate once the bound function is 0.

f(n, i) = f(n, n+1) = n - (n+1) + 1 = n - n - 1 + 1 = 0

10.

Using loop theorem #2 we will prove that the algorithm fGrin eventually terminates when the precondition for the “Grindelwald Gaggle Computation” problem is satisfied, and the while loop is reached and executed (Eberly, 2019). When n=0, line 1 hold true and line 2 is executed to return the value of 1 and the algorithm terminates. When n=1, line 3 hold true and line 4 is executed to return the value of 2 and the algorithm terminates. When n=2, line 5 hold true and line 6 is executed to return the value of 3 and the algorithm terminates. When n=3, line 7 hold true and line 8 is executed to return the value of 4 and the algorithm terminates. When the test at lines 1, 3, 5, and 7 fail and lines 9-14 are executed and the while loop begins. In addition, the loop test does not change the value of any inputs, variables, or global data so therefore has no side effects. When the algorithm’s precondition is satisfied n is a nonnegative number and for n to reach the loop . The loop test and body do not change or modify the value of n so at every execution of the loop n remains a non negative number and this property holds at the end. From the prove in question 9 there does exist a bound function for this loop (Eberly, 2019). Therefore, the loop satisfies all of the loop theorem #2’s properties and will terminate when the precondition is satisfied the algorithm terminates (Eberly, 2019).

11.

If an algorithm is partially correct and terminates, then it follows that the algorithm is correct (Eberly, 2019). We proved that the algorithm is partially correct in question 8 and

that the algorithm terminates in question 10, therefore this algorithm is correct when the precondition is satisfied.

12.

Note that recurrence format used is from the lecture notes (Eberly, 2019).

$$T_{fGrin}(n) = \begin{cases} 2 & n=0 \\ 3 & n=1 \\ 4 & n=2 \\ 5 & n=3 \\ 4n & n \geq 4 \end{cases}$$

13.

```
package cpsc331.assignment1;

/**
30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

description: this program executes an algorithm to solve the Grindelwald sequence problem using a non-recursive function
**/
public class FGrindelwald {

 public static void main(String[] args) {
   int n;
   // attempt to read from command line
         try {
               if(args.length != 1) throw new Exception();
               n = Integer.parseInt(args[0]);
         }catch(Exception e) {
                 // catch if an error occur
                 System.out.println("Gadzooks! One integer input is required.");
                 return;
         };

         try{
           System.out.println(fGrin(n));
       }catch(IllegalArgumentException e){
```

```
            System.out.println("Gadzooks! The integer input cannot be negative.");
        }
 }
  protected static int fGrin(int n) {
     if(n < 0) {
           throw new IllegalArgumentException();
         }

     if (n == 0) {
       return 1;
     }
     else if (n == 1) {
       return 2;
     }
     else if (n == 2) {
       return 3;
     }
     else if (n == 3) {
       return 4;
     }
     else {
       int[] G = new int[n + 1];
       G[0]=1;
       G[1]=2;
       G[2]=3;
       G[3]=4;
       int i=4;
       while (i <= n) {
         if (i%2 == 0) {
           G[i] = 2*G[i-1] - 2*G[i-3] + G[i-4];
         }
         else {
           G[i] = G[i-1] + 3*G[i-2] - 5*G[i-3] + 2*G[i-4];
         }
       i++;
       }
     return G[n];
     }
 }
}
```

14.

Note that the strong induction format used is from the lecture notes (Eberly, 2019).

Let the expression of the n^{th} Grindelwald number be: n+1

Basis:

Let n = 0,

$G_0 = 1$

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

Then $G_n = n + 1 = 0 + 1 = 1$
LHS = RHS, therefore f(n) is true.

Let n = 1,
$G_1 = 2$
Then $G_n = n + 1 = 1 + 1 = 2$
LHS = RHS, therefore f(n) is true.

Let n = 2,
$G_2 = 3$
Then $G_n = n + 1 = 2 + 1 = 3$
LHS = RHS, therefore f(n) is true.

Let n = 3,
$G_3 = 4$
Then $G_n = n + 1 = 3 + 1 = 4$
LHS = RHS, therefore f(n) is true.

Inductive Step: Let k be an arbitrarily chosen integer such that $k \geq 4$.
Inductive Hypothesis:
Assume $G_n = n + 1$ for any integer n such that $0 \leq n \leq k$

Inductive Claim: We want to prove that $G_{k+1} = k+2$
Proof:

Case 1: If n+1 is even:
$G_{n+1} = 2G_{k+1-1} - 2G_{k+1-3} + G_{k+1-4}$
By the inductive hypothesis since $n \leq k$, we can assume n = k
$G_{n+1} = 2G_{n+1-1} - 2G_{n+1-3} + G_{n+1-4}$
$= 2G_n - 2G_n-2 + G_n-3$
$= 2(n+1) -2(n-1) + (n-2)$ by IH
$= 2n + 2 -2n +2 + n - 2$
$= 3n - 2n + 4 -2$
$= n + 2$
$= k + 2$

Case 2: If n+1 is odd:
$G_{k+1} = G_{k+1-1} + 3G_{k+1-2} - 5G_{k+1-3} + 2G_{k+1-4}$
By the inductive hypothesis since $n \leq k$, we can assume n = k

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

$G_{n+1} = G_{n+1-1} + 3G_{n+1-2} - 5G_{n+1-3} + 2G_{n+1-4}$

$= n+1 + 3(n) - 5 (n-1) + 2(n-2)$ by IH

$= n+1 +3n - 5n +5 +2n -4$

$= 6n - 5n + 6 - 4$

$= n +2$

$= k + 2$

Since the value of these 2 cases are both k+2 , this establishes the inductive claim and completes the proof.

30041703: Nguyen, Leslie
30042040: Tran, Jordan
30054457: Yang, Jeffery

References (APA6 Citation)

Eberly, W. (2019, May 7). Computer Science 331 Introduction and Mathematics Review [Lecture #1] . Retrieved from University of Calgary
D2L site: https://d2l.ucalgary.ca

Eberly, W. (2019, May 7). Computer Science 331 Introduction to Correctness of Algorithms I:Correctness of Simple Recursive Algorithms [Lecture #2] . Retrieved from University of Calgary
D2L site: https://d2l.ucalgary.ca

Eberly, W. (2019, May 9). Computer Science 331 Introduction to the Correctness of Algorithms II: Simple Algorithms with a While Loop [Lecture #3] . Retrieved from University of Calgary
D2L site: https://d2l.ucalgary.ca

Eberly, W. (2019, May 14). Computer Science 331 Analyzing the Running Times of Algorithms [Lecture #5] . Retrieved from University of Calgary
D2L site: https://d2l.ucalgary.ca