# Assignment 1 - The Hufflepuff Hierarchy

Betty Zhang - 30040611
William Chan - 30041834
Umair Hassan - 30047693
CPSC 331

October 5, 2018

1. Give a reasonably short proof that the function $f(n) = n$ is a bound function for this recursive algorithm.

**Proof:** Consider the recursive algorithm sHuff and the function $f(n) = n$:

1. Since $n$ is an integer input (given the precondition of the algorithm), $f(n) = n$ is an integer-valued function as required.

2. The algorithm is only calls itself recursively at line 9. At line 9, the input $n$ is replaced by the inputs $(n-1), (n-2), (n-3)$, and $(n-4)$. Since the value of the input $n$ is decreasing each time the recursion is called, the value of the function is decreased by at least one every time the algorithm is called recursively.

3. The precondition of the algorithm notes that the input n is a non-negative integer, therefore $n \geq 0$. It follows that it is only possible that $f(n) \leq 0$ under the conditions if $n = 0$. In this case, the test at line 1 passes, followed by the execution of line 2, where the execution ends immediately after so that the algorithm does not calls itself recursively. The function $f(n) = n$ is, therefore, a bound function of the recursive algorithm sHuff, because it satisfies all the conditions included in the definition bound function for a recursive algorithm.

2. Prove the following claim: for every non-negative integer n, if the sHuff algorithm is executed with n as input then this execution of the algorithm eventually ends, and the nth Hufflepuff number, $H_n$, is returned as output.

**Proof:** We prove that the sHuff algorithm satisfy the above claim by induction on $n$. The strong form of mathematical induction will be used, and the cases that $n = 0, n = 1, n = 2$, and $n = 3$ will be considered as basis.
**Basis:** Suppose $n = 0$ is the input on the execution of the algorithm sHuff, then the test at line 1 succeeds. The execution continues with line 2, causing the execution to terminate with the value $H_0 = 10$ being returned as output - as required by this case.

Suppose $n = 1$ is the input on the execution of the algorithm sHuff, the test at line 1 and the execution continues with the test at line 3. The test at line 3 succeeds, and the execution continues with line 4, where the execution terminates with the first Hufflepuff number, $H_1 = 9$ returned as an output - as required by this case.
Suppose $n = 2$ is the input on the execution of the algorithm sHuff, the test at line 1 and the execution continues with the test at line 3. The test also fails at line 3 and the execution continues with the test at line 5. The test at line 5 succeeds, and the execution continues with line 6 where the execution terminates with the 2nd Hufflepuff number, $H_2 = 8$ returned as an output - as required by this case.
Suppose $n = 3$ is the input on the execution of the algorithm sHuff, the test at line 1 and the execution continues with the test at line 3. The test also fails at line 3 and the execution continues with the test at line 5. The test then fails at line 5 and the execution continues with the test at line 7. The test at line 7 succeeds, and the execution continues with line 8, where the execution terminates with the 3rd Hufflepuff number, $H_3 = 7$ returned as an output - as required by this case.
**Inductive Step:** Let k be an integer such that $k \geq 3$, it is necessary and sufficient to use the following:

Inductive hypothesis: Suppose $n$ is an integer such that $0 \leq n \leq k$. If the algorithm sHuff is executed given the input $n$ as the input, then this execution of the algorithm eventually terminates, with the nth Hufflepuff number, $H_n$ returned as output.

To prove the following

Inductive claim: If the sHuff algorithm is executed with $n = k + 1$ as input, then this execution of the algorithm eventually ends, with k+1st Hufflepuff number $H_k+1 = H_n$ returned as output.

With that noted, suppose of the sHuff algorithm with the integer $n = k + 1$ as input. Since $k$ is an integer such that $k \geq 3$, n is an integer such that $n \geq 4$.

- Since $n \geq 4$, the tests at line 1, line 3, line 5, and line 7 fails, the execution of the algorithm continues at line 9.
- Line 9 includes a recursive of the algorithm with the input $n - 1$. Since $n - 1 = k$, and $0 \leq n - 1 \leq k$, it follows by the Inductive Hypothesis that the execution of the algorithm eventually ends, with $H_{n-1} = H_k$ returned as output
- Line 9 also includes a recursive execution of this algorithm with input $n - 2$. Since $n = k + 1 \geq 4$, $0 \leq n - 2 = k - 1 \leq k$, it follows by Inductive Hypothesis that the execution of the algorithm eventually ends, with $H_{n-2} = H_{k-1}$ returned as output
- Line 9 also includes a recursive execution of this algorithm with input $n - 3$. Since $n = k + 1 \geq 4$, $0 \leq n - 3 = k - 2 \leq k$, it follows by Inductive Hypothesis that the execution of the algorithm eventually ends, with $H_{n-3} = H_{k-2}$ returned as output
- Line 9 also includes a recursive execution of this algorithm with input $n - 4$. Since $n = k + 1 \geq 4$, $0 \leq n - 4 = k - 3 \leq k$, it follows by Inductive Hypothesis that the execution of the algorithm eventually ends, with $H_{n-4} = H_{k-3}$ returned as output

- One can now see, by inspection of line 9, that this execution of algorithm eventually ends. Furthermore, since $k+1 \geq 4$, it follows by the definition of $H_{k+1}$, that the value returned as output is:
  $4H_{n-1} - 6H_{n-2} + 4H_{n-3} - H_{n-4} = 4H_k - 6H_{k-1} + 4H_{k-2} - H_{k-3} = H_{k+1}$
  As required to established the inductive claim.

The result now follows by induction on n.

3. Complete a proof that sHuff algorithm correctly solves the Hufflepuff Hierarchy computation problem.

- Part 2 of this assignment has provided a proof for the following claim: for every non-negative integer $n$, if the sHuff algorithm is executed with n as input then this execution of the algorithm eventually ends, and the nth Hufflepuff number, $H_n$, is returned as output.
- For the algorithm to be correct, we also need to check for undocumented side-effects: One can see by inspection of the code of the algorithm sHuff that the following properties hold:
  - The only input accessed is the input $n$ that is mentioned in the precondition, no global data is accessed
  - The algorithm does not modify any input or global data and only output returned is the Hufflepuff number $H_n$ mentioned in the above problem postcondition

  Thus we have proved that this algorithm has no undocumented side-effects.

4. Write a Java program SHufflepuff
The SHufflepuff program is submitted seperately in the dropbox

5. Give a recurrence for the number $T_{sHuff}(n)$ of steps used by the sHuff algorithm (from Figure 1) when it is executed with a non-negative integer input $n$. Please use the uniform. cost criterion to determine this, counting the steps that are numbered in the algorithm.

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

Explanation:
The algorithm executes two steps (at lines 1 and 2) if it is executed when n = 0.
The algorithm executes three steps (at lines 1, 3 and 4) if it is executed when n = 1.
The algorithm executes four steps (at lines 1, 3, 5 and 6) if it is executed when n = 2.

The algorithm executes five steps (at lines 1, 3, 5, 7 and 8) if it is executed when n = 3. The algorithm executes five steps (at lines 1, 3, 5, 7 and 9) if it is executed, but it also calls itself recursively at line 9 when $n \geq 4$. The algorithm has 4 recursions with the inputs $n-1, n-2, n-3$ and $n-4$. So the recurrence of the algorithm if $n \geq 4$ is $T_{sHuff}(n-1) + T_{sHuff}(n-2) + T_{sHuff}(n-3) + T_{sHuff}(n-4) + 5$.

6. Prove that $T_n(n) \geq (\frac{3}{2})^n$

**Proof:** Proving by mathematical induction on $n$

**Basis:**

Let $n = 0$. Then $T_{sHuff}(n) = T_{sHuff}(0) = 2 \geq 1 = \frac{3}{2}^0 = \frac{3}{2}^n$ as required by the claim.

Let $n = 1$. Then $T_{sHuff}(n) = T_{sHuff}(1) = 3 \geq \frac{3}{2} = \frac{3}{2}^1 = \frac{3}{2}^n$ as required by the claim.

Let $n = 2$. Then $T_{sHuff}(n) = T_{sHuff}(2) = 4 \geq \frac{9}{4} = \frac{3}{2}^2 = \frac{3}{2}^n$ as required by the claim.

Let $n = 3$. Then $T_{sHuff}(n) = T_{sHuff}(3) = 5 \geq \frac{27}{8} = \frac{3}{2}^3 = \frac{3}{2}^n$ as required by the claim.

**Inductive step:** Let $k$ be an integer such that $k \geq 3$.

Inductive Hypothesis: Let the statement $S(n)$ be $T_{sHuff}(n) \geq \frac{3}{2}^n$. Assume $S(n)$ is true for all integers $0 \leq n \leq k$.

Inductive Claim: We will prove that$S(k+1) \geq \frac{3}{2}^{k+1}$.

Note that since $k \geq 3$, $k - 3 \geq 0$. Now we have:

$$
\begin{aligned}
T_{sHuff}(k+1) &= T_{sHuff}(k) + T_{sHuff}(k-1) + T_{sHuff}(k-2) + T_{sHuff}(k-3) + 5 \\
&\geq \frac{3}{2}^k + T_{sHuff}(k-1) + T_{sHuff}(k-2) + T_{sHuff}(k-3) + 5 \\
&\text{(by Inductive Hypothesis)} \\
&\geq \frac{3}{2}^k + \frac{3}{2}^{k-1} + \frac{3}{2}^{k-2} + \frac{3}{2}^{k-3} + 5 \\
&= \frac{3}{2}^{k-3}(\frac{3}{2}^3 + \frac{3}{2}^2 + \frac{3}{2} + 1) + 5 \\
&\text{(Since} \frac{3}{2}^3 + \frac{3}{2}^2 + \frac{3}{2} + 1 = \frac{65}{8}) \\
&= (\frac{65}{8})(\frac{3}{2}^{k-3}) + 5 \\
&\text{(Then we divide } \frac{3}{2} \text{ out of the constant 4 times)} \\
&= (\frac{3}{2})(\frac{65}{12})(\frac{3}{2}^{k-3}) + 5 \\
&= (\frac{3}{2})(\frac{3}{2})(\frac{3}{2})(\frac{3}{2})(\frac{130}{81})(\frac{3}{2}^{k-3}) + 5 \\
&= (\frac{130}{81})(\frac{3}{2}^{k+1}) + 5 \\
&\text{(subtracting 5 would equal less than before)} \\
&\geq (\frac{130}{81})(\frac{3}{2}^{k+1}) \\
&(\frac{130}{81} is \geq 1 \text{ , so dividing it out would make it equal less than before)} \\
&\geq \frac{3}{2}^{k+1}
\end{aligned}
$$

Therefore, $T_{sHuff}(k+1) \geq (\frac{3}{2}^{k+1})$ and follows from S(n) where $0 \leq n \leq k$
By mathematical induction $T_n(n) \geq (\frac{3}{2})^n$

7. Give a loop invariant for the while loop at lines 1420 of this algorithm, and show that your answer is correct.
Loop Invariant:

- $n$ is an integer input such that $n \geq 4$
- $i$ is an integer variable such that $3 \leq i \leq n$.
- Hocus is an integer variable with the value of the $(i-3)^{th}$ Hufflepuff number, $H_{i-3}$
- Pocus is an integer variable with the value of the $(i-2)^{th}$ Hufflepuff number, $H_{i-2}$
- Abra is an integer variable with the value of the $(i-1)^{th}$ Hufflepuff number, $H_{i-1}$
- Kadabra is an integer variable with the value of the $(i)^{th}$ Hufflepuff number, $H_i$

**Proof:**
Loop theorem 1 will be sued to prove that the above loop invariant is correct.

(a) Since the loop test (at line 6) of the algorithm is a simple comparison, the loop test has no side effects.

(b) Consider the precondition for the Hufflepuff Hierarchy Computation problem satisfied at the beginning of execution of the algorithm.
Given the precondition $n \geq 0$, if $0 \leq n \leq 3$, then the claim is trivial - the execution of the algorithm will succeed at one of the tests at line 1,3,5, or 7, and the algorithm will end after the execution of the line after the test that has succeeded. In the case where $0 \leq n \leq 3$, the while loop is never reached. Instead, suppose that $n \leq 4$. Since there is the value of $n$ is never affected (changed) by any of the lines in the algorithm, the first part of the proposed loop invariant is satisfied if the while loop is ever reached (Indeed, the while loop is reached after the execution of line 1, 3, 5 and 7). One can see by inspection of the code that the while loop is never reached again, so it suffices to consider the first execution of the loop.
The variable $i$ is declared at line 13 to be an integer variable with the value 3. Since $3 = i < n$ when the execution of the loop begins, the second part of the proposed loop invariant is satisfied when the loop is reached as well.
Since hocus is an integer variable whose value is set at line 9 to be $10 = H_0 = H_{i-3}$, the third proposed loop invariant is satisfied when the execution of the loop begins.
Since pocus is an integer variable whose value is set at line 10 to be $9 = H_1 = H_{i-2}$, the fourth proposed loop invariant is satisfied when the execution of the loop begins.
Since abra is an integer variable whose value is set at line 11 to be $8 = H_2 = H_{i-1}$, the fifth proposed loop invariant is satisfied when the execution of the loop begins.
Since kadabra is an integer variable whose value is set at line 12 to be $7 = H_3 = H_i$ , the sixth proposed loop invariant is satisfied when the execution of the loop begins.

(c) Consider an execution of the body of the while loop that begins with the proposed loop invariant satisfied.
It follows by part 1 of the proposed loop invariant that $n$ is an integer input such that $n \geq 4$. Since the value of n is not changed during the execution of lines 15-20, it is still true that n is an integer input such that $n \leq 4$ at the end of this execution of the body of the loop.
It follows by part 2 of the loop invariant that $i$ is an integer variable such that $3 \leq i \leq n$. Since the test at line at line 14, $i < n$, was checked and passed, this mean $3 \leq i \leq n-1$ (since $i$ and $n$ both have integer values) at this point. The value of n is never changed and the value of $i$ is increased by one during the execution of line 20 so that $4 \leq i \leq n$ at the end of this execution of the loop body, therefore the second part of the proposed loop invariant is satisfied at the end of this execution of the loop body.
Base on the proposed loop invariant and the proof in part (b), the integers hocus, pocus, abra and kadabra are equal to $H_{i-3}, H_{i-2}, H_{i-1}$, and $H_i$ respectively at the beginning of the loop. After the execution of line 15, the integer variable shazam is created such that shazam $= 4 * H_i - 6 * H_{i-1} + 4 * H_{i-2} - H_{i-3} = H_{i+1}$. Moreover, hocus = pocus $= H_{i-2}$ immediately after the execution of line 16. Next, pocus = abra $= H_{i-1}$ after the immediately execution of line 17. Next, abra = kadabra = Hi immediately after the execution of line 18 and kadabra = shazam $= H_{i+1}$ immediately after the execution of line 19. However, since $i$ is incremented at line 20, hocus $= H_{i-3}$, pocus $= Hi - 2$, abra $= H_{i-1}$, and kadabra $= H_i$ again at this execution of the loop body, re-establishing part 3 to 6 of the proposed loop invariant.Thus the proposed loop invariant is satisfied, once again, at the end of this execution of the loop body.

It follows by loop Theorem 1 that the loop invariant listed above are indeed loop invariant for this loop, as claimed.

8. Use the loop invariant that you gave, when answering the previous question, to prove that this algorithm is partially correct.

One can see by inspection of code that this algorithm has no undocumented side-effects, it is therefore suffices to show that if this algorithm is executed, when the precondition is satisfied, then either:

(a) The execution of the algorithm halts and the postconditions for the Hufflepuff Hierarchy Computation is satisfied, or

(b) The execution of this algorithm never halts at all.

With this noted, consider an execution of this algorithm when the precondition of the problem is satisfied: so that n is an integer input such that $n \geq 0$.

- If $n = 0$, then this execution ends after the execution of lines 1 and 2, with the value $10 = F_0 = F_n$ returned as output, thus the postcondition is satisfied. Condition (a) in the definition of partial correctness has been satisfied.

- If $n = 1$, then this execution ends after the execution of lines 1, 3, and 4, with the value $9 = F_1 = F_n$ returned as output, thus the postcondition is satisfied. Condition (a) in the definition of partial correctness has been satisfied.

- If $n = 2$, then this execution ends after the execution of lines 1, 3, 5, and 6, with the value $8 = F_2 = F_n$ returned as output, thus the postcondition is satisfied. Condition (a) in the definition of partial correctness has been satisfied.

- If $n = 3$, then this execution ends after the execution of lines 1, 3, 5, 7 and 8, with the value $7 = F_3 = F_n$ returned as output, thus the postcondition is satisfied. Condition (a) in the definition of partial correctness has been satisfied.

- If $n \geq 1$, then lines 1, 3, 5, 7, 9, 10, 11, 12, 13 are executed, then the loop starting at line 14 is reached and executed. Either the execution of the loop terminates, or it does not

  - If the execution of the loop terminates, then the loop invariant stated in the previous question is satisfied at this point. So $i$ is an integer variable and $n$ is an integer input such that $3 \leq i \leq n$ (loop invariant 1 and 2). In order for the execution of the loop to end, the test at line 14 must have been checked and failed, so it must be that $i \geq n$ : given this and the loop invariant, $i = n$ at the end of the execution of the while loop.
  - It follows, in this case, that the execution of the algorithm ends with the execution of line 21, at which point the value of Kadabra $= H_i$ is returned as output. Given the loop invariant, Kadabra $= H_i$, and since $i = n$ at this point, the output Kadabra $= H_i = H_n$ satisfies the problems postcondition, as needed to establish condition (a) of the claim.
  - On the other hand, if the execution of the loop does not end, then the execution of the algorithm certainly does not either as needed to establish partial correctness in this case, as well because condition (b) in the definition of partial correctness has been satisfied.

Since either condition (a) or condition (b) is satisfied in every possible case, it follows that the algorithm is partially correct, as claimed.

9. Give a bound function for the while loop in this algorithm and show that your answer is correct.
**Claim:** A bound function for the while loop in this Hufflepuff is: $h(i, n) = n - i$, under the definition of a bound function for a while loop defined in lecture.
**Proof:** Consider the above function $h$.

- Since $n$ is an integer input and $i$ is an integer variable defined and initialized at line 6, the function $h$ is an integer-valued total function of some of the inputs, variables, and global data that are defined when the while loop in the Hufflepuff algorithm is reached.

- When the body of the while loop is executed, the value of $i$ is increased by one in line 20, and the value of $n$ is not changed, so the value of $h$ is decreased by one

- If the value of $h$ is less than or equal to zero, then $n - i \leq 0$, so $i \geq n$. If $i \geq n$, the loop test at line 14 would fail when checked.

If follows, by the definition of a bound function for a while loop, that h is a bound function for the while loop in the Hufflepuff algorithm, as claimed.

10. Use the above to complete a proof that this algorithm correctly solves the Hufflepuff Hierarchy Computation problem.

It follows from the definitions in these notes that if an algorithm for a given computational problem is both partially correct and terminates, whenever it is executed when its problems precondition initially satisfied, then the algorithm is correct. The answer in question 8 provided a proof of the hufflepuff algorithm partial correctness, now we must prove the termination of the algorithm.
Loop theorem 2 will be used to prove the termination of this algorithm:
Consider an execution of the hufflepuff algorithm when the precondition for the Hufflepuff Hierarchy Computation problem is satisfied. A non-negative integer n has been given as input.
If $n = 0$ then the test at line 1 is checked and passed, and the execution of the algorithm ends after the execution of line 2, as required to establish the claim.
If $n = 1$ then the test at line 1 is failed, so the execution of line 3 is passed and the execution of the algorithm ends after the execution of line4, as required to establish the claim.
If $n = 2$ then the test at line 1 and 3 are failed, so the execution of line 5 is passed and the execution of the algorithm ends after the execution of line 6, as required to establish the claim.
If $n = 3$ then the test at line 1,3 and 5 are failed, so the execution of line 7 is passed and the execution of the algorithm ends after the execution of line 8, as required to establish the claim.
If $n \geq 4$ then the tests at line 1, 3, 5, and 7 are failed, so that lines 9, 10, 11, 12 and 13 are reached, and the while loop is reached and executed.

- The loop test at line 14 certainly has no side-effects since it is a simple comparison of an integer input and variable, every execution of this test halts.

- The body of this loop includes only six lines and certainly terminates whenever it is executed.

- Question 9 confirms that this while loop has a bound function.

It now follows by Loop Theorem 2 that every execution of this while loop, during an execution of the algorithm starting with its problems precondition satisfied, will eventually end. The execution of the algorithm will also end - as needed to establish the claim - after an execution of line 21.
Since the algorithm is both partially correct and terminates, whenever it is executed when its problems precondition initially satisfied, then this algorithm is correct.

11. Write a Java program Hufflepuff
The program Hufflepuff.java in submitted separately in the dropbox.

12.Use your bound function (obtained when answering Question 9) to compute an upper-bound for the number $T_{Huff}(n)$ on the number of steps executed by the Hufflepuff algorithm on a non-negative input n.

Consider an execution of the algorithm when the precondition of the algorithm is met:
When $n = 0$, line 1 and 2 are executed and the algorithm halts. Since the Uniform Cost Criterion" is being used here, the number of steps executed is 2 in this case.
when $n = 1$, line 1, 3, and 4 are executed and the algorithm halts. the number of steps executed is 3 in this case.
when $n = 2$, line 1, 3, 5 and 6 are executed and the algorithm halts. the number of steps executed is 4 in this case.
when $n = 3$, line 1, 3, 5, 7and 8 are executed and the algorithm halts. the number of steps executed is 5 in this case.
The bound function was $ni$, so the loop body is executed at most $ni$ times and the loop test is executed at most $n - i + 1$ times. Since the the value of $i$ is to assigned to the value 3 at line 13, the initial value of the bound function is $n - i = n - 3$. The loop body has 6 steps (line 15 to 20), so the loop body is executed as most $6(n - 3)$times. We also have $n - i + 1 = n - 2$ for the maximum number of execution of the loop test. Therefore the maximum number of execution of the loop is $6(n - 3) + n - 2 = 7n - 20$. Now we add the run time for the rest of the code, there are 9 lines of code that has to be executed before reaching the loop (line 1, 3 , 5, 7, 9, 10, 11, 12 and 13) and 1 line to execute after the loop (line 21) . So $T_{Huff}(n)$ has an upper bound of $7n - 10$ when $n \geq 4$.
It follows that the number of steps included in an execution of this algorithm on a non-negative integer input n is:

$$T_{sHuff}(n) = \begin{cases} 2 & n = 0 \\ 3 & n = 1 \\ 4 & n = 2 \\ 5 & n = 3 \\ 7n - 10 & n \geq 4 \end{cases} \tag{2}$$

Bonus

Claim: Let $H_n$ is the Hufflepuff number of n as defined in the assignment
Let the statement $S_n$ be $H_n = 10 - n$ for all positive integers n.
Given $n$, the Hufflepuff number is $10 - n$ (in other words, $S_n$ is correct)

**Basis**:
If $n = 0$ it returns 10, which is 10 - 0, therefore $S_0$ is true
If $n = 1$ it returns 9, which is 10 - 1, therefore $S_1$ is true
If $n = 2$ it returns 8, which is 10 - 2, therefore $S_2$ is true
If $n = 3$ it returns 7, which is 10 - 3, therefore $S_3$ is true

**Inductive step:**
Let $k$ be an integer such that $k \geq 3$.
Inductive Hypothesis: Assume $S_n$ is true for all integers $0 \leq n \leq k$
Inductive Claim: $S_{k+1}$ is true (which means $H_{k+1} = 10 - (k+1)$)

$$\begin{aligned} H_{k+1} &= 4 * (H_k - 6 * (H_{k-1}) + 4 * (H_{k-2}) - H_{k-3} \\ &= 4 * (10 - k) - 6 * (10 - (k-1)) + 4 * (10 - (k-2))(10 - (k-3)) \\ &= 40 - 4k - 66 + 6k + 48 - 4k - 13 + k \\ &= 9 - k \\ &= 9 + 1 - 1 - k \\ &= 10 - k - 1 \\ &= 10 - (k+1). \end{aligned}$$

$H_{k+1} = 10 - (k+1)$
$S_{k+1}$ is true and follows from $S_n$ is true for all integers $0 \leq n \leq k$
Therefore by mathematical induction $H_n = 10 - n$ for all positive integers $n$.

**References**:
Lecture Notes 1 - 6
Tutorial Solutions 1 - 6
Assignment 1 Algorithms
http://edwinckc.com/cpsc331

```java
/*CPSC assignment 1
Umair Hassan 30047693
Betty Zhang 30040611
William Chan 30041834
*/

package cpsc331.A1;

public class Hufflepuff {
// Precondition: An integer n is given as input.
// Postcondition: If n >= 0 then the nth Hufflepuff number, Hn, is returned as output. An
IllegalArgumentException is thrown otherwise.
public static int eval(int n) {
        // Assertion: A non-negative integer n has been given as input.
        if(n>=0) {
                if (n == 0) {
                        return 10;
                } else if (n == 1) {
                        return 9;
                } else if (n == 2) {
                        return 8;
                } else if (n == 3) {
                        return 7;
                } else {
                        int hocus = 10;
                        int pocus = 9;
                        int abra = 8;
                        int kadabra = 7;
                        int i = 3;
                        // Loop Invariant:
                        // 1. n is an integer such that n >= 4.
                        // 2. i is an integer such that 3 <= i <= n.
                        // 3. Hocus is an integer variable with the value of the (i-3)th Hufflepuff
number.
                        // 4. Pocus is an integer variable with the value of the (i-2)th Hufflepuff
number.
                        // 5. Abra is an integer variable with the value of the (i-1)th Hufflepuff number.
                        // 6. Kadabra is an integer variable with the value of the (i)th Hufflepuff
number.
                        // Bound Function: n-i

                        while (i < n) {
                                int shazam = 4*kadabra - 6*abra + 4*pocus - hocus;
                                hocus = pocus;
                                pocus = abra;
                                abra = kadabra;
                                kadabra = shazam;
                                i += 1;
                        }
                        return kadabra;
                }
                // Assertion:
                // 1. A non-negative integer n has been given as input.
                // 2. The nth Hufflepuff number, Hn, has been returned as output.
        }
        else {
                throw new IllegalArgumentException("Silly muggle! The input integer cannot be negative.");
        }
        // Assertion:
        // 1. A non-negative integer n has been given as input.
        // 2. The nth Hufflepuff number, Hn, has been returned as output.
}
// The main method takes an integer input n as an argument in the command line.
// The method checks if a valid argument is present in the command line, it will throw
// an IllegalArgumentException if not. If a valid argument was given, proceed to call
// eval function and return the corresponding Hufflepuff number to user.
public static void main(String[] args) {
```

```
        Boolean IllegalArgument=false;    //Initiate a boolean to check for illegal argument
        if(args.length != 0) {    //If the argument length is not 0
                if (!args[0].matches("-?\\d+(\\.\\d+)?")) //If the first argument are numbers
                { //Set the boolean to true and throw an exception
                        IllegalArgument=true;
                        throw new IllegalArgumentException("Silly muggle! One integer input is
required.");
                }
        }else{  //If the number of arguments was 0, throw an exception
                IllegalArgument=true;
                throw new IllegalArgumentException("Silly muggle! One integer input is required.");
        }
        if(!IllegalArgument) {  //Check the boolean and if the boolean was false then proceed
                if (args.length == 1 && args[0].matches("\\d+")) { //If the number of arguments is 1 and
the first argument are digits
                        System.out.println(eval(Integer.parseInt(args[0]))); //Print the integer returned
from calling sHuffle method with first argument
                } else if (Integer.parseInt(args[0]) < 0) { //If the first argument is a negative
integer, print error
                        throw new IllegalArgumentException("Silly muggle! The input integer cannot be
negative.");
                } else { //Else throw an illegal argument exception
                        throw new IllegalArgumentException("Silly muggle! One integer input is
required.");
                }
        }
}
}
// References:
// eval function: CPSC 331 - Assignment #1 Proving the Correctness of Simple Algorithms - and
Implementing Them as Java Programs
```

```java
/*CPSC assignment 1
Umair Hassan 30047693
Betty Zhang 30040611
William Chan 30041834
*/

package cpsc331.A1;

public class SHufflepuff {
// Precondition: A non-negative integer n is given as input.
// Postcondition: The nth Hufflepuff number, Hn, is returned as output.
protected static int sHuffle(int n){
        // Assertion: A non-negative integer n has been given as input.
        if(n>= 0) {
                if (n == 0) {
                        return 10;
                } else if (n == 1) {
                        return 9;
                } else if (n == 2) {
                        return 8;
                } else if (n == 3) {
                        return 7;
                } else {
                        //Bound Function: n-i
                        return 4*sHuffle(n - 1) - 6*sHuffle(n - 2) + 4*sHuffle(n - 3)
- sHuffle(n - 4);
                }
                // Assertion:
                // 1. A non-negative integer n has been given as input.
                // 2. The nth Hufflepuff number, Hn, has been returned as output.
        }
        else {
                throw new IllegalArgumentException("Silly muggle! The input integer
cannot be negative.");
        }
        // Assertion:
        // 1. A non-negative integer n has been given as input.
        // 2. The nth Hufflepuff number, Hn, has been returned as output.
}
// The main method takes an integer input n as an argument in the command line.
// The method checks if a valid argument is present in the command line, it will throw
// an IllegalArgumentException if not. If a valid argument was given, proceed to call
// sHuffle function and return the corresponding Hufflepuff number to user.
public static void main(String[] args) {
        Boolean IllegalArgument=false;    //Initiate a boolean to check for illegal
argument
        if(args.length != 0) {  //If the argument length is not 0
                if (!args[0].matches("-?\\d+(\\.\\d+)?")) //If the first argument are
numbers
                {       //Set boolean to true and throw exception
                        IllegalArgument=true;
                        throw new IllegalArgumentException("Silly muggle! One integer
input is required.");
                }
        }else{  //If the number of arguments was 0, set boolean to true and throw
exception
                IllegalArgument=true;
                throw new IllegalArgumentException("Silly muggle! One integer input
is required.");
        }
        if(!IllegalArgument) {  //Check the boolean and if the boolean was false then
proceed
```

```
                if (args.length == 1 && args[0].matches("\\d+")) { //If the number of
arguments is 1 and the first argument are digits
                        System.out.println(sHuffle(Integer.parseInt(args[0]))); //
Print the integer returned from calling sHuffle method with first argument
                } else if (Integer.parseInt(args[0]) < 0) { //If the first argument
is a negative integer, print error
                        throw new IllegalArgumentException("Silly muggle! The input
integer cannot be negative.");
                } else { //Else throw an illegal argument exception
                        throw new IllegalArgumentException("Silly muggle! One integer
input is required.");
                }
        }
}
}
// References:
// sHuffle function: CPSC 331 - Assignment #1 Proving the Correctness of Simple
Algorithms - and Implementing Them as Java Programs
```