A Running Time Analysis of Horner's Rule

The following code fragment implements Horner's Rule for numerical solution of a polynomial of degree *n*:

```
1    y = 0;
2    i = n;
3    while (i >= 0) {
4         y = a[i] + x * y;
5         i = i - 1;
6    }
```

Following the process described on pages 24 and 25 of the textbook, the table below shows the cost and number of times that each step in the code fragment takes:

Line	Statement	Cost	Times
1	`y = 0`	c_1	1
2	`i = n`	c_2	1
3	`while (i >= 0) {`	c_3	n + 2
4	`y = a[i] + x * y`	c_4	n + 1
5	`i = i – 1`	c_5	n + 1
6	`}`	0	---

Lines 1 and 2 are straightforward assignments.

Line 3, when combined with the decrement operation in line 5, provides a loop that runs from `n` down to 0. In addition, `i` is tested one more time than the loop actually runs.

Lines 4 and 5 run the same number of times as the loop itself, as described for line 3.

Line 6 is merely a loop semantic that transfers control back to line 3

Adding the cost of all statements gives the following equation:

$$T(n) = c_1(1) + c_2(1) + c_3(n+2) + c_4(n+1) + c_5(n+1) + 0$$
$$T(n) = (c_1 + c_2 + 2c_3 + c_4 + c_5) + (c_3 + c_4 + c_5)n$$

Combine constants:

let: $a = c_3 + c_4 + c_5$
and: $b = c_1 + c_2 + 2c_3 + c_4 + c_5$

Then:

$$T(n) = an + b$$

This running time is directly proportional to *n*. In addition, it can be bounded on both the upper and lower sides by proper selection of arbitrary constants. Therefore, the asymptotic running time of this code fragment for Horner's Rule is:

$$T(n) = \Theta(n)$$