2 Polynomials (22 points)

An *Integer Polynomial* is an expression of the form

$$a_n x^n + \cdots + a_2 x^2 + a_1 x + a_0$$

where all a_i, $0 \leq i \leq n$, are integers. A *nonzero term* of a polynomial is a single $a_i x^i$ with $a_i \neq 0$. The *degree* of a polynomial is the largest exponent n associated with a nonzero term, or 0 for the polynomial 0. Consider the `Polynomial` class below, which represents an integer polynomial as a linked list of terms:

```
public class Polynomial {
    private class Term {
        public int coef;
        public int exp;
        public Term next;

        public Term(int coef, int exp) {
            this.coef = coef;
            this.exp = exp;
        }
    }

    private Term terms;

    public Polynomial() {
        terms = new Term(0, -1);
    }

    public double eval(double input) {
        // Implemented below
    }

    public void addTerm(int coef, int exp) {
        // Implemented below
    }
}
```

Note the following:

- The special term $(0, -1)$ is used as a sentinel value, which is placed at the **end** of the linked list.
- The linked list contains only the nonzero terms of the polynomial, ordered from greatest exponent to least exponent. Thus, every polynomial has a unique representation.
- The linked-list topology is not circular; the sentinel term always has a `next` of `null`.

For example, the polynomial $-3x^{500} + 5x^3 + 4x + 7$ has four nonzero terms, and has degree 500; it is represented by the linked list $(-3, 500) \to (5, 3) \to (4, 1) \to (7, 0) \to (0, -1)$.

The polynomial 0 has no nonzero terms and has degree 0; it is represented by the linked list $(0, -1)$.

The `Polynomial` class, reprinted for your convenience:

```
public class Polynomial {
    private class Term {
        public int coef;
        public int exp;
        public Term next;

        public Term(int coef, int exp) { ... }
    }

    private Term terms;

    public Polynomial() {
        terms = new Term(0, -1);
    }
}
```

(a) (7 points) Implement the `eval` method, which receives a value `input` and returns the result of evaluating the polynomial when $x =$ `input`.

For example, the polynomial $5x^3 + 4x + 7$ evaluated at the input 2 would be $(5 \cdot 2^3) + (4 \cdot 2) + (7) = 55$. If instead, this polynomial is evaluated at 0.5, the result would be $(5 \cdot 0.5^3) + (4 \cdot 0.5) + (7) = 9.625$.

```
public double eval(double input) {
    double result = 0;

    _____1_____;

    while (_____2_____) {

        result += _____3_____ Math.pow(_____4_____, _____5_____);

        _____6_____;
    }

    _____7_____;
}
```

(b) (2 points) What is the asymptotic runtime of `eval`, in terms of n, the degree of the polynomial? Provide the most informative bound possible, in either Θ, Ω, or O notation.

(c) (11 points) Implement the `addTerm` method, which receives a coefficient and exponent representing a nonzero term, and updates the polynomial to include that term.

For example, the polynomial $5x^3 + 4x + 7$, after adding $2x^2$, would become the polynomial $5x^3 + 2x^2 + 4x + 7$, with the underlying linked list $(5, 3) \to (2, 2) \to (4, 1) \to (7, 0) \to (0, -1)$.

You may assume that `coef != 0`, `exp >= 0`, and that the polynomial does not already contain a nonzero term with the same exponent.

```
public void addTerm(int coef, int exp) {

    Term newTerm = _____1_____;

    if (_____2_____) {

        _____3_____;

        _____4_____;
        return;

    }
    Term curr = _____5_____;

    while (_____6_____) {

        _____7_____;
    }

    _____8_____;

    _____9_____;
}
```

(d) (2 points) What is the asymptotic runtime of `addTerm`, in terms of n, the degree of the **original** polynomial, and k, the exponent of the new term? Provide the most informative bound possible, in either Θ, Ω, or O notation.