1. Implement following functions on **Single Linked List Template Class,** which has **head** as well as **tail**.
    a. Constructor, copy constructor, destructor, assignment operator
    b. void InsertAtHead(const T & el), void InsertAtTail(const T & el), void InsertAtKth(int K, const T & el)
    c. void deleteAtHead(), void deleteAtTail(), void deleteAtKth(int K)
    d. Node<T> * findKth(int K), Node<T> * findMid(), Node<T> * findKthFromEnd()
    e. int getLength()
    f. void sort() – Sort using merge sort. You might need to create private static helper functions for the same.
    g. Void reverse() – Reverse a linked list, iterative.
    h. Void clear() – Delete Linked List
    i. Bool isEmpty() – Check If Linked List is Empty
    j. void operator+=(const T ele) – append ele at the end of the linkedlist
    k. void operator+=(const LinkedList &) – append all elements of the passed linkedlist to the current. You need to create new nodes for all the elements in the passed linked list.
    l. LinkedList operator+(const LinkedList &) – Return a new linked list which is same as appending the current and the passed as argument.
    m. friend ostream& operator<<(ostream &, const Linkedlist &) – Friend function to print the list.
2. Create a **Polynomial class** with following functions. You need to create a separate node class for it which has three parts – coeff, power, and address to next node.
    a. Constructor, Destructor, Copy Constructor, Assignment Operator
    b. insertTerm (int power, int coef)
    c. Polynomial operator+(const Polynomial &)
    d. Polynomial& operator+=(const Polynomial &)
    e. Polynomial operator-(const Polynimial &)
    f. Polynomial & operator-=(const Polynomial &)
    g. Polynomial operator*(const Polynomial &)
    h. Polynomial& operator *=(const Polynomial &)
    i. Polynomial& operator~()
    j. friend ostream& operator<<(ostream &, const Polynomial &)