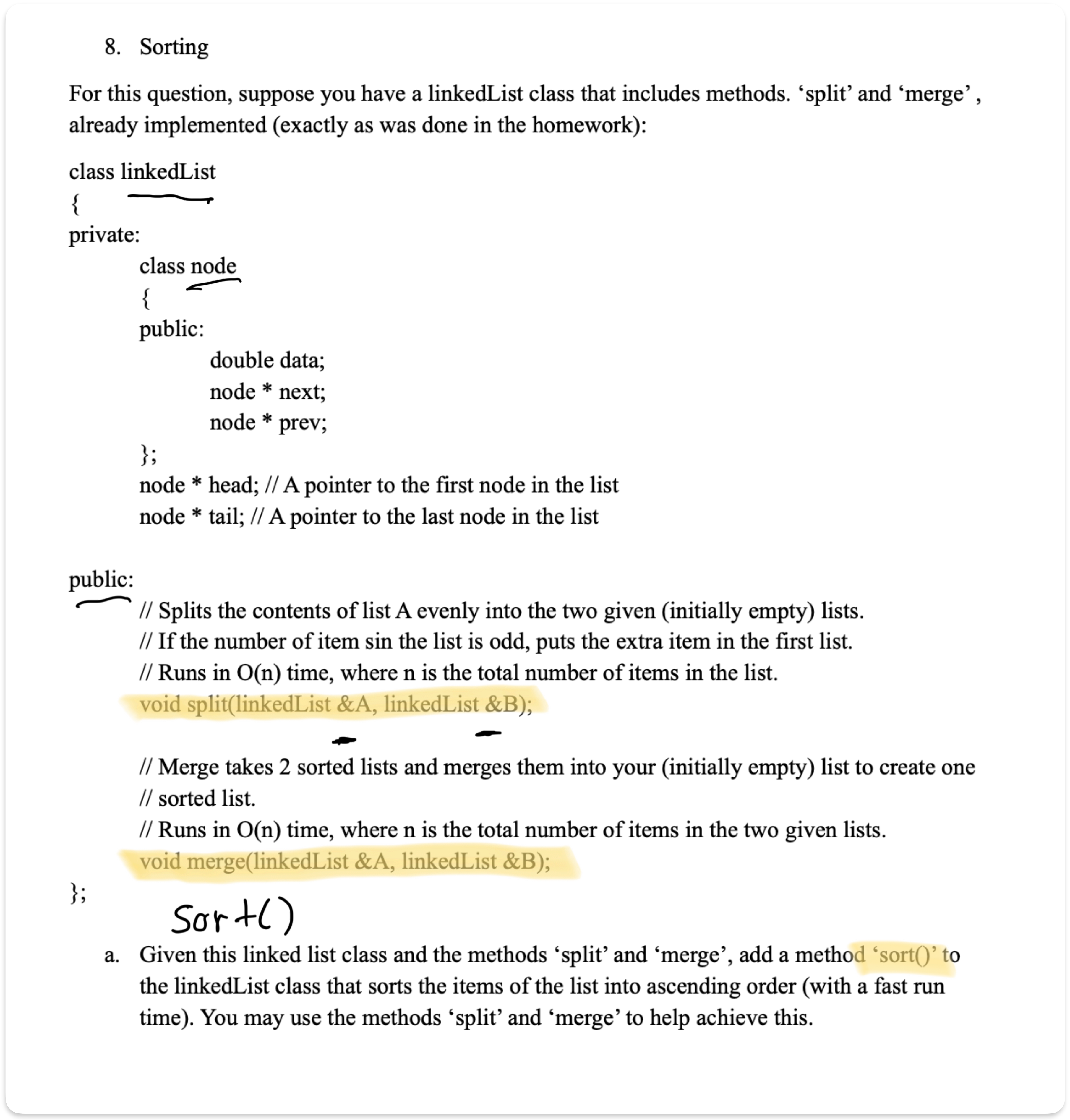
Question 2: Merge sort



8. Sorting

For this question, suppose you have a linkedList class that includes methods. 'split' and 'merge' , already implemented (exactly as was done in the homework):

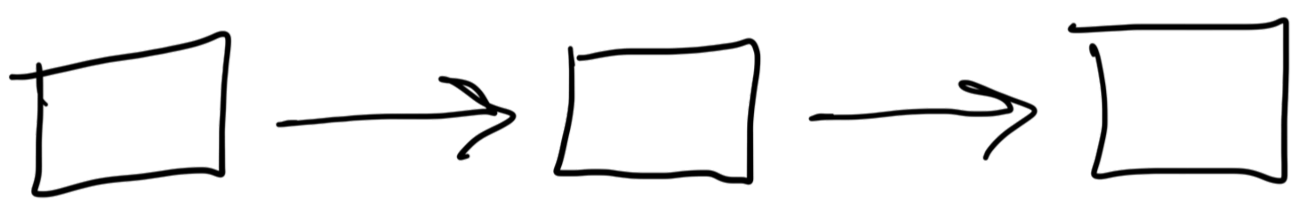
```
class linkedList
{
private:
        class node
        {
        public:
                double data;
                node * next;
                node * prev;
        };
        node * head; // A pointer to the first node in the list
        node * tail; // A pointer to the last node in the list

public:
        // Splits the contents of list A evenly into the two given (initially empty) lists.
        // If the number of item sin the list is odd, puts the extra item in the first list.
        // Runs in O(n) time, where n is the total number of items in the list.
        void split(linkedList &A, linkedList &B);

        // Merge takes 2 sorted lists and merges them into your (initially empty) list to create one
        // sorted list.
        // Runs in O(n) time, where n is the total number of items in the two given lists.
        void merge(linkedList &A, linkedList &B);
};
```

Sort()

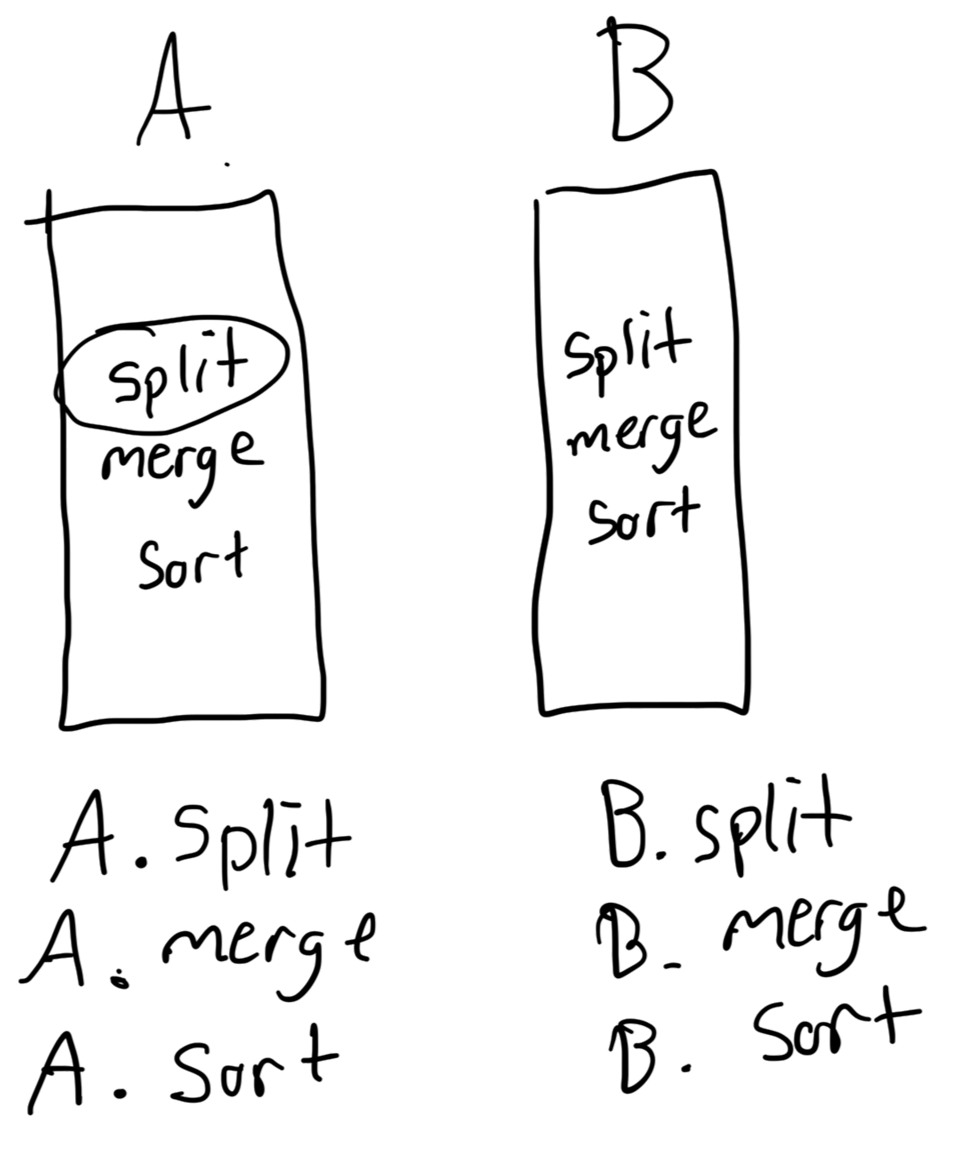
a. Given this linked list class and the methods 'split' and 'merge', add a method 'sort()' to the linkedList class that sorts the items of the list into ascending order (with a fast run time). You may use the methods 'split' and 'merge' to help achieve this.



```
void sort() {                       // O(1)
    // if (nullptr) → false
    if (!head || !head->next)
        return;                     // O(1)
    linkedList<T> A, B;
    split(A, B);                    // O(n)
    A.sort();                       //
    B.sort();                       //   2T(n/2)
    merge(A, B);                    // O(n)
}
```

Steps of merge sort

1. Split ✓
2. Merge Sort left ✓
3. Merge Sort right ✓ — recursively ✓
4. Merge ✓



$$T(n) = 2T(n/2) + cn$$

$$O(n\log n)$$