Linked List

The MyList class is a linked list of Person objects (String name, int age).

1. addLast(String xName, int xAge) – check if xName has the first letter 'B' or xAge < 17 then do nothing, otherwise add new person to the end of the list.
2. Delete the first node having age = 20.
3. Display the first 5 person having age > 22
   (a, 20) (**b,24**), (**c,23**), (**d, 31**) (e,20) (**f, 24**) **(g, 24**) (h, 25)
4. Find the second max age. Display the first person having that age.
5. Sort the list descendingly by age.
6. Delete the last node having age = 20
7. Display the last 5 person having age > 22
8. Find the third max age.
9. add(String xName, int xAge, int index) – insert the new person at the given index (start from 0)
10. sort(int startIndex, int endIndex) – sort the linked list ascendingly by name from startIndex to endIndex

EduNext:

CQ1.

Display the first 5 person having age > 22

(a, 20) (**b,24**), (**c,23**), (**d, 31**) (e,20) (**f, 24**) **(g, 24**) (h, 25)

CQ2.

void sortByAge() – sort the persons by the descending order of the age, if same age, sort the persons in the alphabetical order of the name.

CQ3.

Display the last 5 person having age > 22

CQ4.

sort(int startIndex, int endIndex) – sort the linked list ascendingly by name from startIndex to endIndex

CQ5.

Delete the last node having age = 20

CQ6.

Delete the first node having age = 20

CQ1

What does this method do?

If you find any problem, how do you fix it?

```
void sortByAge() {
    if (isEmpty() || head == tail) {
        return;
    }
    Node prev = head;
    while (prev.next != null) {
        Node later = prev.next;
        while (later != null) {
            if (prev.info.age < later.info.age) {
                int tmp = prev.info.age;
                prev.info.age = later.info.age;
                later.info.age = tmp;
            } else if (prev.info.age == later.info.age) {
                double s = prev.info.name.compareTo(later.info.name);
                if (s > 0) {
                    String tmp = prev.info.name;
                    prev.info.name = later.info.name;
                    later.info.name = tmp;
                }
            }
            later = later.next;
        }
        prev = prev.next;
    }
}
```

CQ2.

```
Node get(int i) {
    Node p = head;
    int count = 0;
    while (p != null) {
        if (count == i) {
            return p;
        }
        count++;
        p = p.next;
    }
    return null;
}

void sortAscendingIndex(int startIndex, int endIndex) {
    Node pi, pj;
    for (int i = startIndex; i < endIndex; i++) {
        for (int j = i + 1; j <= endIndex; j++) {
            pi = get(i);
            pj = get(j);
            if (pi.info.name.compareTo(pj.info.name) > 0) {
                Person temp;
                temp = pi.info;
                pi.info = pj.info;
                pj.info = temp;
            }
        }
    }
    traverse();
}
```

CQ3.

What does the method do? Give another solution for it.

```
void displayLastQuantityPersonHaveAgeGreaterThanAge(int quantity, int age) {
    if (isEmpty()) {
        System.out.println("Your list does not contain any elements!");
        return;
    }
    if (quantity > count()) {
        System.out.println("Quantity must smaller than the number of elements!");
        return;
    }
    int count = displayLastQuantityPersonHaveAgeGreaterThanAge(quantity, age, head);
    System.out.println();
    if(count < quantity) {
        System.out.println("Do not have " + (quantity-count) + " more person have age greater than " + age + ".");
    }
}
int displayLastQuantityPersonHaveAgeGreaterThanAge(int quantity, int age, Node p){
    if(p==null)
        return 0;
    int count = displayLastQuantityPersonHaveAgeGreaterThanAge(quantity, age, p.next);
    if(quantity == count)
        return count;
    if(p.info.age > age){
        System.out.print(p.info+" ");
        return count+1;
    }
    return count;
}
```

CQ4.

```
void displayFirstQuantityPersonHaveAgeGreaterThanAge(int quantity, int age) {
    if (isEmpty()) {
        System.out.println("Your list does not contain any elements!");
        return;
    }
    if (quantity > count()) {
        System.out.println("Quantity must smaller than the number of elements!");
        return;
    }
    Node p = head;
    while (p != null && quantity != 0) {
        if (p.info.age > age) {
            System.out.print(p.info + " ");
            quantity--;
        }
        p = p.next;
    }
    System.out.println();
    if (quantity > 0) {
        System.out.println("Do not have " + quantity + " more person have age greater than " + age + ".");
    }
}
```

CQ5.

```
void del() {
    if (isEmpty()) {
        return;

    }
    Node f = head;
    while (f != null && f.next != null) {
        if (f.next.info.age == 20) {
            break;
        }
        f = f.next;
    }
    if (f == null) {
        System.out.println("Doesn't exist !!!");
        return;
    }
    f.next = f.next.next;
    if (f.next == null) {
        tail = f;
    }
}
```