Northern Michigan University (Marquette Co, MI)
CS222-61-21W Computer Science I (Andrew A. Poe)
Quiz 7
Friday 19 March 2021 2:00 P.M. EDT
Time: 10 minutes

Given the following classes:

```
class LL {
 private:
  LLN *head;
};
```

```
class LLN {
 private:
  string data;
  LLN *next;
};
```

write the method

```
void LL::DeleteFirstProfessorEinstein ();
```

This method removes the node containing the string “PROFESSOR” from the linked list, provided that PROFESSOR is followed by EINSTEIN. However, EINSTEIN isn't removed; only PROFESSOR is. If there is more than one such node, this method removes the node closest to the head that satisfies the requirements.

For example, if the list were
PROFESSOR-->EINSTEIN-->IS-->SMARTER-->THAN-->PROFESSOR-->NEWTON
after running it would be
EINSTEIN-->IS-->SMARTER-->THAN-->PROFESSOR-->NEWTON

Do not use loops; use recursion only. You may write additional methods in LL and LLN if you wish. You may assume that standard constructors, destructors, accessors, and mutators have already been written. Make sure your code contains no memory leaks.

Northern Michigan University (Marquette Co, MI)
CS222-61-21W Computer Science I (Andrew A. Poe)
Quiz 7
Friday 19 March 2021 2:00 P.M. EDT
Time: 10 minutes

```
void LL::DeleteFirstProfessorEinstein () {

 if (!head || !head->getnext()) return;
 if (head->getdata()=="PROFESSOR" &&
     head->getnext()->getdata()=="EINSTEIN") {
  LLN *t = head;
  head = t->getnext();
  t->setnext(nullptr);
  delete t;
 } else
  head->DeleteFirstProfessorEinstein();
}

void LLN::DeleteFirstProfessorEinstein () {

 if (!next || !next->getnext()) return;
 if (next->getdata()=="PROFESSOR" &&
     next->getnext()->getdata()=="EINSTEIN") {
  LLN *t = next;
  next = t->getnext();
  t->setnext(nullptr);
  delete t;
 } else
  next->DeleteFirstProfessorEinstein();
}
```