# CMPSCI 187 Discussion #5: Hacking Linked Lists
# Individual Handout

David Mix Barrington
10 October 2012

Remember that every student gets a copy of this handout, but every **group** still hands in one response, on a separate answer sheet provided. **Do not hand in this paper.**

We've seen linked data structures several times now in lecture, some using the `LLNode<T>` generic class from DJW. Today we will play with a class that uses `LLNode` objects to describe a sled dog team. Here the class `SledDog` extends the class `Dog` from the exam, where a SledDog has an additional `String` field called `breed` and appropriate get and set methods.

Here is the code for `LLNode<T>` and some of the code for the classes `Dog`, `SledDog`, and `DogTeam`. These four classes in their complete form are in this directory:

`http://www.cs.umass.edu/~barring/cs187/disc`

```
public class LLNode<T> {
// taken directly from DJW on 8 Oct 2012

   private LLNode link;
   private T info;
   public LLNode (T info) {
      this.info = info;
      link = null;}
   // getters and setters

public class Dog {
   protected String name;
   protected int age;
   public Dog (String n, int a) { name = n; age = a;}
   // getters and setters

public class SledDog extends Dog {
   private String breed;
   public SledDog (String n, int a, String b) {
      super (n, a);
      breed = b;}
   public SledDog (String n, int a) {
      this (n, a, "Husky");}
   // getters and setters, toString
```

```
public class DogTeam {
   private LLNode<SledDog> lead;
   private int size;
   public DogTeam( ) { size = 0;  lead = null;}
   // getters and setters, isEmpty

   public void addToLead (SledDog newLead) {
      LLNode<SledDog> newNode = new LLNode<SledDog> (newLead);
      newNode.setLink (lead);
      lead = newNode; size++;}

   public String toString ( ) {
      LLNode cur = lead;  String out = "";
      while (cur != null) {
          out += (cur.getInfo( ) + "\n");
          cur = cur.getLink( );}
      return out;}

   public static void main (String [ ] args) {
      DogTeam dt = new DogTeam( );
      dt.loadExample( ); // method included in .java file, not written here
      System.out.println (dt);}
}
```

Your task is to write and test five methods for `DogTeam` as indicated on the response sheet:

**Question 1:** A method `public SledDog removeLead( )` that removes and returns the lead dog. If the team is empty, your method should return `null` and not throw an exception. Don't forget to update the size of the team.

**Question 2:** A method `public void switchLastTwo( )` that will reverse the order of the last two dogs in the team, if the team has at least two dogs. (If it has zero or one dog the method should do nothing.)

**Question 3:** A method `public void rotate( )` that moves the lead dog to the rear of the team, so that each other dog moves up one space. If the team is empty it should do nothing.

**Question 4:** A method `public int countHuskies( )` for the `DogTeam` class that returns the number of dogs in the team whose `breed` field is exactly `"Husky"`. Remember that the `.equals` method of the `String` class returns whether the parameter string has the same letters in the same order as the calling string.

**Question 5:** A method `public SledDog removeYoungest( )` that removes and returns the dog in the team with the smallest `age`, taking the first dog of that age if there are more than one. If the team is empty, return `null` and don't throw an exception. Don't forget to update the size of the team.