Name:		
(as it would appear on official course roster)		
Umail address:	**@umail.ucsb.edu**	**section 9am, 10am, 11am, 12pm**
Optional: name you wish to be called if different from name above.		
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone"		

1

h05

CS32 F19

h05: Recursive Algorithms Review

ready?	assigned	due	points
true	Tue 10/15 02:00PM	Tue 10/22 02:00PM	50

You may collaborate on this homework with AT MOST one person, an optional "homework buddy".

MAY ONLY BE TURNED IN IN THE LECTURE LISTED ABOVE AS THE DUE DATE,
OR IF APPLICABLE, SUBMITTED ON GRADESCOPE. There is NO MAKEUP for missed assignments;
in place of that, we drop the lowest scores (if you have zeros, those are the lowest scores.)

Reading: Recursive Algorithms Review, DS 9.1, PS 14.1 and 14.2

1. (10 pts) Fill in the information in the header. The following are required to get the 10 "participation" points.
    - Filling in your name and umail address.

   Also: For paper submission PLEASE submit on ONE SHEET OF PAPER, double-sided if at all possible. If you must submit on two printed sheets write name on BOTH sheets and no staples, paperclips, or folded corners.

2. List two important pieces of information stored in an **activation record** (as discussed on p. 442 in the DS textbook):
    a. (5 pts) 1.

    b. (5 pts) 2.

3. There are two important parts to every simple recursive function: the base case, and the recursive call that makes progress towards the base case. (There are other forms of recursion, such as "mutual recursion", where `foo()` calls `bar()` and `bar()` calls `foo()`, but let's set those aside for the moment, and deal only with simple recursive functions). Something that can go wrong with recursion when it is used incorrectly is a **stack overflow**. Explain two different ways that a recursive function could be writen incorrectly that could lead to stack overflow. Hint: one has something to do with the base case, and the other has to do with the recursive call.
    a. (5 pts) 1.

    b. (5 pts) 2.

4. Given a fairly common definition for a `struct Node` that can be used to make a singly linked list of int values:

```
struct Node {
  int data;
  Node *next;
}
```

2

h05

CS32 F19

   a. (10 pts) Write an **iterative** function `printList(Node* head)` that takes a pointer to the head Node of the singly linked list and prints each value of the linked list, one per line. Write the entire function (including the function signature), and be sure to write correct and compilable C++ code in your solution.

   b. (10 pts) Rewrite the `printList(Node* head)` function from part a. **recursively**.