Problems 1, 2 and 3 all refer to these pointer declarations:

```c
Node *listData, *ptr1, *ptr2;
```

And assume that later in the program, listData is set to the address of the 
first node of a list that holds these data in this order: 25,30,45,60,65,80,90.

1. (6 pts) Write C statements to set ptr1 and ptr2 as directed.
   a. Set ptr1 to the address of the node holding 45.
   b. Set ptr2 to the address of the node holding 80.     [Hint: use ptr1.]

2. (4 pts) Given what you did in problem 1, and what you've learned about 
   linked lists, show the values of each of the following expressions:
   a. ptr1->info ____________
   b. ptr1->next->info ____________
   c. ptr1->next->next->next->info == ptr2->info ____________      [1 or 0]
   d. ptr2->next->next ____________

3. Write C statements to perform each of the following tasks, using the 
   pointer variables listData, ptr1 and ptr2, as they are set in all parts above:
   a. (3 pts) Reset the info member of the node containing 30 to 50.
   b. (6 pts) Now the list isn't sorted anymore. Without using any constants, 
   swap the info now stored in the two nodes holding 50 and 45. You may declare 
   another variable - think about what type it should be.
4. (10 pts) Complete the function named lastInfo below. The function takes a Node * (a pointer to a struct node object), and it should return a copy of the info held in the last node of the list. For example, if p points to the first node of the list holding 7, 19, 1, 5, the function returns 5. You may assume the list contains at least one node.

/* return info contained in last node */
int lastInfo (Node *p) {
}

5. (6 pts) Complete the function named newNode below. The function takes an integer value, and it returns a pointer to a new Node object that is dynamically created using malloc (to allocate sizeof(Node) bytes). The info field of this new Node should contain a copy of value, and the next field should be NULL.

/* return dynamically allocated Node containing value and pointing to NULL */
Node *newNode(int value) {
}

6. (9 pts) Create a new list of Nodes containing these values in this order: 17, 45, 82. Point the variable named myList below at the first Node. Create new Nodes dynamically - by using the newNode function you wrote in part 5 above.

Node *myList = NULL;