The question is concerned with reorganizing a data structure so that nodes that are closely together (based on the logical structure of the data) are in addresses that are close together. Your answer should be designed to make it easy for someone to program the algorithms in *any* common algebraic (imperative) language (such as Algol, C, Pascal, and FORTRAN). Your comments should clearly explain the approach that the algorithms are taking. Ideally they should be a joy to read and no longer than necessary for a clear explanation. The algorithms should be no more complex than necessary.

Give algorithms for the three problems below. In all cases the input consists of a binary tree with the following fields: Data, Llink, and Rlink. For some of the problems you may find it useful to have a Mark field. When this is the case carefully describe the initial and final values of this field as well as its use. There is a pointer T that points to the initial node of the tree. All remaining nodes of the structure can be reached by following pointers.

It is desirable that the algorithms are efficient in their use of both space and time. Some additional assumptions appear to be necessary to achieve this. Examples of such assumptions are that all the data is in a single area logically equivalent to an array (which would not normally be the case in a C program) and the general calculations can be done with pointers (which cannot be done in standard Pascal). Please list any assumptions you make. Do not make more assumptions than necessary.

In all cases the goal of the program is to take the nodes that are initially in random order and rearrange them so that all nonempty Llinks point to the node in the next address. The Rlinks should point to nodes at some later address, and it is slightly better when that address is not too far away.

1. The program does not change the original structure and produces a copy in a new area of memory. (This is the trivial version of the problem.)
2. The program destroys the original version of the structure as it computes and puts the new version on top of the original version.
3. The program is like the previous one except that it also compresses the data by replacing the original Llink field with a one bit Ltag field.

When the question is graded, approximately half of the credit will be given for clear descriptions of the ideas that the programs are based on and the other hald will be given for the details of the algorithms.