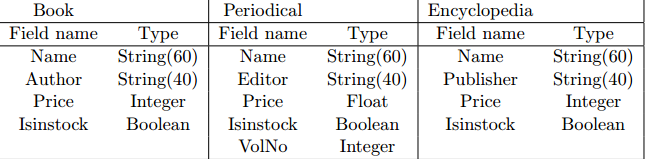
In a bookstore, there are three kinds of publications on sale: books, periodicals and encyclopedia. For each publication, a set of fields are maintained. The fields along with their types are given in the table below. For string types, the numbers in parenthesis show the maximum length.



There is a text file named ”inputbookdb.txt” storing all the publications. The file consists of a number of lines where each line is a publication, the format of which is dependent on the publication kind as specified next (note that the kind of the publication a line refers to is identified by looking at the first character in it).



Upon invocation, your program is expected to read the file ”inputbookdb.txt” and construct a binary search tree of heterogeneous publications dynamically in the heap. In the binary search tree, there is a node for each publication and name of the publication serves as the key. Your program should be interactive and should provide the user with the facility of traversing 1the list of publications and showing the details upon request. Also the program is expected to provide the user with adding new publications and deleting the existing ones. The content of the publication list should be able to stored in the file ”outputbookdb.txt” that has the same format as ”inputbookdb.txt”. The functionality is detailed next. • Display: Display the details (all fields) of the current node. At the beginning the root is the current node. • Go left: Advance the current node so that it points to the left child. If there is no left child, report this and do not update the current node. • Go right: Defined analogously to ”Go left” above. • Go up: Advance the current node so that it points to the parent. If there is no parent, report this and do not update the current node. • Deletion: Delete the current node, reorganize the tree, and reset the current node to the root. • Insertion: Add a new publication to the tree. You are free to insert it to anywhere in the tree provided that binary search tree property is maintained. After the insertion, reset the current node to the root. • Save: Save the content of the binary search tree into the file named ”outputbookdb.txt”. The entries of the file need to be ordered (increasing order) based on the publication name (the key). • Quitting: Quit the program. Some other key information about the assignment; • The publication names serve as keys. • Your binary search tree need not to be balanced. Hence you can use simple algorithms to maintain binary search tree property. • Use C style memory allocation/deallocation, i.e. malloc and free functions. Do not use C++ style memory allocation/deallocation, i.e. the functions new and delete. • You are expected to define and use enumeration, union and record types as appropriate. Do not use class constructs. • Implement the binary search tree yourself using pointers, i.e. do not use libraries. Also pay attention that implementations not using pointers will be awarded marginally. 2• For uniformity, you have to use gcc C compiler, which is freely available on most Linux installations. You may also use it through Cygwin (http://www.cygwin.com/) under Windows platform. Cygwin is also for free. • Develop your program as a console application, i.e. application with no graphical user interface. A sample ”inputbookdb.txt” file with four publications will look like

