# CS 218 DATA STRUCTURES

# ASSIGNMENT 2- Binary Search Trees

# Fall 2023

**DUE**: 27th October 2023

NOTE: Late submissions will not be accepted

**TO SUBMIT:** Documented and well-written structured code in C++ in the classroom. Undocumented code will be assigned a zero.

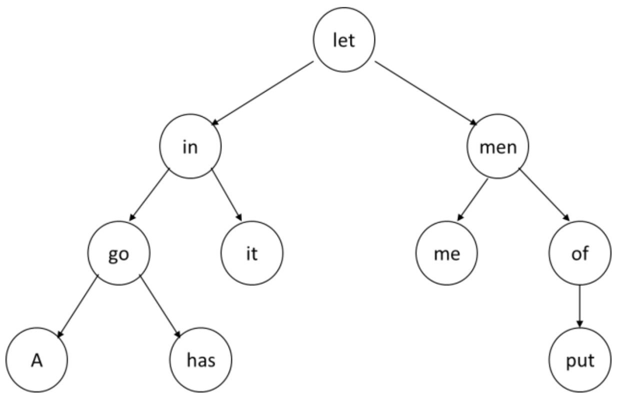
## Assignment Description

In this assignment, you will develop a cross-reference program that inputs a text file and constructs a specialized binary search tree with the words in the text file. The program records the details about the occurrence of the words in the text, such as chapter number, page number, and line number. These details should be stored on linked lists associated with the nodes of the tree.

Note that the text contains many commonly used English words such as **"a," "the," "is," "are,"** etc; such words are called Stop words. In CS, stop words are removed from the text before storing the details of the occurrence of each word.

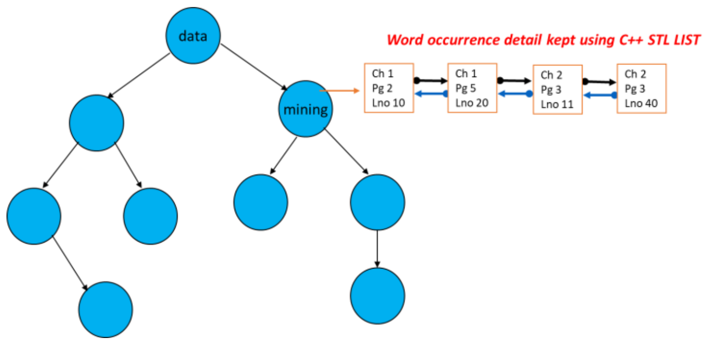
Your program should perform the following tasks.

1. **INPUT a list of stop words from the given file and store them in a binary search tree for efficient retrieval. Use the template-based BST you have developed in class (Lab) for stop words.**

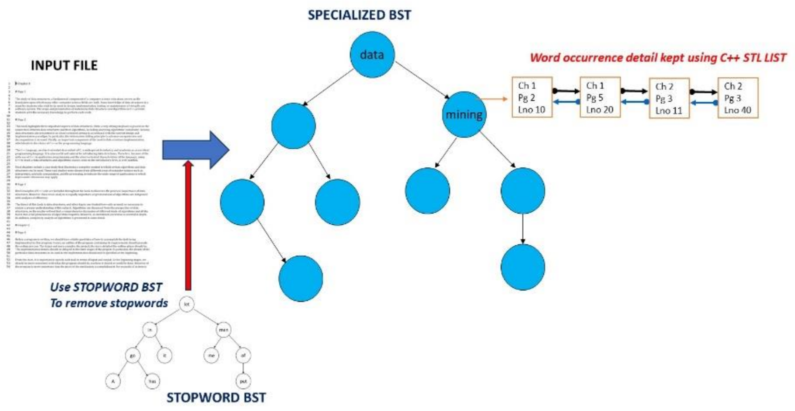


2. **CREATE a Class for specialized binary search tree (BST)** to store the details of the occurrences of words in the text. Each node of the specialized BST contains pointers to the left child and right child, a Key (a string to store words that occur in text), and a List to store the details of each occurrence of the word, such as chapter number, page number, and line number. For the LIST, you will use STL (standard template library LIST; ***each list element would be an object with a chapter, page, and line number).*** **FOR STL LINKED LIST** https://cplusplus.com/reference/list/list/

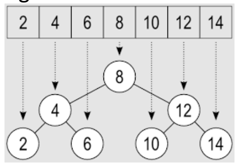
   *Note that your class for **specialized BST** should have a constructor, copy constructor, and destructor.*



3. **POPULATE specialized BST.** Input text from the given file (line by line), ignore extra spaces and punctuation, remove stopwords using stopword BST, and insert the remaining words along with the detail of their occurrence in the specialized BST tree as shown above.



4. **PRINT** in alphabetical order all words in the **specialized BST** along with the corresponding details of each occurrence, i.e., chapter no, page no, and line no.
5. **SEARCH,** get a word as input, search for it in the specialized BST, and print the details of all its occurrences.
6. **DELETE WORD:** Delete a given word from specialized BST along with all its occurrences**.**
7. **MERGE two specialized BSTs using the following idea.**
   a. Perform inorder traversal of the first **specialized BST** and store the values in a temp array in O(n1) time.
   b. Perform inorder traversal of the second **specialized BST** and store the values in another temp array. O(n2) time.
   c. The above arrays are sorted due to inorder traversal, merge them into one array in O(n1+n2) time.
   d. Construct a balanced **specialized BST** from the merged array in O(n1+n2) time using the following method:
      - Set the middle element of the merged array as the root of **specialized BST**.
      - Recursively do the same for the left half and right half.
         - Find the middle of the left half and make it the left child of the root created in step 1.
         - Find the middle of the right half and make it the right child of the root created in step 1.



8. **Find COMMON WORDS in two given Specialized BSTs in O(n1+n2) time. Use the idea similar to the one given above.**
   a. Perform inorder traversal of the first specialized BST and store the values in a temp array.
   b. Perform inorder traversal of the second specialized BST and store the values in another temp array.
   c. The above arrays are sorted, so find common words in O(n1+n2) time.