CS 2001 DATA STRUCTURES
ASSIGNMENT 3
SECTION C, D
Fall 2021

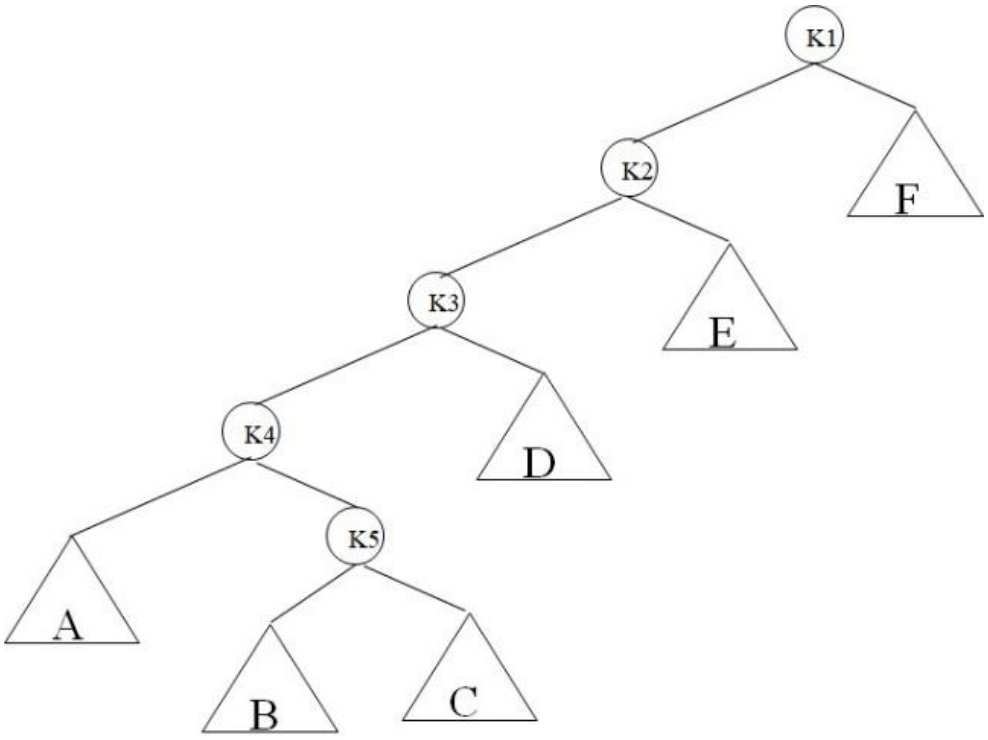
DUE: Nov 29, 2021

NOTE: Late submissions will not be accepted

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

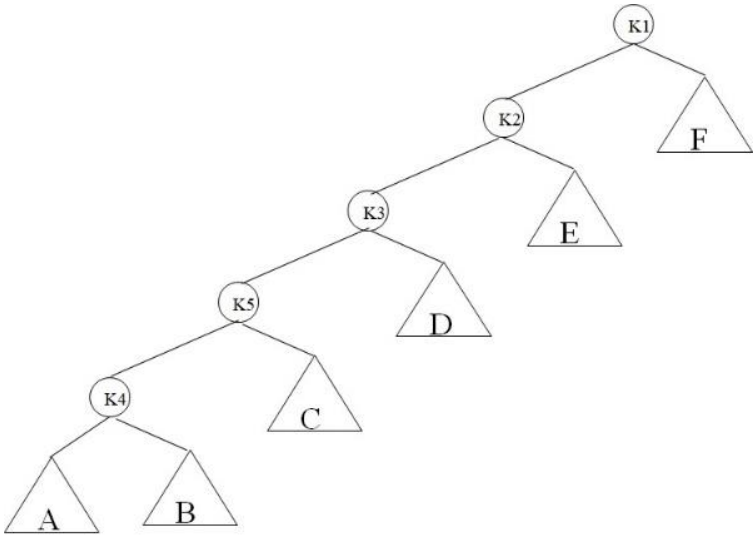
PROBLEM BACKGROUND

Many times, it is noticed that not all the data is frequently used in a particular time frame. For example, a hospital maintains the data of thousands of patients but only the records of those patients that are currently admitted in the hospital are accessed more frequently. One way to make our search operation efficient is to place the frequently accessed data item near root node. We want to modify the implementation of binary search trees such that recently accessed items are moved near to the root node (these items are highly likely to be accessed again). This way we can increase the efficiency of search operation. A simple way to move a particular data item to the root node is by rotating nodes on the search path with its parent node. Consider the following example where we want to move data item k5 near to the root node.

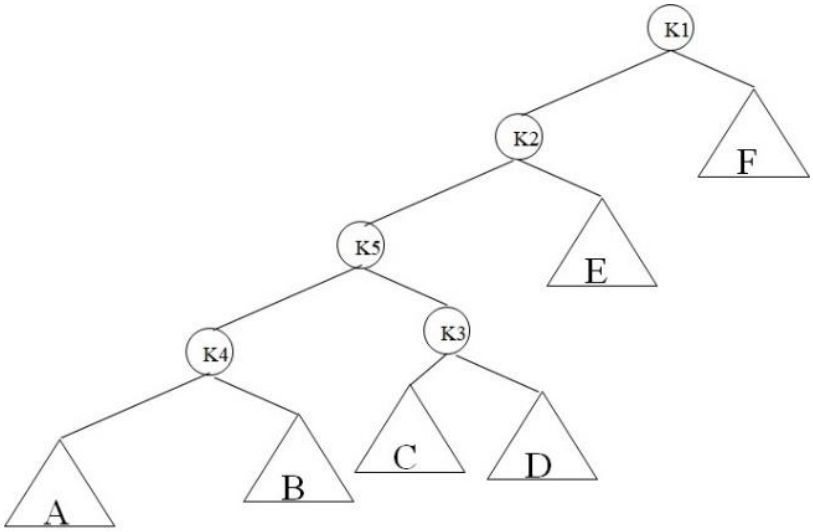


Here level/depth of k5 is 4 (root is at level 0) and we want to move k5 at depth 1. The following operations will be performed.

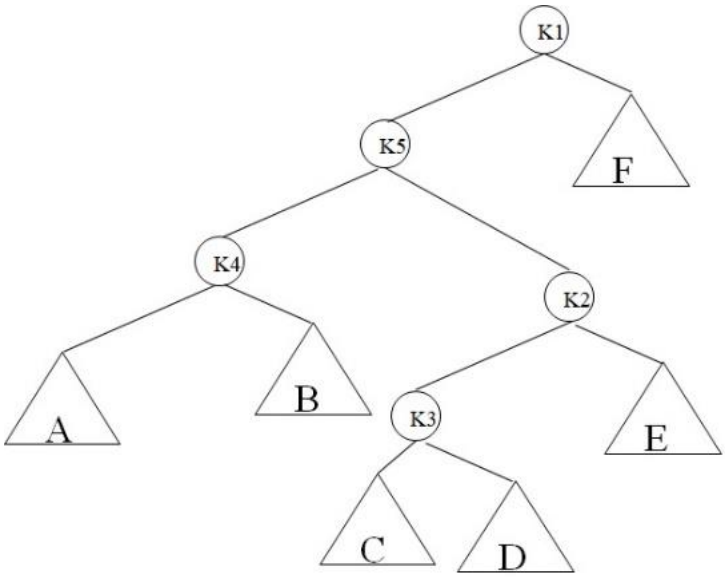
1



2



3



SYSTEM REQUIREMENTS

We want to design a software system for a hospital, where searching of its patients' data is the most frequent operation. We will store the records of patients in a binary search tree (BST) where search operation is modified according to the idea discussed above.

Each patient's record must include a patient id, patient name, admission date, disease diagnosis and patient status (admitted/discharged). The tree will be constructed on patient Ids. The hospital management want to perform following operations on patient records:

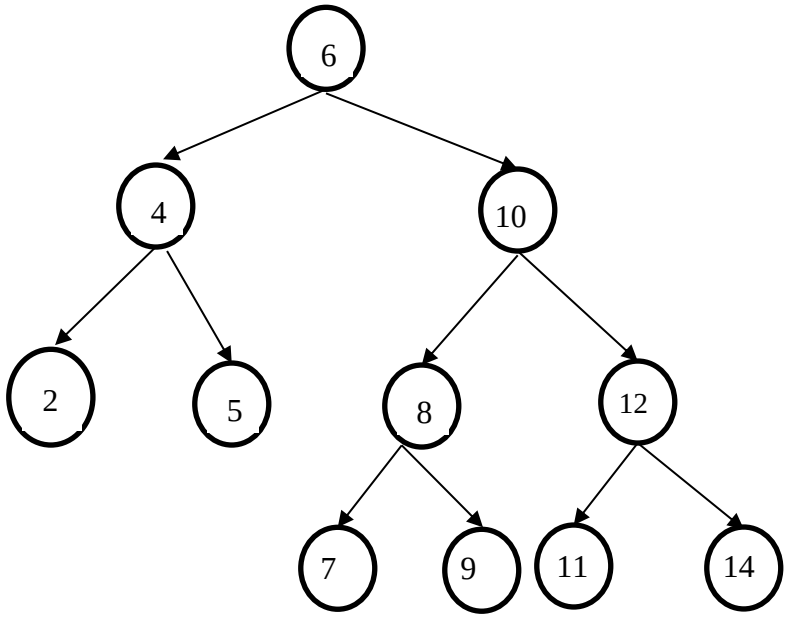
Insert a new patient record: this operation must insert the record of a newly admitted patient in the BST based on its patient id and move this node to the root node (it is most likely to be searched again)

Search a patient record and move its level up: this operation will search and returns the desired patients record given the patient id. Moreover, it will move the specified patient record (if found) up at level k. where k is given as parameter. The example given above demonstrates the result of a call where k5 is searched and is moved to level 1. If k is larger than the level/depth of the desired patient record then it will not be moved up.

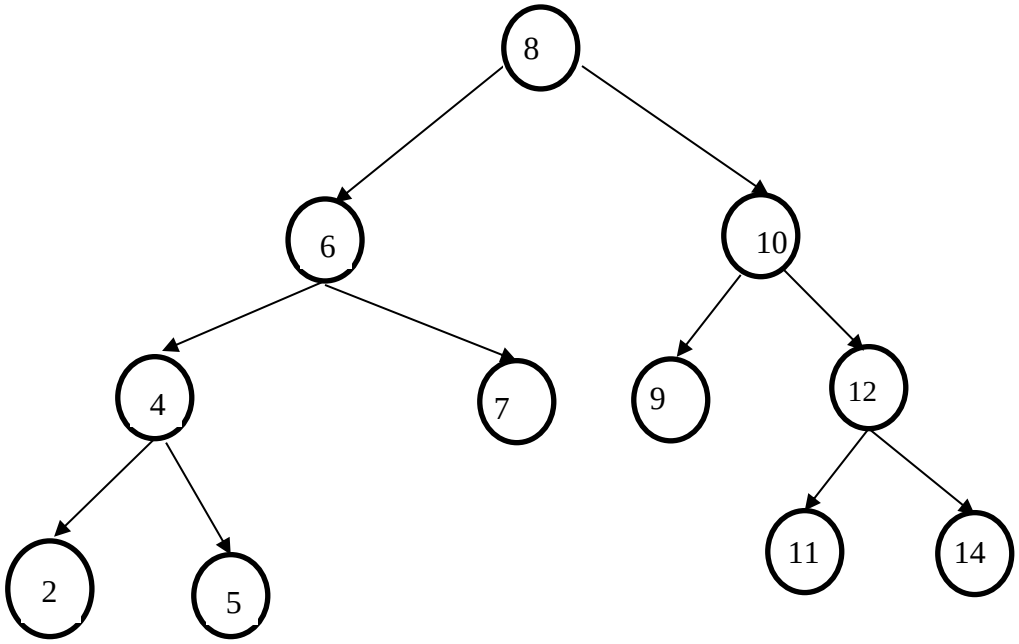
Remove an existing patient record: this must remove an existing patient record from the tree.

Edit a patient's record: This operation must edit the desired patient record to the new values given as parameter.

Split the patient record among 2 BSTs of same size: This operation must find the median of all the patient ids and construct two BSTs. One BST must contain records less than the median value and the other BST must include the records greater than the median. This operation must be performed in O(n) where n is the total number of patients. This can be done by moving median element to level 0 and its left and right subtrees will be the desired trees. Consider the following sample tree. Its median value is 8.



The resultant tree after moving the node containing 8 at level 0 will be



Print the records of all the admitted patients: This operation must print the records of all the patients that are currently admitted to the hospital.
Print the record of a given patient id: this operation must print the record of a patient whose patient id is given as parameter.

IMPLEMENTATION

Your task is to design a hospital management system that can fulfill the requirements specified in the above section.

IMPORTANT CLASSES

You have to implement the following classes

Class PatientRecord

This class must have following data members

- patient id
- patient name
- admission date
- disease diagnosis
- status (admitted/discharged)

Class TNode

This class must have following members:

- PatientRecord
- leftchild
- right child

Class HospitalData
This class must implement the following data members and member functions:
Data Members:

- Pointer to root of tree
- Size (number of records in the tree)

Member function:

Insert(): This function must take patientRecord as parameter and insert it in the tree according to the method described in the above section.

Search(): This function takes a patientid ***pid*** to be searched and level number ***k*** as parameters and returns the patientrecord if such patient exists. Otherwise it must give an error of not found. Moreover, if found this function will move the patient record with pid at level k.

Remove(): This function must take a patient id as parameter and removes the patient record from the tree.

Edit(): This function must take a patient id pid as parameter and edit its record.

Split(): This function must split the entire tree into two trees in linear time (O(n)) as described in the above section and returns the second tree.

Output(): This function takes patient id pid as parameter and output its record.

OutputAdmitted(): This function output all the records of patients whose status is admitted.