README - assignment 4
Soham Gaikwad , 2018cs10394

T.C denotes time complexity (in worst case generally)

1. Trie

Using node array of size 95 to store ASCII characters

a. boolean insert(String word, T value);
   Word is key.
   Insert node at position as in any trie using character of words
   T.C = O(length of word)
b. TrieNode<T> search(String word);
   @param word Search for this word, Case-Sensitive
   @return Returns the Trienode associated if the word is found else NULL
   Searches by matching indices with word characters
   T.C = O(length of word)
c. TrieNode<T> startsWith(String prefix); similar to search but differ as mentioned in interface
d. void printTrie(TrieNode trieNode);
   TC = O(const) as max iteration is const = 95
e. void printLevel(int level); // uses bfs using queues to print leve nodes // TC = O(log n)
f. void print(); // pritnLevel on all valid levels // TC = O(log n) , n is nodes of trie

REDBLACK
RedBlackNode
Many funtions are just setter and getter
isRed returns true if node is red
getSibling returns sibling of node TC=constant

RBTree
setLeftChild // links child to left of parent
TC = O(1)
setRightChild //similar

fixProblem(node) // resolves if node and parent have same key TC = O(logn) worst case when recur up to root is req.
insertIterative // inserts iteratively new node TC= O(logn) as for any bst
Insert // color new node red and resolve using fixProblem
Searcher // helper fn for search // search key in subtree of root TC =O(log n)
Search // uses searcher at root TC = o(log n)

ColorValue // enum for color of node

PriorityQueue

Student // same as description, compareTo compares student based on marks

MaxHeap
maxHeap // is arrayList of Node
HeapTime // incremented when inserted
Node // stores T element and its time of insertion
Compares on time of insertion if element compareto return 0
Left // return index of left child = 2 * curr + 1
Right // similar
Swap / / swaps heap node at given indices TC = O(1) or constant
moveUp //
Moves node at i s.t. Heap property is maintained
 TC = O(log n), n is no. of nodes in heap
Insert // add element to last of array heap and then call moveUp on new element to maintain heap property TC = TC of moveUp = O(logn)
extractMax // removes first in queue and replace last of queue with it. Then iterate downwards till heap property is maintained TC = O(log n)

Schedular part

In Job, project, user => setter and getter function
Job.completeTime is time when job is finished

SchedularDriver
All jobs are stored in RBtree as mentioned in assing
Finished jobs and unfinished jobs (called but budget less) are stored in Arraylist to maintain fifo order.

Methods work as described
Data Structures used are that which were suggested in assign