```
/**
 * This is the driver class for the project.
 * This program is able to read commands from an input file and prints out the results to a user
specified output file.
 * Commands are insert, deleted, contains, findMin, findMax, printTree, height and size.
 * Invalid commands that are read from the input file will raise an error and will be gracefully
handled.
 * Commands are case sensitive!
 */
public class Main

/**
 * This is where the program starts. Main method takes in 2 arguments from the command line, such
as "input.txt output.txt".
 * It reads the commands from the input file and passes them to processCommandsFile(). Commands are
case sensitive!
 *
 * @param args command-line arguments specifying input and output file names
 */
public static void main(String[] args)

/**
 * Read the command of the current line and execute its corresponding operation
 *
 * @param textLine a line from the input file representing a command
 * @param tree the LazyBinarySearchTree on which to perform operations
 * @param writer the PrintWriter for writing output
 */
private static void processCommandsFile(String textLine, LazyBinarySearchTree tree, PrintWriter
writer)

/**
 * Helper method that extracts an integer value from a command line string.
 *
 * @param line the command line containing a number
 * @return the extracted number
 * @throws IllegalArgumentException if the number format is invalid
 */
private static int extractNumber(String line)

/**
 * This class goes alongside Main. We receive commands from Main and process the logic here.
 * Keys must be in the range [1, 99]. Otherwise, IllegalArgumentException will get thrown.
 * Deleted nodes are marked as "deleted" but are not physically removed.
 */
public class LazyBinarySearchTree

/**
 * Inner class representing a node in the binary search tree.
 */
private class TreeNode
```

```
/**
 * Physically inserts a new element into the tree if it is not a duplicate.
 * It undeletes an existing logically deleted node.
 *
 * @param key the value to insert (must be in range 1–99)
 * @return true if a new node was physically inserted, false if a deleted node was undeleted or key
already existed
 * @throws IllegalArgumentException if key is outside valid range
 */
public boolean insert(int key)

/**
 * Searches for a node with the given key.
 *
 * @param key the key to search for
 * @return the TreeNode with the given key, or null if not found
 */
public TreeNode search(int key)

/**
 * Marks a node with the given key as logically deleted.
 *
 * @param key the value to delete (must be in range 1–99)
 * @return true if the node was successfully marked as deleted, false otherwise
 * @throws IllegalArgumentException if key is outside valid range
 */
public boolean delete(int key)

/**
 * Finds the minimum key that is not marked as deleted.
 *
 * @return the minimum undeleted key, or -1 if none exists
 */
public int findMin()

/**
 * Finds the maximum key that is not marked as deleted.
 *
 * @return the maximum undeleted key, or -1 if none exists
 */
public int findMax()

/**
 * Checks whether the tree contains the given key and it is not marked as deleted.
 *
 * @param key the key to search for (must be in range 1–99)
 * @return true if the key exists and is not deleted, false otherwise
 * @throws IllegalArgumentException if key is outside valid range
 */
public boolean contains(int key)

/**
 * Returns a pre-order traversal of the tree including deleted nodes (marked with *).
 *
 * @return a string representing the pre-order traversal of the tree
 */
@Override
public String toString()
```

```
/**
 * Returns the height of the tree (including deleted nodes).
 *
 * @return the height of the tree, or -1 if the tree is empty
 */
public int height()

/**
 * Returns the number of nodes in the tree (including deleted nodes).
 *
 * @return the size of the tree
 */
public int size()

/**
 * Validates that a key is within the allowed range [1, 99].
 *
 * @param key the key to validate
 * @throws IllegalArgumentException if key is outside the valid range
 */
private void validateKey(int key)
```