Word Binary Search Tree
Due Date: See WebCourses

In this assignment you will implement a binary search tree for storing words.

Your assignment should use this node structure.

```
typedef struct node_t{
   char *word;
   struct node_t *left, *right;
} node_t;
```

Your program will read in an input file called *words.txt*.
The format of input file is:

- Line 1 – An integer N, (0 <= N <=300), containing the number of words to be inserted in the tree at the beginning of processing
- Next *N* lines – Each line contains one word that should be inserted into the tree immediately
- Line *N*+2 – An integer *M*, (0 <= M <=30), describing the number of commands to be executed
- Next *M* lines – Each line contains an integer command code and an optional word. All lines will have command codes, but not all lines will contain a word.

Command Code	Command requires word?	Operation
1	Yes	Insert Word
2	Yes	Is word present in tree?
3	No	Print the height of tree
4	Yes	Find the word and print the height of the subtree rooted at that word
5	No	Print an in-order traversal of the tree

Requirements:
The words must be stored in dynamically allocated memory. You should allocate just enough memory to hold the word and the NULL character.

Notes:

- See the slides for a definition of height
- You must use a binary search tree.
- Use strcmp to compare words.
- For command 1, the system should print *Inserted <word>.* (See example below)
- For command 2, if the word is present in the tree, the system should print *<word> is present.* (See the example below)

- For commands 2 and 4, if the word is not present in the tree, the system should print, *<word> is not present*. (see the example below)
- For command 3 the system should print *The height is X.*
- For command 4, the the system should print *The height of the subtree at <word> is X.*
- For command 5, the system should print an in-order traversal of the tree. All words will be printed on one line, separated by a space.
- Your system should match the output examples below ***exactly***
- The input file will be named *words.txt*
- The file you turn in should be named *wordbst.c*
- **Don't forget the periods in the sentences!**
- **Follow the spec exactly!**

For this input file:

```
4
bravo
alpha
gamma
delta
12
5
2 chi
2 alpha
2 gamma
3
4 chi
4 alpha
4 bravo
4 gamma
4 delta
1 epsilon
3
```

The output should be

```
alpha bravo delta gamma
chi is not present.
alpha is present.
gamma is present.
The height is 2.
chi is not present.
The height of the subtree at alpha is 0.
The height of the subtree at bravo is 2.
The height of the subtree at gamma is 1.
The height of the subtree at delta is 0.
Inserted epsilon.
The height is 3.
```