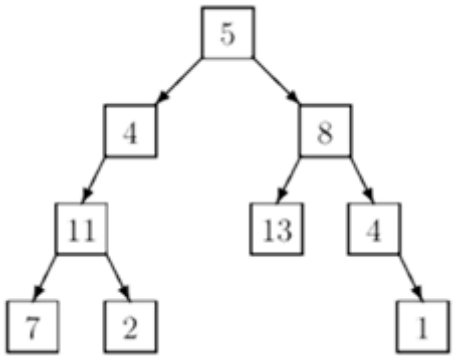
Assignment 4 - Trees
TA: Cody, email: codyhsu3196@gmail.com
Deadline: Nov. 26, 11:59 p.m.

In this homework, you will be asked to implement a binary tree using linked lists. Please implement the operation of summing of nodes in a binary tree. Additionally, you will be asked to implement a binary search tree using linked lists for storing words and the corresponding operations.

1. Summing of nodes in a binary tree (40%)

LISP was one of the earliest high-level programming languages and, with FORTRAN, is one of the oldest languages currently being used. Lists, which are the fundamental data structures in LISP, can easily be adapted to represent other important data structures such as trees. This problem deals with determining whether binary trees represented as LISP S-expressions possess a certain property. Given a binary tree of integers, you must build a binary tree according to the input. You need to write a program that determines whether there exists a root-to-leaf path whose nodes sum to a specified integer. For example, in the tree shown as below, there are exactly four root-to-leaf paths. The sums of the paths are 27, 22, 26, and 18. Binary trees are represented in the input file as LISP S-expressions having the following form.



An example of a binary tree

empty tree ::= ()
tree ::= empty tree | (integer tree tree)

The tree diagrammed above is represented by the expression:

(5 (4 (11 (7 () ()) (2 () ())) ()) (8 (13 () ()) (4 () (1 () ()))))

Note that with this formulation all leaves of a tree are of the form:

(integer () ())

Since an empty tree has no root-to-leaf paths, any query as to whether a path exists whose sum is a specified integer in an empty tree must be answered negatively (i.e., “no”). Otherwise, if the path whose nodes sum to the specified integer, answer “yes”. The input consists of a specified integer followed by a space and then a binary tree of integers in the LISP S-expression.

Each test case consists of an integer followed by one or more spaces followed by a binary tree formatted as an S-expression as described above. All binary tree S-expressions will be valid, but expressions may be spread over several lines and may contain spaces. There will be one or more test cases in an input file, and input is terminated by end-of-file.

Note: You must use linked lists; otherwise, no points will be given.

Sample Input

```
22 (5(4(11(7()())(2()()))()) (8(13()())(4()(1()()))))
20 (5(4(11(7()())(2()()))()) (8(13()())(4()(1()()))))
10 (3(2(4()())(8()())) (1(6()())(4()())))
8 (3(2(4()())(8()())) (1(6()())(4()())))
```

Sample Output

```
yes
no
yes
yes
```

2. Storing strings in a binary search tree (50%)

You will implement a binary search tree for storing words. The input consists of a sequence of strings. You must build a binary search tree according to the input order. When you insert a node with the value of the input string to the binary search tree, you must use strcmp() to compare words. The maximum length of a string is 10.

Note: In the C Programming Language, the strcmp function returns a negative, zero, or positive integer depending on whether the object pointed to by s1 is less than, equal to, or greater than the object pointed to by s2.

Your program will read in an input.
The format of the input 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 <= N <= 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	Is the parameter required? (yes or no)	Description
1	Yes	Insert a word.
2	Yes	Is the word in the tree?
3	No	Print the height of the 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 binary search tree

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

Sample Input

```
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
```

Sample Output

```
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.
```

3. Submission

To submit your files electronically, login DomJudge website through the following url :

https://108-1-ds-judge.ate.cs.ccu.edu.tw/

Press the submit button and choose the homework questions you want to submit. After submitting your code, DomJudge will give you a result to tell you whether your code is correct or not. However, during the demo time, your code will be evaluated by different sets of test cases. Please make sure your code can work correctly based on the description above. Additionally, you must compress your code and the README file into a zip file and upload it to Ecourse2.

4. Grading policies

The TA(s) will mark and give points according to the following policies:

- (40%) Part 1 source code can be compiled without any errors and the results are correct.
- (50%) Part 2 source code can be compiled without any errors and the results are correct.
- (10%) Readme file, coding style, and comments in the source code.

The readme file should include your name, student id, class id, a brief description of the code, and other issues students think that will be helpful for the TAs to understand the homework.