CSI 402 – Spring 2012

Programming Assignment I

Administrative Information

- **Deadline:** 11 PM, Friday, Feb. 10, 2012.

  Cutoff: 11 PM, Sunday, Feb. 12, 2012.

- The program must have three or more C source files.
  (More information on this is provided later.)

- All the files (C source files, header files (if any) and the `makefile`) must be submitted together using the `turnin-csi402` command.

- README file

  `~csi402/public/prog1/prog1.README`

  will be available by 10 PM on Saturday, Feb. 4, 2012.

- The `README` file will contain information regarding `turnin-csi402` and additional specifications for the `makefile`.

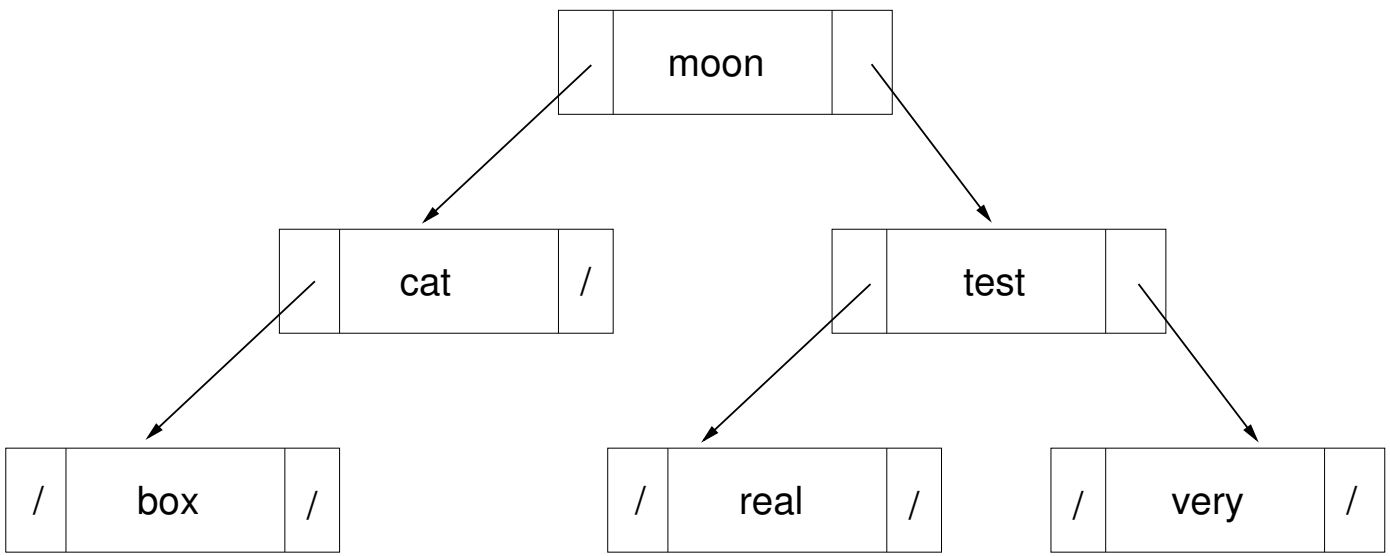
Binary Search Trees: A Brief Introduction

- Assemblers (and compilers) maintain a variety of (large) tables.
- Data structures for such tables must efficiently support insertion, deletion and search operations.
- Binary Search Trees (BSTs) represent one such data structure.
- For this program, each node of the tree has
  - A string (of length at most 15),
  - a pointer to its left child and
  - a pointer to its right child.

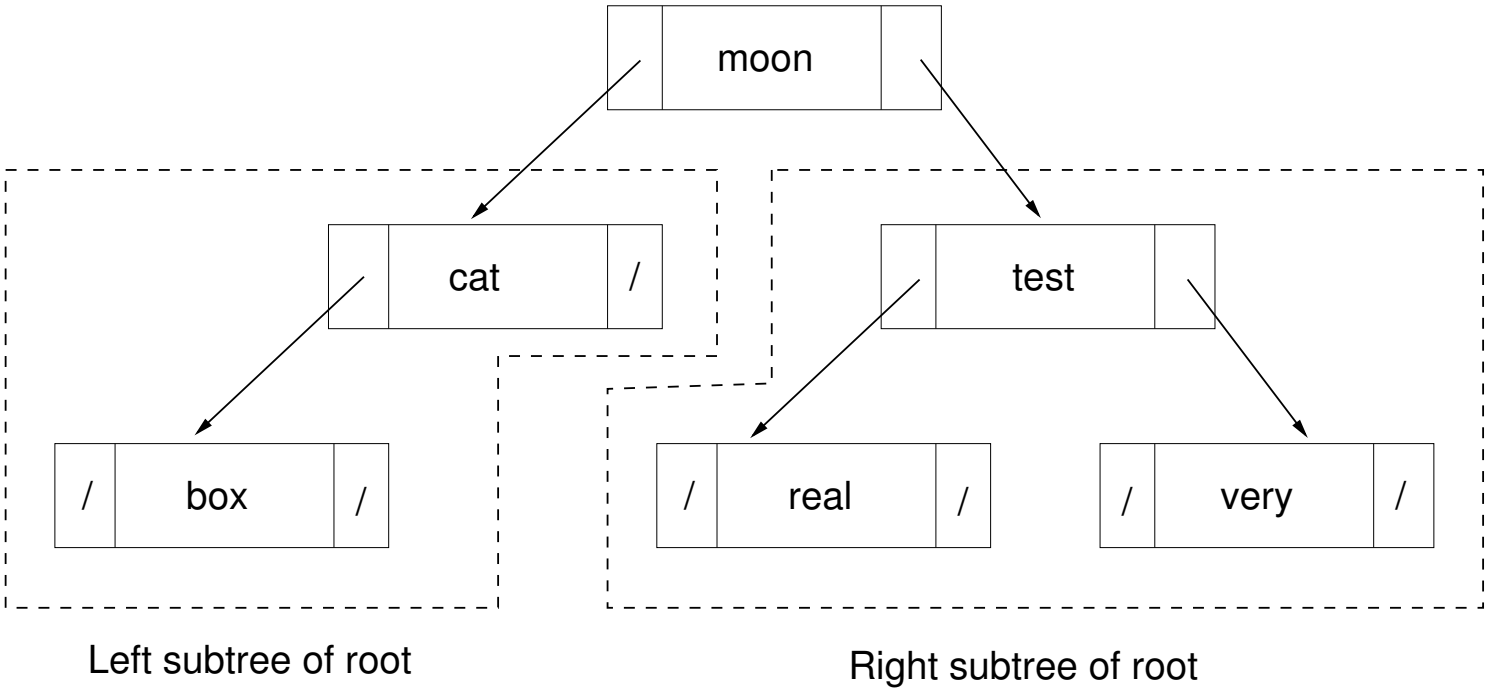
Binary Search Trees (continued)

- For each node containing string X:
  - Strings stored in all the nodes in the **left subtree** *precede* X in dictionary order.
  - Symbols of all the nodes in the **right subtree** *follow* X in dictionary order.

Example:



Binary Search Trees (continued)



- **Leaf:** A node which has no children. (In the above tree, nodes with the strings `box`, `real` and `very` are all leaves.)
- **Height of a tree:** The number of links in a *longest* path from the root to a leaf. (The height of the above tree is 2.)

Project Description

Goal: To construct a binary search tree from a collection of input strings and measure various parameters of the resulting tree.

Weightage: 5%

Total Points: 100 (Correctness: 85, Str. & doc: 15).

Unix Command Line:

% `p1` *infile* *outfile*

- `p1`: Executable version of your program.
- The input file (text file) contains the strings from which a binary search tree (BST) must be constructed.
- Parameters of the resulting tree must be written to the output file.
- **Errors to be detected:** Usual command line errors (see handout).

Project Description (continued)

Rules for Tree Construction:

- The tree is initially empty.
- Each line of the input file contains one string (of length at most 15).
- Strings must be inserted into the BST in the order specified in the file.
- The first string in the input file is stored at the root of the resulting tree.
- No rebalancing operations are done.

Project Description (continued)

List of Parameters to be Written to the Output File:

1. The total number of strings in the input file.
2. The height of the binary search tree.
3. The number of leaves in the binary search tree.
4. The height of the left subtree of the root.
5. The number of strings in the left subtree of the root.
6. The height of the right subtree of the root.
7. The number of strings in the right subtree of the root.

Project Description (continued)

Input file – Example 1:

```
moon
cat
box
test
very
real
```

Note: See the handout for assumptions regarding the input.

Project Description (continued)

Contents of Output File for Example 1:

```
Total number of strings in the input file =  6
Height of the binary search tree = 2
No. of leaves in the binary search tree = 3
Height of the left subtree of the root = 1
No. of strings in the left subtree of the root = 2
Height of the right subtree of the root = 1
No. of strings in the right subtree of the root = 3
```

Project Description (continued)

Input file – Example 2:

```
box
cat
moon
real
test
very
```

Project Description (continued)

Contents of Output File for Example 2:

```
Total number of strings in the input file =  6
Height of the binary search tree = 5
No. of leaves in the binary search tree = 1
Height of the left subtree of the root = 0
No. of strings in the left subtree of the root = 0
Height of the right subtree of the root = 4
No. of strings in the right subtree of the root = 5
```

Additional Notes

Structural Requirements:

- Your program must have **at least three C source files**.
  - One source file must contain just the `main` function.
  - A second source file must contain only the function that inserts a new node (containing a string from the input file) into the BST.
  - The functions that compute various parameters (e.g. height, number of leaves) of the BST must be in one (or more) source files.

Other Requirements:

- Your submission must contain all the C source files, header files (if any) and the `makefile`.
- You must use the `strcmp` function (from `<string.h>`) to correctly decide whether a string precedes or follows another in dictionary order.

Additional Notes (continued)

Suggestions:

- A suitable structure definition for each node of tree:

```
#define   MAXLEN   15

struct  tree_node {
   char  string[MAXLEN+1];
   struct tree_node  *left_child, *right_child;
};
```

- Use `fscanf` with format `"%s"` to read each string from the input file.
- You may use an iterative or recursive function for inserting a new node into the BST.

Additional Notes (continued)

Suggestions (continued):

- Using recursive functions will make it easier to compute parameters such as tree height and the number of leaves.

Example – Recursive Definition of Tree Height:

- The height of an empty tree or a tree consisting of just one node (leaf) is zero.

- The height of a tree T with two or more nodes is given by

$$\text{Height}(T) \;=\; 1 + \max\{\, \text{Height}(T_L),\;\; \text{Height}(T_R) \,\}$$

  where T_L and T_R denote the left and right subtrees of the root respectively.