

Letterkenny I.T., BSc Games Development, Year 3, Algorithms and Data Structures for Games  
Programming, Spring Term 2008. Lecturer: J.G. Campbell

**Assignment 2, Deadline for paper copy Wed. 5th March 2008, 10.30am.** Demonstrations must be completed by Wed. 5th March 10.30am. Worth 20% of module marks (one third of CA).

Late assignments without documented extenuating circumstances **will lose 10% of marks per day or part of day late. Those seven days late will receive zero marks.** If you have extenuating circumstances, you should contact me as soon as possible; in particular, be careful about wasting time on completing an assignment only to hand it in well after the deadline; whether you have valid reasons or not I cannot give any marks for work handed in after outline answers have been handed out or discussed in class.

**Computing Department Rules.** Assignments must typed and neatly presented and spell-checked. Sloppily presented work will be severely penalised or returned for correction and resubmission.

Poor layout of programs will be penalised.

**Expect to be questioned about the content of assignments.** I want assignments to be handed to me in the practical class of Wed. 13th Feb., so that I can ask questions.

Please use only one side of the page. There is no need for ornate front sheets or plastic binding; simply include the heading above (including my name), and your own name. A staple at the top left corner is the preferred binding – I can staple them if you wish. There is no need for plastic or other covering. Please use the Computing Department cover sheet and signed declaration.

**For numerical answers, where appropriate, show your working; a numerical answer written down with no explanation of working is likely to get zero marks.**

Software will be available in my public folder and on my website [www.jgcampbell.com/adsgp/](http://www.jgcampbell.com/adsgp/).

1. *100% of marks for demonstration.* Compile and execute BSTT1.cpp.

[10 marks]

2. *50% of marks for demonstration.*

- (a) Create a new program BSTT2.cpp that tests the use of our *binary search tree* for storing strings. Note: if you want to be able to display the tree *as a tree*, you will have to limit yourself to two or three character strings. Insert the initials of at least twelve people that you know; for ease of interpretation of the results, stick to all upper-case or all lower-case.

Hint:

```
BST<string> t;  
t.insert("rh");
```

Print the tree: (i) inorder, (ii) preorder, (iii) postorder, and (iv) as a tree.

[10 + 10 = 20 marks]

- (b) Now use `insert` to insert two more initials.

Print the tree: (i) inorder, (ii) preorder, (iii) postorder, and (iv) as a tree.

[5 + 5 = 10 marks]

- (c) Now use `erase` to erase two of the original initials.

Print the tree: (i) inorder, (ii) preorder, (iii) postorder, and (iv) as a tree.

[5 + 5 = 10 marks]

- (d) Now test `find` by calling it for an initials that is present, and for one that is not present.

Print the results.

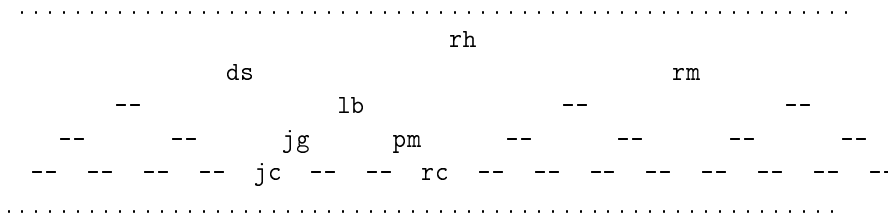
[5 + 5 = 10 marks]

3. 50% of marks for demonstration.

Add `size` and `empty` methods to `BST.h` and add code to `BSTT2.cpp` to test them. Make sure that `empty` is as efficient as possible and add comments to indicate how you decided on the implementation.

[20 + 20 = 40 marks]

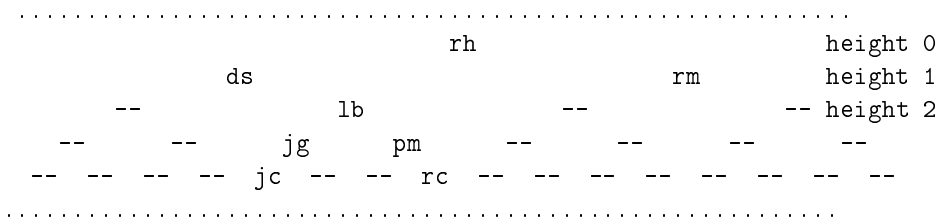
4. The *binary search tree* below contains strings of two characters.



- (a) What will the tree look like when we add (i) "xy", (ii) "ab", (ii) "rn", (ii) "rm"?
- (b) In the tree displayed above, give an explanation of the of the steps will be taken to find: (i) "jc", (ii) "zy" (not present), (iii) "jg".

[20 marks]

5. The *height* of a *binary tree* is defined as in the following diagram. A *complete* binary tree has nodes at each possible position, for example `xx` or `--` below.



- (a) How many nodes in a complete binary tree at height 0, 1, 2, 3, 4?
- (b) How many nodes in a complete binary tree of height 0, 1, 2, 3, 4?
- (c) How many nodes in a complete binary tree of height n? In addition to a formula, give a plausible explanation / proof of the answer.

[30 marks]

7. 50% of marks for demonstration. Implement and test (write a test code fragment) the following methods for BST (`BST.h`): (i) `size`, counts the number of nodes; (ii) `boolean empty`, returns `true` if the tree is empty, `false` otherwise. Make `empty` as efficient as possible.

[20 + 20 = 40 marks]

8. Give an outline, including diagrams as appropriate, of how to implement a general (n-ary) tree. Include, in outline (pseudocode), how you would implement methods:

- `size` (count the number of nodes);
- `empty` (boolean, is tree empty?);
- *pre-order* traversal;
- *in-order* traversal;
- *post-order* traversal.

[30 marks]

9. Based on the example in notes Chapter 9.3, apply the *minimax* algorithm to the following state of a noughts-and-crosses game. Show the full game tree containing all game states that follow that state.

```

      o| |o           turn x, max
    ---+---
      x|x|o
    ---+---
      | |x
  
```

[50 marks]

```

      o| |o           turn x, max
    ---+---
      x|x|o
    ---+---
      | |x
  
```

```

      | |           | |           | |           | |           | |           | |           | |
    ---+---  ---+---  ---+---  ---+---  ---+---  ---+---  ---+---
      | |           | |           | |           | |           | |           | |           | |
    ---+---  ---+---  ---+---  ---+---  ---+---  ---+---  ---+---
      | |           | |           | |           | |           | |           | |           | |
    ---+---  ---+---  ---+---  ---+---  ---+---  ---+---  ---+---
      | |           | |           | |           | |           | |           | |           | |
    ---+---  ---+---  ---+---  ---+---  ---+---  ---+---  ---+---
      | |           | |           | |           | |           | |           | |           | |
    ---+---  ---+---  ---+---  ---+---  ---+---  ---+---  ---+---
      | |           | |           | |           | |           | |           | |           | |
    ---+---  ---+---  ---+---  ---+---  ---+---  ---+---  ---+---
  
```