Computer Science

# COMPSCI 105 S2 C - Assignment 2

Due Date: Friday 27th October 2017 at 6:00pm

*100 marks in total = 7.5% of the final grade*

## *Assessment*

- Due: 27th October 2017 (6:00 pm)
- Worth: 7.5% of your final mark

## *Resources and Submission*

- Q4: Resources – `table1.txt`
- Q6: Resources - `ListBinaryTree.py`
- Please note:
  - All submitted files must contain your name and UPI
  - All programs you submit must display your UPI in the first line of the program output
  - All your files should be able to be compiled without requiring any editing.
  - All your files should include a good layout structure, meaningful variable names, and comments explaining the key ideas of your solution
  - All required resources are found on the assignment section in the course web page

## *Aims of Part II of Assignment*

- Understanding and solving problems using sorting algorithms and trees

## *Q4. Ordered Structures (15 Marks)*

In this exercise we develop an algorithm to sort the result table of the Super 18 rugby competition.

Input is a file with comma separated values. Each line of the file contains the name of a team, the "conference" the team is assigned to, number of points of the team and the goals for and against the team. The goal difference is computed as goals for a team minus goals against the team.

Please write a program `ResultTable.py,` which reads such a file and (a) outputs all teams in sorted order and (b) outputs the teams for each conference in sorted order.

The sorting order is defined as follows: higher points means a higher rank, if the points are equal then a higher goal difference corresponds to a higher rank. If the points and goal difference are equal then a higher number of goals for the team corresponds to a higher rank. If two teams have exactly the same number of points, goal difference, and goals for the team, then the order of these two teams is undefined (i.e. either could be in front).

The images on the next page show an example of an input file (left) and the corresponding output (right) of your program.

**Example:** In the input file (left) you can see:

Highlanders, 4, 52, 430, 286 => 52 points, 430 goals for and 286 goals against => goal difference is 144

Hurricanes, 4, 52, 458, 315 => 52 points, 458 goals for and 314 goals against => goal difference is 144

In the output (right) the Hurricanes are ranked higher because both teams have the same number of points and goal difference, but the Hurricanes have scored more goals then the Highlanders.

table1.txt - Notepad

File Edit Format View Help

```
Blues, 4, 40, 374, 380
Brumbies, 3, 43, 425, 326
Bulls, 1, 47, 420, 260
Cheetahs, 1, 22, 372, 487
Chiefs, 4, 51, 491, 341
Crusaders, 4, 50, 487, 317
Force, 3, 13, 260, 441
Highlanders, 4, 52, 430, 286
Hurricanes, 4, 52, 458, 314
Jaguars, 2, 22, 418, 476
Lions, 2, 52, 535, 349
Rebels, 3, 31, 365, 486
Reds, 3, 17, 290, 458
Sharks, 2, 47, 418, 231
Southern Kings, 2, 9, 282, 684
Stormers, 1, 47, 425, 265
Sunwolves, 1, 9, 293, 627
Waratahs, 3, 40, 413, 317
```

```
Process started >>>
    Team              Conference Points  Diff    Goals
 1. Lions                  2       52   186    535:349
 2. Hurricanes             4       52   144    458:314
 3. Highlanders            4       52   144    430:286
 4. Chiefs                 4       51   150    491:341
 5. Crusaders              4       50   170    487:317
 6. Sharks                 2       47   187    418:231
 7. Stormers               1       47   160    425:265
 8. Bulls                  1       47   160    420:260
 9. Brumbies               3       43    99    425:326
10. Waratahs               3       40    96    413:317
11. Blues                  4       40    -6    374:380
12. Rebels                 3       31  -121    365:486
13. Jaguars                2       22   -58    418:476
14. Cheetahs               1       22  -115    372:487
15. Reds                   3       17  -168    290:458
16. Force                  3       13  -181    260:441
17. Sunwolves              1        9  -334    293:627
18. Southern Kings         2        9  -402    282:684

 Conference  1
    Team              Conference Points  Diff    Goals
 1. Stormers               1       47   160    425:265
 2. Bulls                  1       47   160    420:260
 3. Cheetahs               1       22  -115    372:487
 4. Sunwolves              1        9  -334    293:627

 Conference  2
    Team              Conference Points  Diff    Goals
 1. Lions                  2       52   186    535:349
 2. Sharks                 2       47   187    418:231
 3. Jaguars                2       22   -58    418:476
 4. Southern Kings         2        9  -402    282:684

 Conference  3
    Team              Conference Points  Diff    Goals
 1. Brumbies               3       43    99    425:326
 2. Waratahs               3       40    96    413:317
 3. Rebels                 3       31  -121    365:486
 4. Reds                   3       17  -168    290:458
 5. Force                  3       13  -181    260:441

 Conference  4
    Team              Conference Points  Diff    Goals
 1. Hurricanes             4       52   144    458:314
 2. Highlanders            4       52   144    430:286
 3. Chiefs                 4       51   150    491:341
 4. Crusaders              4       50   170    487:317
 5. Blues                  4       40    -6    374:380
<<< Process finished. (Exit code 0)
```

**Please note:**

- Your program should display the rank, team name, conference, points, goal difference, and for and against goals separated by a ":". The output should be neatly formatted as shown above. For formatting I use the string format function (Python tutorial section 7.1 – See CS105 Resources page), but you can use any solution you like.
- You can use any sorting method you like. The simplest solution is probably to use one of the sorting methods from the lecture and modify the comparison operation appropriately.

However, it is also possible to use the in-build `sort()` method and provide a suitable key function, or to overload one of the comparison operators.

- In order to see whether your output is correctly formatted you need to use a typewrite font (i.e. where every character has the same width), e.g. Courier New. If you use the Wing IDE or the Windows console this should be automatically the case. If you use Notepad++ you can change the font using the menu item "Plugins->NppExec->Change Console Font".

## Marking Scheme for Question 4

| | |
|---|---|
| Reading input file and producing correctly formatted output | 3 marks |
| Teams correctly sorted by points | 3 marks |
| Teams correctly sorted by goal difference (if points equal) | 3 marks |
| Teams correctly sorted by goals scored (if points and goal difference equal) | 3 marks |
| Teams correctly sorted for each individual conference | 3 marks |

Put all the code that is needed to run the program in `ResultTable.py`

## *Q5. Binary Tree* *(10 Marks)*

A full node in a binary tree is a node with two children. Prove that the number of full nodes in a (non-empty) binary tree is one less than the number of leaves. Submit your proof as a pdf-file `Ass2Proof.pdf`.

**Please note:** You don't have to provide a formal mathematical proof. However, your proof should demonstrate understanding of the key ideas behind this proof, and an ability to explain technical issues using English language.

**Hint:** It is a good idea to use a process called "Structural induction"
Show that the claim is valid for a tree with one node (i.e. just the root)
Assume the claim is valid for a tree with n nodes.
Now show that the claim is valid for a tree with (n+1) nodes. In order to show this, consider any leaf node of the tree. Removing the leaf will give a tree with n nodes. With our assumption for that tree the number of full nodes is one less than the number of leaves. Now show that by adding the leaf back the number of full nodes is still one less than the number of leaves.

### Marking Scheme for Question 5

| | |
|---|---|
| Correct argument that claim is true for tree with one node (root) | 2 marks |
| Correct argument that claim is true for all trees | 5 marks |
| Acceptable use of English language and terminology | 3 marks |

Create a pdf-file `Ass2Proof.pdf` with your proof. If you write your proof by hand (e.g. in order to add illustrations) make sure that your handwriting in the resulting scanned document is readable. Markers are instructed not to give points for something they cannot read.

## *Q6. Binary Tree* *(25 Marks)*

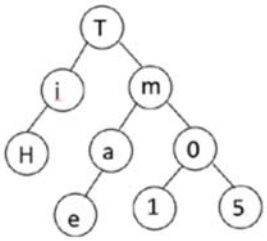
In this exercise we develop a method to reconstruct a binary tree from an inorder and postorder traversal sequence of the unknown tree.
Please write a program `ReconstructTree.py`, which lets the user input the inorder and postorder traversal sequences and from this reconstructs the corresponding binary tree and outputs it using the print() method.

**Please note:**

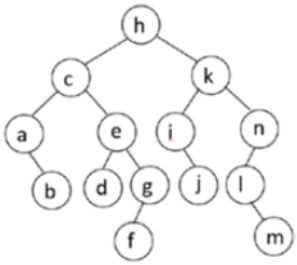
- All nodes of the tree contain a single character and the inorder and postorder traversal sequences are strings formed by these characters in the corresponding order.
- You can assume that all characters are different.
- Note that in our examples the reconstructed tree is a binary tree, but **not** a binary search tree
- Please construct the tree using a list-of-list representation and use the supplied file `ListBinaryTree.py`

**Example:** The screenshots below show on the left two examples of the program with input and corresponding output (left). The images on the right show graphical representations of the resulting trees (not required for the assignment).

```
Process started >>>
Binary Tree reconstructed by abcd001:
Please enter the inorder sequence: HiTeam105
Please enter the postorder sequence: Hiea150mT
[T, [i, [H, None, None], None], [m, [a, [e, None, None], None], [0, [1, None, None], [5, None, None]]]]
<<< Process finished. (Exit code 0)
================ READY ================
```



```
Process started >>>
Binary Tree reconstructed by abcd001:
Please enter the inorder sequence: abcdefghijklmn
Please enter the postorder sequence: badfgecjimlnkh
[h, [c, [a, None, [b, None, None]], [e, [d, None, None], [g, [f, None, None], None]]],
 [k, [i, None, [j, None, None]], [n, [l, None, [m, None, None]], None]]]
<<< Process finished. (Exit code 0)
================ READY ================
```




## Marking Scheme for Question 6

| | |
|---|---|
| Correct root of tree | 5 marks |
| Correct level 1 of tree | 5 marks |
| Correct level 2 of tree | 5 marks |
| All higher levels of tree are correct | 10 marks |

Put all the code that is needed to run the program in `ReconstructTree.py`