Instruction to candidates:

Your program code and output for each of Task 1 to N should be saved in a single `.ipynb` file using Jupyter Notebook. For example, your program code and output for Task 1 should be saved as:

```
Task1_<your name>_<centre number>_<index number>.ipynb
```

Make sure that each of your `.ipynb` files shows the required output in Jupyter Notebook.

1 Name your Jupyter Notebook as

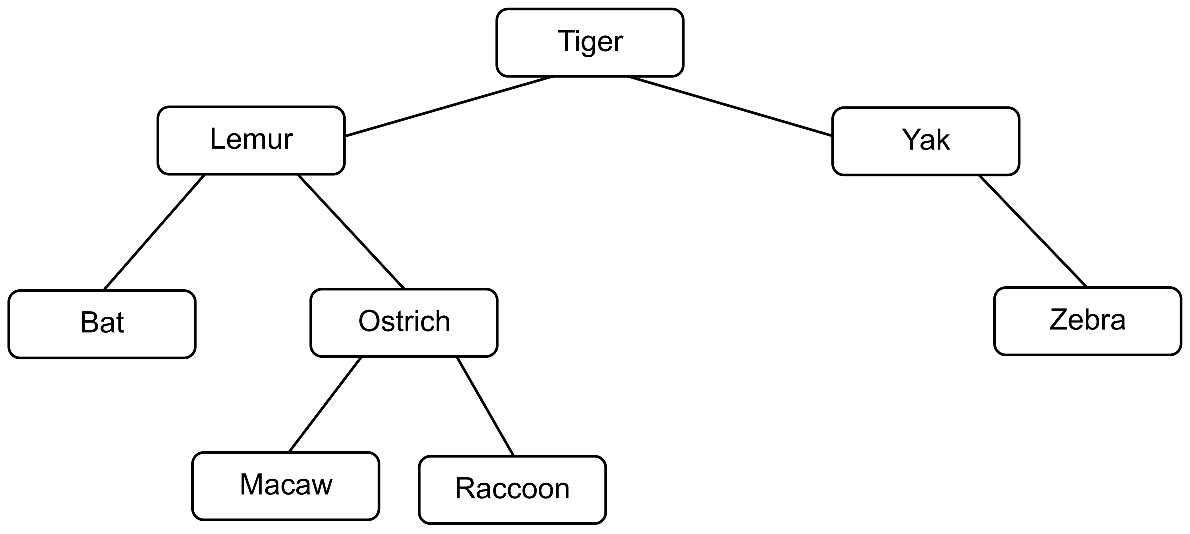
```
Task3_<your name>_<centre number>_<index number>.ipynb
```

A binary tree structure is used to store the names of animals in the Singapore Zoo. Each animal's name is unique. The binary tree abstract data type (ADT) has commands to create a new tree, add data items and print the tree.

The following sequence of commands:

```
CreateTree()
AddToTree('Tiger')
AddToTree('Lemur')
AddToTree('Bat')
AddToTree('Yak')
AddToTree('Ostrich')
AddToTree('Raccoon')
AddToTree('Macaw')
AddToTree('Zebra')
```

would create the following binary tree:



The program to implement the ADT will use the classes `Tree` and `Node` designed as follows:

Tree
- thisTree : ARRAY of Node - root : INTEGER

+ add(newItem)
+ print()

Node
- data : STRING - leftPtr : INTEGER - rightPtr : INTEGER

+ setData(s : STRING)
+ setLeftPtr(x : INTEGER)
+ setRightPtr(y : INTEGER)
+ getData() : STRING
+ getLeftPtr() : INTEGER
+ getRightPtr() : INTEGER

The program code will do the following:

- create a new tree, which has:
  - no nodes
  - the value of `root` set to `-1`
- use `root` as a pointer to the first node in the tree
- add a new node to the tree in the appropriate position, where `leftPtr` and `rightPtr` of this node should have the initial value of `-1`
- use the `print()` method to output, for each node, in array order:
  - `data`
  - `leftPtr`
  - `rightPtr`

For each of the sub-tasks, add a comment statement at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

```
In [1] :    # Task 3.1
            Program code

            Output:
```

Task 3.1

Write program code to define the classes `Tree` and `Node`.

[26]

Task 3.2

Test your implementation by writing a sequence of program statements to:

- create a new tree
- add the data items as shown in the sequence of commands on the previous page
- print the array contents

[3]

Task 3.3

A method `postOrderTraversal()` is to be added. This left-to-right post-order traversal outputs the data stored in the child nodes before outputting the data stored in the root node.
Write program code to:

- implement this method
- test the program code with the data stored in the binary tree created in **Task 3.2**.

[7]

Save your Jupyter Notebook for Task 3.