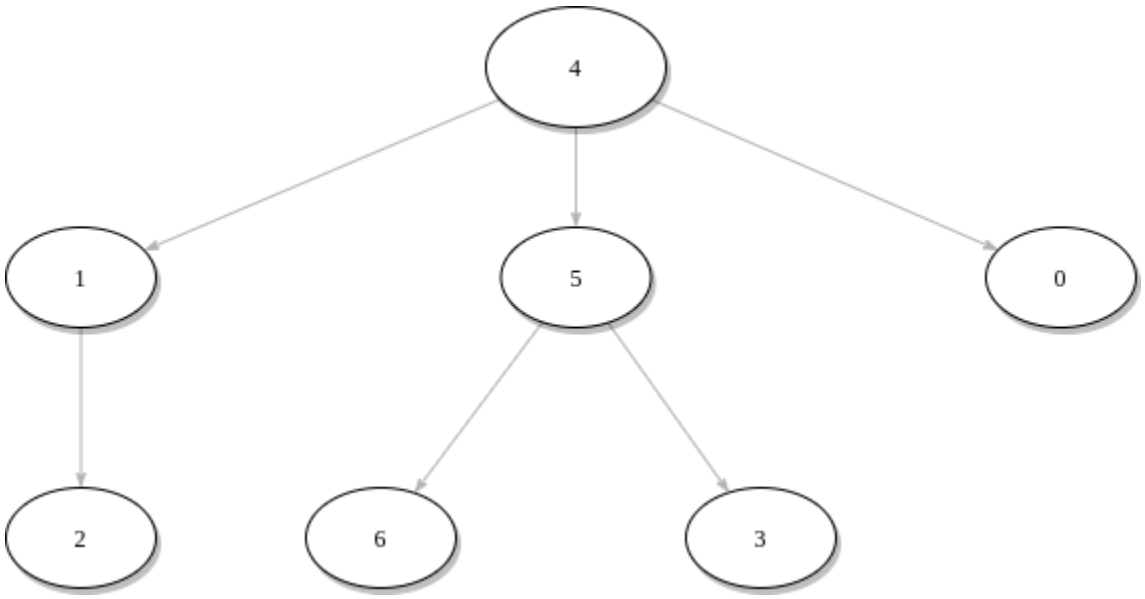
Senior Python Technical Test

1. Find internal nodes

Let's assume we have a generic tree composed of consecutive integers (so if there is a 6 all numbers starting from and including 0 up to it also need to exist on the tree), such as follows:



Then we define this tree with a list L: [4, 4, 1, 5, -1, 4, 5] such as L(i) identifies the parent of i (the root has no parent and is denoted with -1).

An internal node is any node (except the root) of a tree that has at least one child, so in this case the total number of internal nodes is **3**.

Implement the find_internal_nodes_num function (total number of internal nodes) as shown in the following snippet, knowing that:

- an internal node is any node of a tree that has at least one child;
- the solution should take into account complexity (time-wise) for very large trees;
- the solution should be implemented using Python.

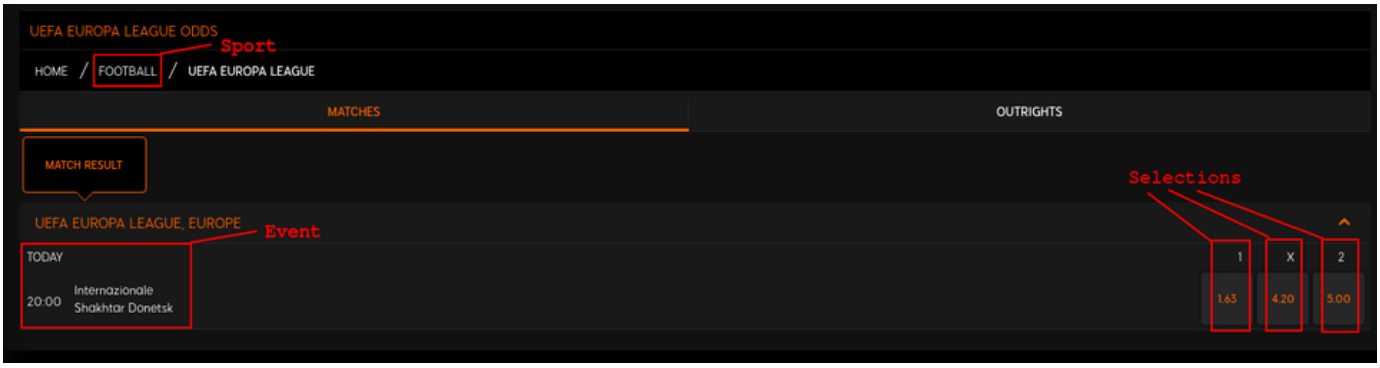
```
def find_internal_nodes_num(tree):
    pass

my_tree = [4, 4, 1, 5, -1, 4, 5]
print(find_internal_nodes_num(my_tree))
```

2. REST Application

888Spectate consists of a sportsbook product which is responsible for managing **sports**, **events** and **selections**.

The below screenshot illustrates how sports, events and selections are displayed (reference only, no need to implement a GUI):



Implement a simple REST API which manages multiple sports, events and selections, knowing that:

A **sport** contains the following elements:

- Name
- Slug (url friendly version of name)
- Active (either true or false)

An **event** contains the following elements:

- Name
- Slug (url friendly version of name)
- Active (Either true or false)
- Type (Either preplay or inplay)
- Sport
- Status (Pending, Started, Ended or Cancelled)
- Scheduled start (UTC datetime)
- Actual start (created at the time the event has the status changed to "Started")

A **selection** contains the following elements:

- Name
- Event
- Price (Decimal value, to 2 decimal places)
- Active (Either true or false)
- Outcome (Unsettled, Void, Lose or Win)

System Requirements

- REST API demonstrating the following functionalities:
  - Creating sports, events or selections
  - Searching for sports, events or selections
    - The system should be able to combine **N** filters with an **AND** expression
      - Filters may be more or less complex
  - Updating sports, events or selections
- A sport may have multiple events
- An event may have multiple selections
- When all the selections of a particular event are inactive, the event becomes inactive
- When all the events of a sport are inactive, the sport becomes inactive
- Sports, events and selections need to be persistently stored (SQLite is allowed)

Examples of Filters:

- All (sports/events/selections) with a name satisfying a particular regex
- All (sports/events) with a minimum number of active (events/selections) higher than a threshold
- Events scheduled to start in a specific timeframe for a specific timezone

NOTES:

- **Please provide the source code for the solution on GitHub or in a git archive.**
- **Mode of deployment is free (Make, containers, etc.).**
- **Candidate should submit solution using Python.**
- **Solution may be implemented using any framework but persistence level operations (i.e. MySQL, SQLite, PostgreSQL, etc.) MUST BE raw (i.e. models definition / creation is allowed, ORMs are not).**
- **Anything not specified is optional and up to the candidate (i.e. sports/events/selections removal functionality, caching, message formats, API endpoints, etc)**
- **Please add relevant unit tests.**