<u>Group 5 (RUMGuide): Homework #2</u>

<u>Exercise 1</u>: Refine Solutions

- Entities:
    - **Student:** This atomic entity contains the value of what task they are looking to accomplish. Such tasks consist of looking for an office in a building, calling a phone number for information or looking up information on how to complete a wanted task.
    - **Buildings:** this entity is composed of other entities such as different Floor and Office entities..
    - **Floors:** these composed entities contain another composed entity such as Offices.
    - **Offices:**a composed entity where Faculty members are found, which are an atomic entity.
    - **Faculty:** this atomic entity contains the value of information or documents to complete a task.
    - **White Pages:** This composed entity contains the different phone numbers that the university has for contact.
    - **University phone number:** this composed entity is the phone number to contact the university which contains extensions to the different offices, departments and buildings in the campus.
    - **Extensions:** This atomic entity has the value for each individual office in the university.
    - **FAQ:** This atomic entity holds the value for common questions that students make and a solution to them.
- Functions:
    - **Lookup building on map:** This function will receive as input a Student, determine where the location of his task within the campus is and return a Building entity according to what was determined.
    - **Lookup phone number: White Pages:** This function will receive as an input a Student, evaluate the Student's value (task) and return a phone number with the appropriate extension of the Building or Faculty that would best be suited to complete the task. This function can also receive no input and provide an organized list of all the phone numbers we have collected.
    - **Task Manager:** This function will receive as input a Student and determine their Task value, based on this it will redirect their value to one of these three inner functions within the Task Manager:
        - **Transcript Help:** The Transcript Help function is a derivative of the Task Manager function, this function returns detailed instructions on how the Student should go about collecting his transcripts.
        - **Summer Class Request Help:** this function is a derivative of the Task Manager function, this function returns a set of instructions for the

Student to follow to help the Student solicit summer classes in other universities within the UPR system.
- **Academic Minor Help:** this function is a derivative of the Task Manager function, this functions purpose is to return a step by step guide for a Student who is seeking to apply for an Academic Minor within the UPRM campus.

- Events:
    - **Located Office:** Event that occurs after looking up an office in the map and finding the exact place of the office a Student needs.
    - **Found number:** This event happens right after the student has searched for a phone number that they wish to call.
    - **Found Extension:** Similar to the previous event, this event marks the end of a search process the Student had to make to find the extension that corresponds to the office they wish to call.
    - **Call phone number:** Event that marks the end of a searching and dialing action and transitions to the action of gathering needed information.
    - **Receive instruction to complete task:** event where the Student collects every piece of information needed to complete a task after completing such actions as calling or going to offices and searching online.

- Behaviors
    - **Lookup on map:** Student opens the map feature. Following this, they look at the legend to find the abbreviation of the wanted building. Then, they proceed to find it on the map. When found, the Student closes the Map.

    - **White Pages:** Student opened *White Pages* feature, searched through catalogue of campus-related phone numbers and extensions. Afterwards, retrieved the specific phone number they were looking for, and proceeded to call the number.

    - **Task Manager:** Student opened the *Task Manager* feature and searched to find the task they wanted to complete. After this, they followed the instructions laid out in each specific part, to complete the errand.

Exercise 2: Algebra

A. Map:
    a. For the project, the main algebra to be used will be *Dictionary Algebra* for modelling the campus' map with the location of the student. This algebra uses functions such as:
        - create empty directory
        - insert entry in directory
        - directory look-up
        - edit directory entry

- remove directory entry as operations.

```
class
  type
   Student, Building, Map
   value
    lookup: Student→Building
    insert: Building→Map→Map
    edit: Building→Map→Building
    remove: Building→Map
Axiom: if any value, such as looking up, inserting, removing, or editing,  is called without a
parameter, then attempting to execute it is undefined.
    axiom
       lookup(null)≡chaos
       insert(null)≡chaos
       edit(null)≡chaos
       remove(null)≡chaos
    end
Axiom: if a Map is empty, attempting to remove, edit, or lookup, is undefined.
   axiom
       isEmpty(Map),
       lookup(Student,Map)≡chaos
       edit(Building,Map)≡chaos
       remove(Building,Map)≡chaos
    end
Axiom: if a parameter of a value is not in the Map, it is undefined.
   axiom
       ∀ Student':Map, Building':Map:
       lookup(Student,Map)≡chaos
       edit(Building,Map)≡chaos
       remove(Building,Map)≡chaos
    end
  end
```

B. White Pages:

   a. Directory Algebra (Sets of ArrayLists)

```
Class
    Type
    Building, Office, Phone Extensions
    Value
    Lookup: Building → Office → Phone Extensions
    Insert: Building → newValue
    Edit: Building → oldValue
    Remove: Building → currentValue
```

Axiom

lookup(**NULL**) ≡ **chaos**
insert(**NULL**) ≡ **chaos**
edit(lookup(**NULL**)) ≡ **chaos**
remove(lookup(**NULL**)) ≡ **chaos**
∀ e,s:S
lookup(e) ≡ s
insert(e) ≡ s
edit(e) ∧ lookup(e) ≡ s
remove(e) ∧ lookup(e) ≡ s

End

C. Task Manager:

a. Graph: For Task Manager, the main algebra to be used will be *Graph* for the options structure for the student. This algebra uses functions such as:

i. create empty graph
ii. insert node in graph
iii. insert edge in graph,
iv. Trace edges in graph from node to node
v. depth first search in graph and
vi. breadth first search in graph, as operations

Class

Type

Node <E> Task

Value

Task →

Axiom

lookup(NULL) ≡ chaos
insert(NULL) ≡ chaos
edit(lookup(NULL)) ≡ chaos
remove(lookup(NULL)) ≡ chaos
∀ e,s:S
lookup(e) ≡ s
insert(e) ≡ s
edit(e) ∧ lookup(e) ≡ s
remove(e) ∧ lookup(e) ≡ s

End

Exercise 3: Mathematical Logic

Using the SDL(Specification and Description Language)

- Assuming that every office belongs only to one building.

Here, the axiom can be read that for all pairs of buildings and office, for each office from all the offices of a building, each office belongs to only one building in other words only one building contains that office.

```
Scheme Buildings =
    Class
        Type Buildings, Offices
        Value
            getOffices: Building -> Office-set,
            containsOffice: Building x Office -> Bool
        Axiom
            ∀ building: Buildings, office:Offices ●
                Let offices = getOffices(building) in
                    office ∈ offices =>
                        Card  {b | b:Building ● containsOffice(b, office)} ≦ 1
                    end
                end
```

- Assuming that each phone number extension belongs only to one faculty.

```
Scheme Faculty =
    Class
        Type Faculty, Extension
        Value
            getExtensions: Faculty-> Faculty-set,
            containsExtension: Faculty x Extension-> Bool
        Axiom
            ∀ faculty: Faculty, ext:Extension ●
                Let extensions= getExtensions(faculty) in
                    ext∈ extensions=>1
                        Card  {f | f:Faculty ● containsExtension(f, ext)} ≦ 1
                    end
                end
```