Project Report

- *Jessica Tingley*

Introduction

I implemented a partially persistent binary search tree for efficiently storing data with previous tree versions accessible in functional Asteroid. The algorithm is given below:

- Insert(x): Inserts x into the most recent version of the binary search tree.
    - Recursively insert into the left or right subtree (depending on key and current root values), returning a new node with the updated left/right.
    - Adds new root node to maintained list of root nodes.
- Contains(x, t): Check if the binary search tree at time t contains x.
    - Recursively iterate through the left or right subtree (depending on key and current root values).
        - Compare key value and root value at each iteration.
- Delete(x): Delete x from the most recent version of the binary search tree.
    - Recursively delete from the left or right subtree (depending on key and current root values).
        - If the key to be deleted is found, and it has
            - No children: Return a new node with no connections to the deleted node.
            - No left child: Return a new node with a pointer to the old node's right child.
            - No right child: Return a new node with a pointer to the old node's left child.
            - Two children: Return a new node containing the value of the old node's inorder successor. This new node will point to the unchanged subtree of the old node and a new node containing the changed subtree of the old node.

I also provided a use case of the data structure through a terminal based interface. Users are able to input individual or lists of gym members, delete individual or lists of members, and search for members at a specified iteration of the member list.

Implementation

The project consists of four separate files, the first being the main.ast file which contains the user interface logic. The partially_persistent_bst.ast file contains the functional implementation of the

data structure. The unit_tests.ast file contains smoke tests for the insert and delete functionality of the partially persistent binary search tree under various structural scenarios.

The partially persistent binary search tree is implemented as a PartiallyPersistentSet structure. This structure contains the 'public' functions of the tree as well as a maintained list of roots. These public functions handle possible improper input, call the 'private' recursive functions, and update the list of roots with the new returned roots.

Examples

In the first example, the user enters three names and then deletes one name. The node deleted is a leaf node. It can be seen that the delete is successful as the name is contained in the second to last version and is not contained in the most recent version.

```
(base) jessicatingley@Jessicas-Air-6 Tingley_301Midterm % asteroid -F main.ast
Welcome to the Gym's Member Database.

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit.

Note that each time you enter or remove a member, a new iteration of the member list will be created. With the first insertion, the iteration will be one.
Keep this in mind when searching for members.

Also note that failing to follow specified input format can result in the system crashing.

Enter a command to begin: E
Enter member name or list of member names (comma separated): Alice, Bob, Charlie

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: D
Enter member name or list of member names to be deleted (comma separated): Charlie

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Charlie, 3
Charlie is in the database at version 3: true

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Charlie, 4
Charlie is in the database at version 4: false
```

The next example demonstrates successful deletion of a node with one child. It can be seen that the node containing Charlie exists at iteration four and not at iteration five, but the node containing Diana still exists at iteration five.

```
(base) jessicatingley@Jessicas-Air-6 Tingley_301Midterm % asteroid -F main.ast
Welcome to the Gym's Member Database.

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit.

Note that each time you enter or remove a member, a new iteration of the member list will be created. With the first insertion, the iteration will be one.
Keep this in mind when searching for members.

Also note that failing to follow specified input format can result in the system crashing.

Enter a command to begin: E
Enter member name or list of member names (comma separated): Alice, Bob, Charlie, Diana

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: D
Enter member name or list of member names to be deleted (comma separated): Charlie

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Charlie, 4
Charlie is in the database at version 4: true

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Charlie, 5
Charlie is in the database at version 5: false

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Diana, 5
Diana is in the database at version 5: true

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: Q
Exiting...
(base) jessicatingley@Jessicas-Air-6 Tingley_301Midterm %
```

The user again chooses to delete Charlie in the next example, though this time the node contains two children. By using the search functionality, it can be seen that Charlie exists at iteration five but not at iteration six. Nodes Eve and Diana are intact at iteration six.

```
(base) jessicatingley@Jessicas-Air-6 Tingley_301Midterm % asteroid -F main.ast
Welcome to the Gym's Member Database.

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit.

Note that each time you enter or remove a member, a new iteration of the member list will be created. With the first insertion, the iteration will be one.
Keep this in mind when searching for members.

Also note that failing to follow specified input format can result in the system crashing.

Enter a command to begin: E
Enter member name or list of member names (comma separated): Alice, Bob, Charlie, Diana, Eve

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: D
Enter member name or list of member names to be deleted (comma separated): Charlie

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Charlie, 5
Charlie is in the database at version 5: true

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Charlie, 6
Charlie is in the database at version 6: false

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Diana, 6
Diana is in the database at version 6: true

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Eve, 6
Eve is in the database at version 6: true

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: Q
Exiting...
(base) jessicatingley@Jessicas-Air-6 Tingley_301Midterm %
```

The final example demonstrates the deleting a list of names functionality. A list of five names is input and a list of three names is deleted. It can be seen that the names exist before deletion and no longer exist within the tree after deletion.

```
(base) jessicatingley@Jessicas-Air-6 Tingley_301Midterm % asteroid -F main.ast
Welcome to the Gym's Member Database.

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit.

Note that each time you enter or remove a member, a new iteration of the member list will be created. With the first insertion, the iteration will be one.
Keep this in mind when searching for members.

Also note that failing to follow specified input format can result in the system crashing.

Enter a command to begin: E
Enter member name or list of member names (comma separated): Alice, Bob, Charlie, Ben, Jane

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: D
Enter member name or list of member names to be deleted (comma separated): Bob, Ben, Jane

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Bob, 2
Bob is in the database at version 2: true

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Bob, 6
Bob is in the database at version 6: false

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Ben, 4
Ben is in the database at version 4: true

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Ben, 7
Ben is in the database at version 7: false

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Jane, 5
Jane is in the database at version 5: true

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: S
Enter name of member followed by the member list iteration which you would like to search (comma separated): Jane, 8
Jane is in the database at version 8: false

Enter 'E' to enter member names, 'D' to delete member names, 'S' to search for member names, or 'Q' to quit: Q
Exiting...
(base) jessicatingley@Jessicas-Air-6 Tingley_301Midterm %
```

Conclusion

The immutable nature of data in functional programs lends itself to natural implementation of persistent data structures. With the need to have access to each previous version of the structure, the desired solution involves creating new data rather than changing existing data. This project also demonstrated the benefits of multi-paradigm programming as well as pattern matching. All parts of this project, from unit test to user interfacing, were able to be cleanly implemented in a recursive manner.