Program Description / Purpose of Assignment

The purpose of this assignment was to learn about disjoint sets and to implement a disjoint set class in C++. A disjoint set class is useful when dealing with several nodes (each node starts out as a set of size 1). These sets can be combined into different sets where no two sets share the same node.

Data Structure Description

The disjoint set class was based around a vector of DListNode pointers. The DListNode class was taken from the supplemental file package. The DListNode class was already implemented, but needed additional work. I implemented the getRepresentative(), getTrailer(), setRepresentative(), setTrailer(), getListSize(), and setListSize() class methods.

When making a new disjoint set, the first step is to initialize the class with X number of nodes that the disjoint set should hold. The constructor then reserves X elements in a vector. When a new node is added to the disjoint set, a new DListNode pointer is created and its address is stored in the vector. Nodes are placed at a given index in the vector based on that node's key value. The destructor iterates through this vector and deletes all nodes.

The node pointers are stored in a vector to provide O(1) access to all elements in the disjoint set class. A function called FindSet() will find an node's representative when either a node or key is passed as a function argument. When a node argument is passed to the function the function will simply return the value of the node's representative pointer. When an integer value is passed to the function the function will access the nodeLocator vector with that integer to find the given node and return its representative pointer.

The Union function takes two DListNode arguments. The second node's set is appended to the first node's set. Then corresponding values for ListSize, Representative, and Trailer are updated for the newly formed set. The Representative to this new set is returned at the end of the function.

Runtime Analysis

The best and worst case analysis for the FindSet() function is O(1) since with the given input arguments, the function will either access data in that node (its Representative) or will know precisely what element to access in the vector of all nodes.

The Union function has a linear runtime O(n). When uniting two sets, the reassignment of pointers to create a new set is a constant runtime, however, the entire set must have certain values updated such as ListSize, and each node in the newly appended set has to change its Representative pointers. Traversing this new set and changing values gives this function a O(n) runtime in all scenarios.

Compile and Running Instructions

Extract tar file to desired directory. In the terminal navigate to the folder where source files are located and run the following command to compile:

```
g++ *.cpp
```

After compiling, you can execute the program by running the following command:

```
./a.out
```

Logical Exceptions

Some possible bugs in the program include:

- Union-ing two nodes that are part of the same set. When this occurs, the program will alert the user and return the value of the given set's representative.
- Running MakeSet() for a key value larger than the predefined DisjointSet class size. Doing this will result in no changes to the class and the user being notified of the error.
- When allocating space to the DisjointSet, if the size is negative the user will be notified of the error and no data will be created or changed.
- When FindSet() is called with an invalid key value, the user will be notified and a NULL pointer will be returned.

C++ Object Oriented / Generic Programming / C++ 11 Features

This program relies on templates classes to provide maximum compatibility with different datatypes.

Testing Results

The program functions as intended and passed all tests with correct results.