Project 1: Birthday Party 09.29.20

Andrew Shiraki

1. Overview

1.1. Program Description

BirthdayParty project contains the BirthdayPartyClass which allows the creation, manipulation, and querying of the party's guest list. The BirthdayParty class and its function definitions are contained in BirthdayParty.h. Implementation of class and class functions is contained in BirthdayParty.cpp.

1.2. Implementation

A BirthdayParty object contains a list array which stores guest information in a hash table of linked lists. Each element of the list array contains a struct called "Bucket". Each Bucket can hold one guest's information and a pointer to the next Bucket. The last Bucket in each link list points to nullptr.

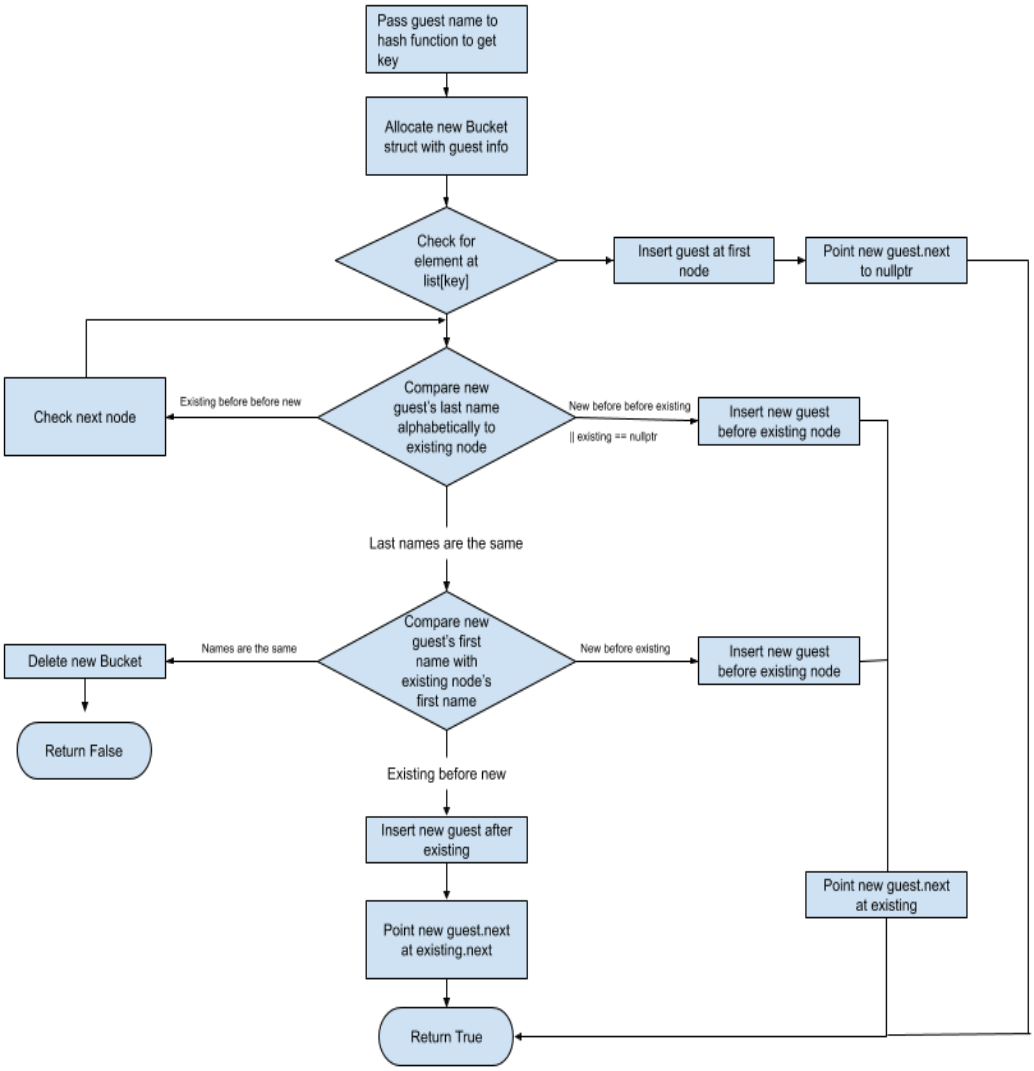
2. Algorithms

2.1. Hash Function

In order to maintain the hash table in a condition which allows alphabetical iteration through the guests, the hash function used returns the int value which corresponds with the letter number of the guest's last name. (e.g. `.hash("Anderson")` returns an int 0)

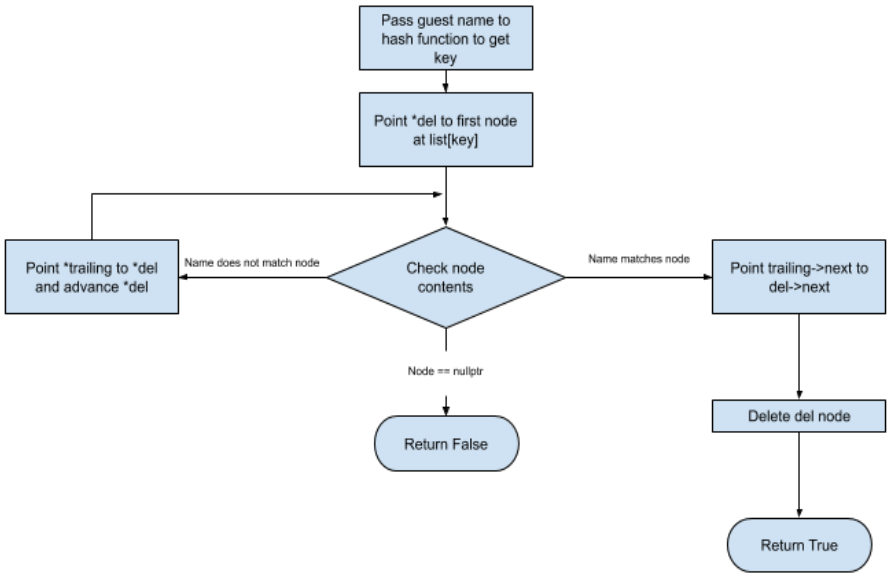
2.2. addInvitee

The addInvitee function accepts 3 arguments addInvitee(string firstName, string lastName, BirthdayType value) the function then hashes the name and attempts to insert it alphabetically into the linked list at the locations of the hash value



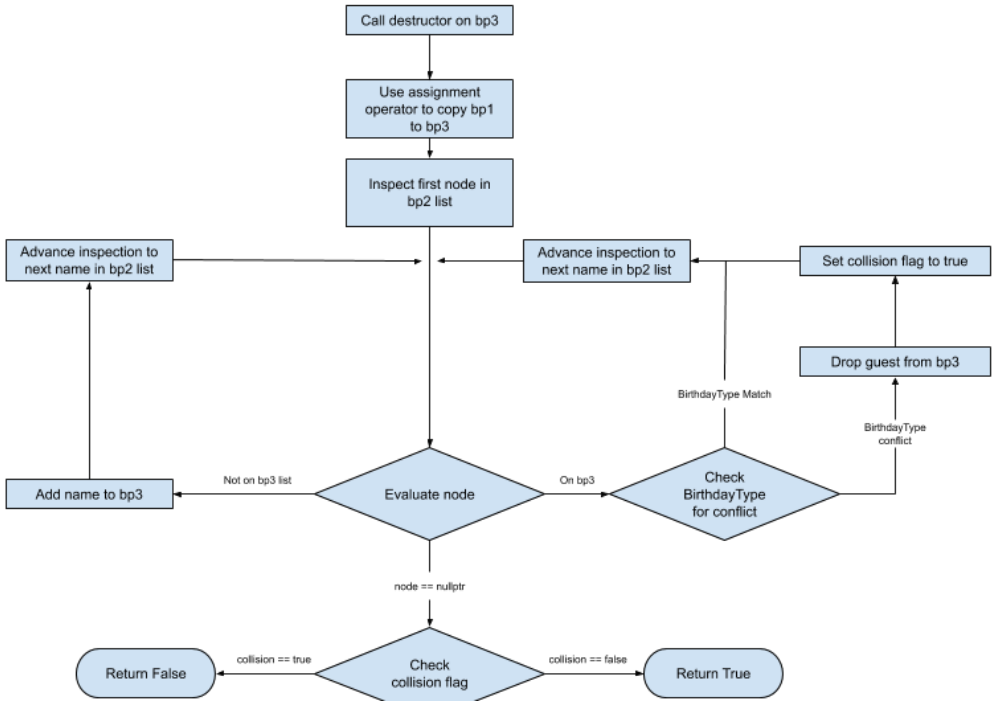
2.3. dropFromGuestList

The dropFromGuestList function accepts 2 args dropFromGuestList(string FirstName, string LastName), searches the guest list for an exact match, and removes that guest if found. dropFromGuest list returns true if the drop was successful and false if it was not



2.4. combineGuestLists

combineGuestLists is not a member of the BirthdayParty class. combineGuestList accepts 3 BirthdayParty objects as args combineGuestList(bp1, bp2, bp3) and combines bp1 and bp2 in bp3. A “collision” occurs when a guest is found on bp1 and bp2 with a contradicting BirthdayType value. combineGuestList returns true if there were no collisions and false if there were. The guest name which created the collision will not be included in bp3



3. Significant Obstacles

The obstacle which I found most challenging about this project was hashing guest’s names in a way which allows functions to iterate through the guest list alphabetically without knowing the contents of the hash table.

4. Test Cases:

4.1. spamList() function in main.cpp to iterate through guest list

```
void spamList(BirthdayParty bp)
{
	for (int n = 0; n < bp.whosOnTheGuestList(); n++)
	{
		std::string first;
		std::string last;
		BirthdayType val;
```

```
            bp.selectInvitee(n, first, last, val);
            std::cout << first << " " << last << " " << val <<
std::endl;
      }
}
```

4.2. theLastDance: this code block tests addInvitee function and alphabetical sorting

```
{
      BirthdayParty theLastDance;
      theLastDance.addInvitee("Michael", "Jordan", 23);
      theLastDance.addInvitee("Scottie", "Pippen", 33);
      theLastDance.addInvitee("Dennis", "Rodman", 91);
      theLastDance.addInvitee("Luc", "Longley", 13);
      theLastDance.addInvitee("Ron", "Harper", 9);
      spamList(theLastDance);
}
```

4.3. Dodgers: This block tests for proper handling of empty strings

```
      BirthdayParty dodgers;
      dodgers.addInvitee("Clayton", "Kershaw", 31);
      dodgers.addInvitee("Cody", "Bellinger", 11);
      assert(!dodgers.personOnGuestList("", ""));
      dodgers.addInvitee("Mookie", "Betts", 27);
      dodgers.addInvitee("", "", 57);
      dodgers.addInvitee("Justin", "Turner", 20);
      assert(dodgers.personOnGuestList("", ""));
      dodgers.dropFromGuestList("Mookie", "Betts");
      assert(dodgers.whosOnTheGuestList() == 4
            && dodgers.personOnGuestList("Clayton", "Kershaw")
            && dodgers.personOnGuestList("Cody", "Bellinger")
            && dodgers.personOnGuestList("Justin", "Turner")
            && dodgers.personOnGuestList("", ""));
```

4.4. Bps: this code block tests non member functions, combineGuestLists, and verifyGuestList. Additionally this block utilizes dropFromGuestList

```
      //test combineguest list
      BirthdayParty bp1, bp2, bp3;
      bp1 = BirthdayParty();
      bp2 = BirthdayParty();
      bp3 = BirthdayParty();
      bp1.addInvitee("Kobe", "Bryant", 8);
      bp1.addInvitee("Gianna", "Bryant", 19);
      bp1.addInvitee("Pau", "gasol", 24);
      bp1.addInvitee("Kobe", "bean", 12);
      //combineGuestLists(bp1, bp2, bp3);
      verifyGuestList("Kobe", "*", bp1, bp3);
```

```
      spamList(bp3);
      std::cout <<
"\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
      bp3.dropFromGuestList("Kobe", "Bryant");
      bp3.addInvitee("Andrew", "Shiraki", 1);
      bp3.addInvitee("Olive", "Shiraki", 5);
      bp3.addInvitee("", "Blanky", 5);
      bp3.addInvitee("blankerson", "", 23);

      spamList(bp3);
      std::cout <<
"\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
      spamList(bp1);
```