Data Structure Assignment 4 Document

```
    FlightHASHTABLE myFlightHASHTABLE(126683);
```

```
Capacity = 126683
```

Summation method:

Result:

```
    //hash function 1:
    //1k.file 1012 collisions
    //10k.file 102679 collisions
    //100k.file 10379641 collisions
```

Code:

```
    unsigned long h = 0;
    for (int i = 0; i < key.length(); i++) {
        h = long(h + key[i]) % capacity;
    }
    return long(h) ;
```

Polynomial method:

Result:

```
    //hash function 2:
    //1k.file 613 collisions
    //10k.file 55156 collisions
    //100k.file 5469478 collisions
```

Code:

```
    unsigned long h = 0;
    for (int i = 0; i < key.length(); i++) {
        h = long((h + pow(97,key.length()-i-1)*key[i])) %
capacity;
    }
    return long(h) ;
```

Cycle shift method:

Result:

```
    //hash function 3:
    //1k.file 0 collisions (7,57)
    //10k.file 105 collisions (7,57)
    //100k.file 8189 collisions (7,57)
```

Code:

```
    unsigned long h = 0;
    for (int i = 0; i < key.length(); i++) {
        char temp;
        if(isupper(int(key[i]))){
            temp = tolower(int(key[i]));
        }
        else{
```

```
            temp = key[i];
        }
        h = (h << 7) | (h >> 57); h = (h + long(temp)) % capacity;
    }
    return long(h);
```

Conclusion:

After implementing three different hash code functions, it is clearly shown that the cycle shift method is the best way to avoid key collisions. So, I set the cycle shift method as the default when running the code. Within the same dataset, the number of collisions has dramatically decreased from summation method, polynomial method, and to cycle shift method. The averge collisions for each item added from the file of 100k data drop from around 10, when implementing the summation method, to below 1, when implementing the cycle shift method.