Lab Exercise 11
Hashing

1 Introduction
This lab will demonstrate the effect of the hash function on the performance of a hash table. You will examine the effect of one simple hash function in detail, then try to devise your own.

Using Eclipse, import “Lab11” in the following directory:

/afs/cs.uwm.edu/users/classes/cs351/401/pantherid/git/lab11.git

2 Hashing
A hash table stores values in a table (array) and uses a hash function to generate integral indexes from keys. The key is passed through the hash function which returns an integer. This integer modulo the table size is used as index which determines where a specific value will be stored. Because there are frequently more possible keys than spaces in the array, it is possible for two keys to hash to the same array element. This is called a collision. There are many ways to deal with collisions; in this lab we will use linear probing, searching forward in the table until an open space is found.

3 A Simple Hash Function
We will be storing Person objects in our hash table. A Person object has a first and last name and implements the methods hashCode() and equals() that will be used to determine the hash table array index in which to store the Person.

The current implementation of hashCode() for Person simply returns the ASCII value of the first character of its lastName field, which is set as the last name of its associated Person. The hash table will generate the initial array index in which to store the Person using the absolute value of the result of hashCode() modulo the capacity of the hash table (which is initially 11). If the initial array index produced is already occupied, subsequent array locations will be examined until an open location is seen in which to store the object.

In the driver supplied, the following values are put into the hash table. Next to them, write the hash values for the current implementation. You may use the table on below the list of entries.

Casey McKeehe :
Corey Post :
Gregg Zaun :
Alcides Escobar :
Imad Zabalawi :

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>K - 75</td>
<td>L - 76</td>
<td>M - 77</td>
<td>N - 78</td>
<td>O - 79</td>
<td>P - 80</td>
<td>Q - 81</td>
<td>R - 82</td>
<td>S - 83</td>
<td>T - 84</td>
</tr>
<tr>
<td>U - 85</td>
<td>V - 86</td>
<td>W - 87</td>
<td>X - 88</td>
<td>Y - 89</td>
<td>Z - 90</td>
<td>a - 97</td>
<td>b - 98</td>
<td>c - 99</td>
<td>d - 100</td>
</tr>
<tr>
<td>e - 101</td>
<td>f - 102</td>
<td>g - 103</td>
<td>h - 104</td>
<td>i - 105</td>
<td>j - 106</td>
<td>k - 107</td>
<td>l - 108</td>
<td>m - 109</td>
<td>n - 110</td>
</tr>
<tr>
<td>o - 111</td>
<td>p - 112</td>
<td>q - 113</td>
<td>r - 114</td>
<td>s - 115</td>
<td>t - 116</td>
<td>u - 117</td>
<td>v - 118</td>
<td>w - 119</td>
<td>x - 120</td>
</tr>
</tbody>
</table>
Now draw the contents of the table as the Person objects are put in the table in the same order as above (see the put method in the Table class for further clarification).

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

What was the number of collisions? (If you had to probe repeatedly for one element, count that as multiple collisions)

Run the driver and verify that you had filled out the contents correctly.

4 A better hash function

The number of collisions is directly related to the hashCode implementation of the key. Exactly which hashCode implementation is best depends on both the distribution of the data and size of the hash table. Change the implementation of hashCode so that it produces half the number of collisions (or fewer) as the existing implementation.

Casey McKeehe: ________________________________
Corey Post: ________________________________
Gregg Zaun: ________________________________
Alcides Escobar: ________________________________
Imad Zabalawi: ________________________________

How many collisions does it generate? ________________________________

5 Eclipse hash function

Comment out the equals and hashCode methods from Person.java. Let Eclipse generate these methods by right clicking on the editor while Person.java is open and selecting Source > Generate hashCode() and equals()... How many collisions are generated when this implementation is used? Note that you probably will not want to compute the hash code by hand. You'll need to find out how many collisions happen during execution.

When you are happy with your hashCode implementation, see your TA for the lab checkoff.