2) (5 pts) ALG (Hash Tables)

Consider the following hash function, and then answer the questions that follow:

```
// This function assumes str is non-NULL and non-empty.
int hash(char *str) {

   int index = strlen(str) - 1;

   // Note: This converts letters on the range 'a' through 'z' or
   // 'A' through 'Z' to integers on the range 0 through 25.
   // For example: 'a' -> 0, 'b' -> 1, ..., 'z' -> 25.
   return tolower(str[index]) - 'a';
}
```

a) (2 pts) Give the hash code produced for each of the following strings:

hash("Not") = **19**

hash("Know") = **22**

hash("Bright") = **19**

hash("Moon") = **13**

hash("Now") = **22**

Points are based on number of correct answers:

0 or 1 correct answers -> 0 out of 2 points

2, 3, or 4 correct answers -> 1 out of 2 points

5 correct answers -> 2 out of 2 points

b) (3 pts) Using the hash values above, insert the strings (one by one, in the order given above) into the following hash table. Use **quadratic probing** to resolve any collisions. Note that there is a standard technique for dealing with hash values that exceed the length of a table (e.g., values that exceed 9 in the case of this particular table), and it's up to you to use that technique here.

Note: The length of the hash table is **10**.

0	1	2	3	4	5	6	7	8	9
Bright		**Know**	**Moon**			**Now**			**Not**

Grading: 1 pt for "Now" being in correct spot, 2 pts for **all** other strings being in correct spots.