1. [6] Starting with an empty hash table with a fixed size of 11, insert the following keys in order into three distinct hash tables (one for each collision mechanism): {12, 9, 1, 0, 42, 98, 70, 3}. You are only required to show the final result of each hash table. In the **very likely** event that a collision resolution mechanism is unable to successfully resolve, simply record the state of the last successful insert and note that collision resolution failed. For each hashtable type, compute the hash as follows:

hashkey(key) = (key * key + 3) % 11

12 = 4
9 = 7
1 = 4
0 = 3
42 = 7
98 = 4
70 = 8
3 = 1

Separate Chaining (buckets)

0	1	2	3	4	5	6	7	8	9	10
	3		0	12 → 1 → 98			9 → 42	70		

To probe on a collision, start at hashkey(key) and add the current probe(i') offset. If that bucket is full, increment i until you find an empty bucket.

Linear Probing: probe(i') = (i + 1) % TableSize

0	1	2	3	4	5	6	7	8	9	10
	3		0	12	1	98	9	42	70	

Quadratic Probing: probe(i') = (i * i + 5) % TableSize

0	1	2	3	4	5	6	7	8	9	10
	42		0	12	3		9	70	1	98

2. [3] For implementing a hash table. Which of these would probably be the best initial table size to pick?

Table Sizes:

NO 1 | 100 | 101 | NO 15 | 500

Why did you choose that one?

Id pick 500 because I don't know the size needed. The other options will have thier load factor reach .50 at minimum 5 times faster. Also with 500 Buckets we have less chance at collisions.