serial #	Test Case	visibility	difficulty	score	Comment
0	addressing.probe(1, 0)==30	Public	Easy	4	Testing default case of open addressing probe, where w = 10, seed for random number generator = 0: g(k, i) fails for k = 1, i = 0 Expected output: 30
1	addressing.probe(1, 1)==31	Public	Easy	4	Testing i of open addressing probe, where w = 10, seed for random number generator = 0: g(k, i) is called with k = 1, i = 1 Expected output: 31
2	addressing.probe(1, 3)==1	Public	Easy	4	Testing i modulo of open addressing probe, where w = 10, seed for random number generator = 0: g(k, i) is called with k = 1, i = 3 Expected output: 1
3	addressing.probe(2, 0)==28	Private	Easy	4	Testing a different key for open addressing probe, where w = 10, seed for random number generator = 0: g(k, i) is called with k = 2, i = 0 Expected output: 28
4	addressing.probe(4, 0)==25	Private	Easy	4	Testing a different key for open addressing probe, where w = 10, seed for random number generator = 0: g(k, i) is called with k = 4, i = 0 Expected output: 25
5	chaining.chain(1)==30	Public	Easy	4	Testing default case of chaining probe with w = 10, seed for random number generator = 0: h(k) is called with k = 1 Expected output: 30
6	chaining.chain(4)==25	Public	Easy	4	Testing a different key for chaining probe with w = 10, seed for random number generator = 0: h(k) is called with k = 4 Expected output: 25
7	chaining.chain(8)==19	Public	Easy	4	Testing a different key for chaining probe with w = 10, seed for random number generator = 0: h(k) is called with k = 8 Expected output: 19
8	chaining.chain(16)==6	Private	Easy	4	Testing a different key for chaining probe with w = 10, seed for random number generator = 0: h(k) is called with k = 16 Expected output: 6
9	chainInsert.insertKey(0)	Private	Easy	4	Testing insert key through chaining: inserted the key 0 to a hash table Expected Output: Size of the table should increase by 1 after the insertion

10	chainInsert.insertKey(32);
	chainInsert.insertKey(52);
	chainInsert.insertKey(72)
		Private	Easy	4	Testing collision after inserting 32, 52, 72 through chaining:
					Expected Output: Number of collisions should be equal to 2

11	openInsert.insertKey(0)	Private	Easy	4	Testing insertKey() function of open addressing: 0 is inserted into the hash table Expected Output : When probing the hash table with 0, probe() function should return 0

12	try {
	openInsertFull.insertKey(33);
	}
	catch(Exception e){
	return new GraderManager.Mark(0, 1);
	}
		Private	Easy	4	Testing insert key when the hash table is full:
					insert key 33 to an already full hash table thorugh open addressing.
					Expected output: should throw an error

13	openInsert.insertKey(32);
	openInsert.insertKey(52);
	openInsert.insertKey(72);
	int collisions = openInsert.insertKey(92)
		Private	Medium	6	Testing collision after inserting 32, 52, 72, 92 through insertKey() of open addressing:
					Expected output: Number of collisions should be equal to 3 after inserting 92

14	openInsert.removeKey(0)	Private	Easy	4	Testing remove key default case for open addressing: Removed key 0 from the hash table. Expected output: Key 0 should no longer be in the hash table.

15	openInsert.removeKey(52)
	return openInsert.Table[14]!=52
		Private	Medium	6	Testing remove key more than 1 collided case of open addressing:
					Removed key 52 from the hash table.
					Expected output: Table[14] should no longer hold 52

16	insert 32, 52, 72 remove 52 insert 92	Private	Medium	6	Testing remove key with more than 1 collided then adding a key back: insert 32, 52, insert 72, remove 52, insert 92 insert 32 --> Table[13] = 32 insert 52 --> Table[14] = 52 insert 72 --> Table[15] = 72 remove 52 --> Table[14] is empty insert 92 --> Table[14] = 92 Expected output: Table[13] = 32, Table[14] = 92, Table[15] = 72

17	openInsert.insertKey(69);
	openInsert.insertKey(89);
	openInsert.insertKey(109);
	openInsert.insertKey(129);
	int collisions = openInsert.removeKey(109);
		Private	Medium	6	Testing collisions in open addressing:
					insert 69, 89, 109, 129
					remove 109
					Expected output: number of collisions after removing 109 should be 2

18	openRemoveInvalidKey.insertKey(32);
	openRemoveInvalidKey.insertKey(52);
	openRemoveInvalidKey.insertKey(72);
	openRemoveInvalidKey.insertKey(92);
	collisions = openRemoveInvalidKey.removeKey(1);
		Private	Hard	10	Testing collisions with open addressing if key is not in HashTable:
					insert 32, 52, 72, 92
					remove 1
					Expected Output: number of collisions on trying to remove non-existent key 1 should be 1

19	openRemoveInvalidKey.insertKey(32);
	openRemoveInvalidKey.insertKey(52);
	openRemoveInvalidKey.insertKey(72);
	openRemoveInvalidKey.removeKey(32);
	collisions = openRemoveInvalidKey.removeKey(92);
		Private	Hard	10	Testing visits with open addressing in removeKey() if key one key is removed and the second key is not in HashTable:
					insert 32, 52, 72, remove 32,
					remove 92
					Expected Output: number of visits on trying to remove non-existent key 92 should be 4, which will happen if the symbols for EMPTY and REMOVED are different. Otherwise removeKey() could have missed the real key if it were present

				100	
		Quantity	Weight	Total	
	Easy	14	4	56	
	Medium	4	6	24	
	Hard	2	10	20	
	Total			100	