4. (25%) Suppose you have an empty hash table T with $m = 8$ slots that stores integer records (i.e. the record and its key are one and the same). The table resolves collisions by open addressing with double hashing, and its hash functions are $h_1(k) = (5 \cdot k) \bmod 8$ and $h_2(k) = (3 \cdot k) \bmod 7 + 1$. (*Disclaimer*: these are *not* good hash functions, just examples to help you reinforce your understanding of open-addressed hashing.)

   (a) Sketch diagrams showing the state of the hash table and give its load factor after *each* of the following series of calls:

   i. `T.insert(293)`

Index	Slot Value
0	
1	*
2	
3	
4	
5	
6	
7	

   Load Factor = $\frac{1}{8}$

   ii. `T.insert(30181)`

Index	Slot Value
0	
1	*
2	
3	
4	
5	
6	
7	*

   Load Factor = $\frac{2}{8}$

   iii. `T.insert(38388)`

Index	Slot Value
0	
1	*
2	
3	
4	*
5	
6	
7	*

   Load Factor = $\frac{3}{8}$

   iv. `T.insert(62)`

Index	Slot Value
0	
1	*
2	
3	
4	*
5	
6	*
7	*

Load Factor = $\frac{4}{8}$

v. `T.insert(3421)`

Index	Slot Value
0	
1	*
2	
3	*
4	*
5	
6	*
7	*

Load Factor = $\frac{5}{8}$

vi. `T.delete(293)`

Index	Slot Value
0	
1	`deleted`
2	
3	*
4	*
5	
6	*
7	*

Load Factor = $\frac{4}{8}$

(b) After the hash table T has performed the above operations, what is the sequence of slots in the table inspected by each of the following `find` operations before it returns?

i. `T.find(293)`
$h_1(293) = 1$, and $h_2(k) = 5$, so this operation will check, in order, Slots: $1 \to 6 \to 3 \to 0 \to 5 \to 2 \to 7 \to 4$. Because slot 1 is `deleted`, it will look at the next slot, and therefore return whatever is in slot 6.

ii. `T.find(38924)`
$h_1(38924) = 4$, and $h_2(38924) = 6$. Because Slot 4 is full from above, it will return the item stored in that slot The order of slots is: $4 \to 2 \to 0 \to 6 \to 4$....

iii. `T.find(62)`
$h_1(62) = 6$ and $h_2(62) = 5$. The order of slots is therefore: $6 \to 3 \to 0 \to 5 \to 2 \to 7 \to 4 \to 1$. Because slot 6 has an item, it will return whatever is in that slot.

(c) Just to show that the above table and hash function parameters aren't great, give an example of a key for which the `insert` operation could fail even if the table is not full. Explain why insertion could fail for this key.

We can use the example above, where will `T.insert(38924)` fail even if slots 1, 3, 5, and 7 are open if the remaining slots are filled. In this case $h_1(38924) = 4$, and $h_2(38924) = 6$. Insert will fail if slots 0, 2, 4, and 6 are filled because $h_2(38924)$ shares a common divisor with 8, 2. $h_1(38924) + i * h_2(38924)$ will form a cyclic subgroup of $\mathbb{Z} \bmod 8$, and thus will not reach every slot.