I do not have answers typed for all of these; your best use of these problems is to attempt them and discuss them with your study group; ideally, everyone in the study group will have attempted them individually first. If your answers are the same, this is evidence you are all correct. If they differ, attempt to solve the problem together, discussing how each decision is made. You will likely resolve the discrepancy *and* learn about how these work.

Furthermore, a good plan would be to do a few of the cuckoo hashing problems *before* you start to program project three, and as you work on them, begin to think about how the code would look for programming these. While the written problems are integers and the assignment is strings, the general strategy will be the same.

Quadratic Probing

For these problems, first insert the values, in the given order, into an initially empty hash table that uses quadratic probing for collision resolution. Do not resize the hash table, regardless of the load factor. After inserting the full set of values, attempt to find a value not inserted that now *cannot be* inserted without a resize/rehash happening. This illustrates why we rehash – note that the number of values we ask you to insert would violate a reasonable load factor limit.

For purposes of these exercises, quadratic probing is defined to mean that the ith choice is $(h(k) + i^2)\%m$, with the initial choice being $i = 0$.

1. Hash table size 13, values to insert are $\{38, 39, 26, 25, 16, 14, 22, 9\}$

0	1	2	3	4	5	6	7	8	9	10	11	12

2. Table size 13, values to insert are $\{25, 16, 14, 38, 39, 26, 22\}$

0	1	2	3	4	5	6	7	8	9	10	11	12

3. Hash table of size $m = 17$, insert the values $\{0, 48, 33, 16, 34, 2, 18, 9, 26\}$

0	1	2	3	4	5	6	7	8

9	10	11	12	13	14	15	16

4. Hash table of size $m = 17$, insert the values $\{20, 35, 13, 3, 26, 39, 37, 5, 30\}$

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	

5. Table size $m = 13$, insert $5, 26, 39, 7, 0, 30, 16$

0	1	2	3	4	5	6	7	8	9	10	11	12

6. Table size $m = 13$, insert $24, 5, 3, 37, 14, 18, 11$

0	1	2	3	4	5	6	7	8	9	10	11	12

7. Table size $m = 13$, insert $38, 9, 18, 20, 7, 33, 21$

0	1	2	3	4	5	6	7	8	9	10	11	12

8. Table size $m = 11$, insert $20, 23, 19, 34, 11, 30$

0	1	2	3	4	5	6	7	8	9	10

9. Table size $m = 11$, insert $1, 12, 7, 24, 32, 20$

0	1	2	3	4	5	6	7	8	9	10

10. Table size $m = 11$, insert $24, 29, 11, 6, 17, 13, 16$

0	1	2	3	4	5	6	7	8	9	10

Cuckoo Hashing

For the first three questions, the instructions were given as follows:

Suppose we have a Cuckoo Hash Table with each table having room for m entries each. Our hash functions are $h_0(x) = x\%m$ and $h_1(x) = (x/m)\%m$, where the / is integer division (floor of division; discard remainder). For example, $h_0(1289) = 2$ and $h_1(1289) = 7$ if $m = 11$. Do not resize/rehash, even if a long eviction cycle is seen.

After inserting the values, give an unsigned value you could attempt to insert into this table that would result in a need to rehash (due to an actual cycle). To get credit, your answer must be a positive integer and be no more than one billion.

1. $m = 11$, insert the following values:

x	890	1183	1189	158	561	366	8	279	1304
$h_0(x)$	10	6	1	4	0	3	8	4	6
$h_1(x)$	3	8	9	3	7	0	0	3	8

0	1	2	3	4	5	6	7	8	9	10

2. $m = 11$, insert the following values:

x	921	936	701	212	569	789	1042
$h_0(x)$	8	1	8	3	8	8	8
$h_1(x)$	6	8	8	8	7	5	6

0	1	2	3	4	5	6	7	8	9	10

For the remainder, the instructions were:

Suppose we have a Cuckoo Hash Table with each table having room for $m = 11$ entries each. Our hash functions are $h_0(x) = x\%11$ and $h_1(x) = (x/11)\%11$, where the / is integer division (floor of division; discard remainder). For example, $h_0(1289) = 2$ and $h_1(1289) = 7$. We will resize and rehash the table when an item *cannot* be inserted into the table – we have no *a priori* maximum eviction length.

We insert the following values, in the following order, into the table. Answer the following two questions about each. The exam also had the initially empty table available.

- For each of the first five elements inserted, are they in the upper array (indexed by h_0) or the lower array (indexed by h_1) after the first five elements (and only the first five elements) have been inserted? We asked students to circle a choice clearly.
- Which value will be the last one **inserted successfully** into the table prior to the rehash/resize? Recall that we are rehashing / resizing only after an element cannot be inserted; there is no *a priori* maximum eviction length as there was in project three.

3.

x	202	466	207	400	108	385	277	23	498	79	425	170	483
$h_0(x)$	4	4	9	4	9	0	2	1	3	2	7	5	10
$h_1(x)$	7	9	7	3	9	2	3	2	1	7	5	4	10

4.

x	207	400	202	108	466	61	423	122	34	216	246	80	230
$h_0(x)$	9	4	4	9	4	6	5	1	1	7	4	3	10
$h_1(x)$	7	3	7	9	9	5	5	0	3	8	0	7	9

5.

x	107	137	104	41	159	138	55	195	215	214	23	53	116	100	70	27
$h_0(x)$	8	5	5	8	5	6	0	8	6	5	1	9	6	1	4	5
$h_1(x)$	9	1	9	3	3	1	5	6	8	8	2	4	10	9	6	2

6.

x	159	41	137	107	104	72	47	82	136	122	146	81	61	66	118	111
$h_0(x)$	5	8	5	8	5	6	3	5	4	1	3	4	6	0	8	1
$h_1(x)$	3	3	1	9	9	6	4	7	1	0	2	7	5	6	10	10

7.

x	107	137	104	159	41	194	222	244	234	247	245	149	116	103	22	146
$h_0(x)$	8	5	5	8	5	7	2	2	3	5	3	6	6	4	0	3
$h_1(x)$	9	1	9	3	3	6	9	0	10	0	0	2	10	9	2	2

8.

x	41	159	137	107	104	123	86	75	58	96	64	91	97	76	23	41
$h_0(x)$	5	8	5	8	5	2	9	9	3	8	9	3	9	10	1	8
$h_1(x)$	3	3	1	9	9	0	7	6	5	8	5	8	8	6	2	3