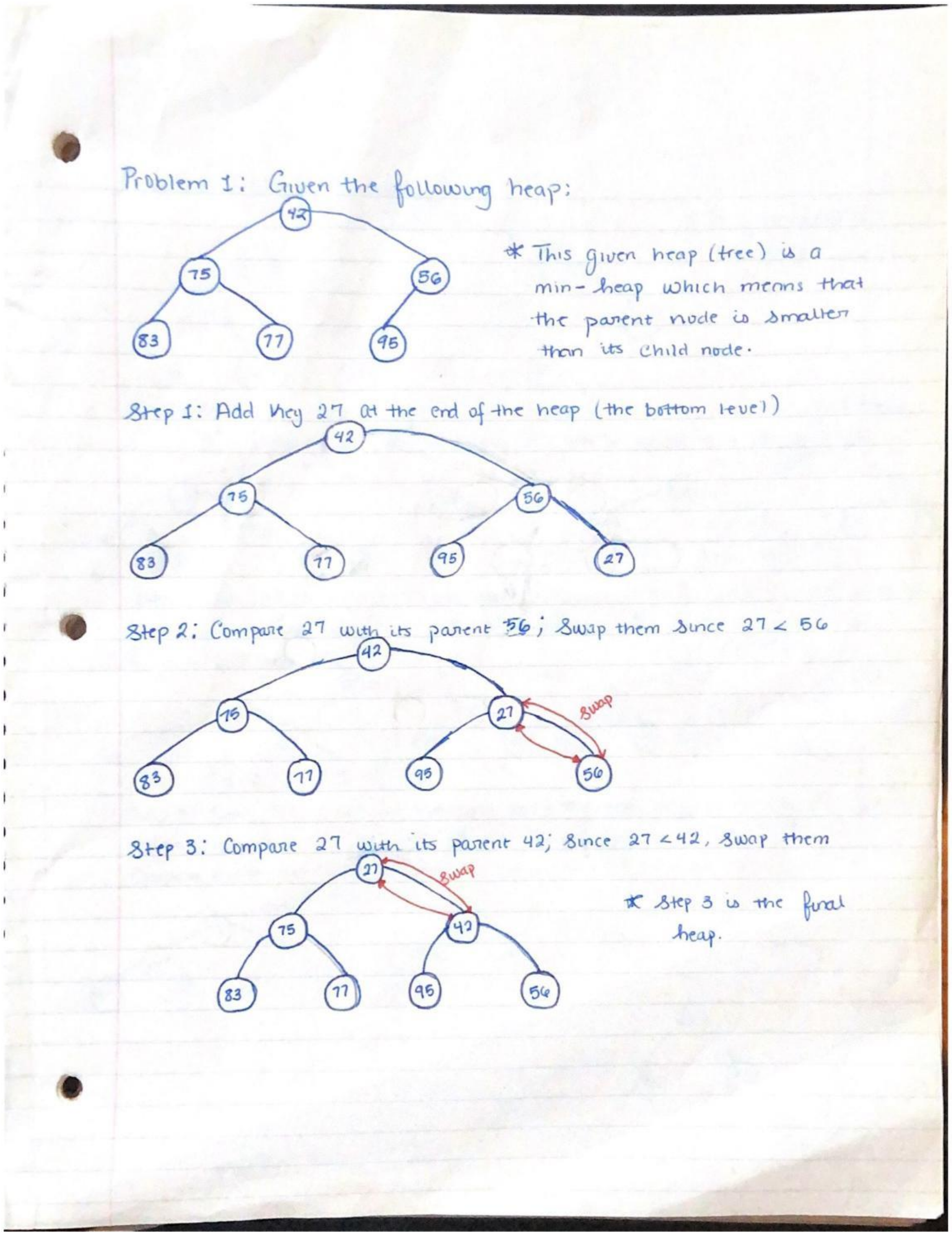
Elizabeth Oyebade

MET CS 526-O2

04/09/2022

Problem 1:



Problem 1: Given the following heap:

42
75
56
83
77
95

* This given heap (tree) is a min-heap which means that the parent node is smaller than its child node.

Step 1: Add key 27 at the end of the heap (the bottom level)

42
75
56
83
77
95
27

Step 2: Compare 27 with its parent 56; Swap them since 27 < 56

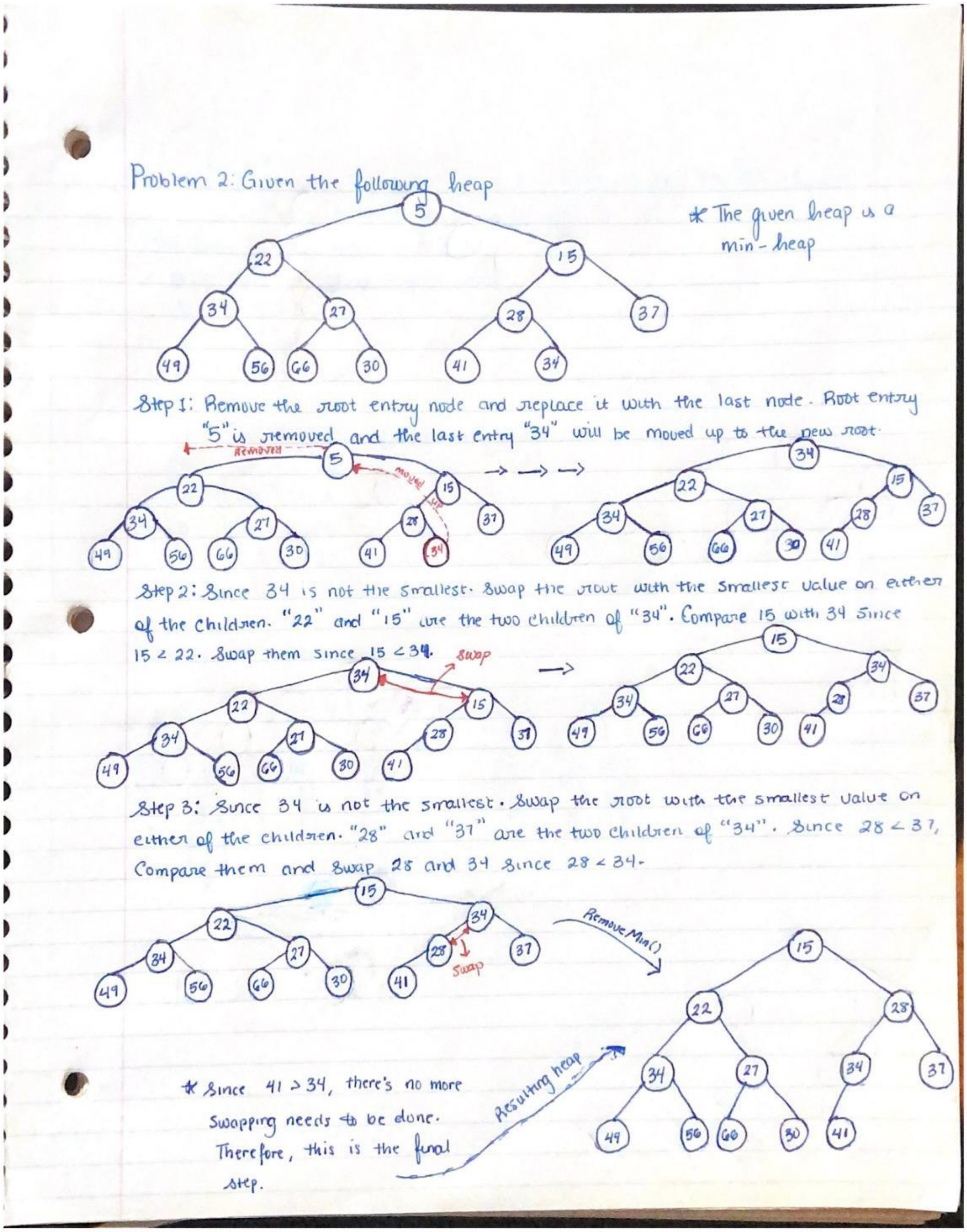
42
75
27
Swap
83
77
95
56

Step 3: Compare 27 with its parent 42; Since 27 < 42, Swap them

27
Swap
75
42
83
77
95
56

* Step 3 is the final heap.

Problem 2:



Problem 2: Given the following heap

* The given heap is a min-heap

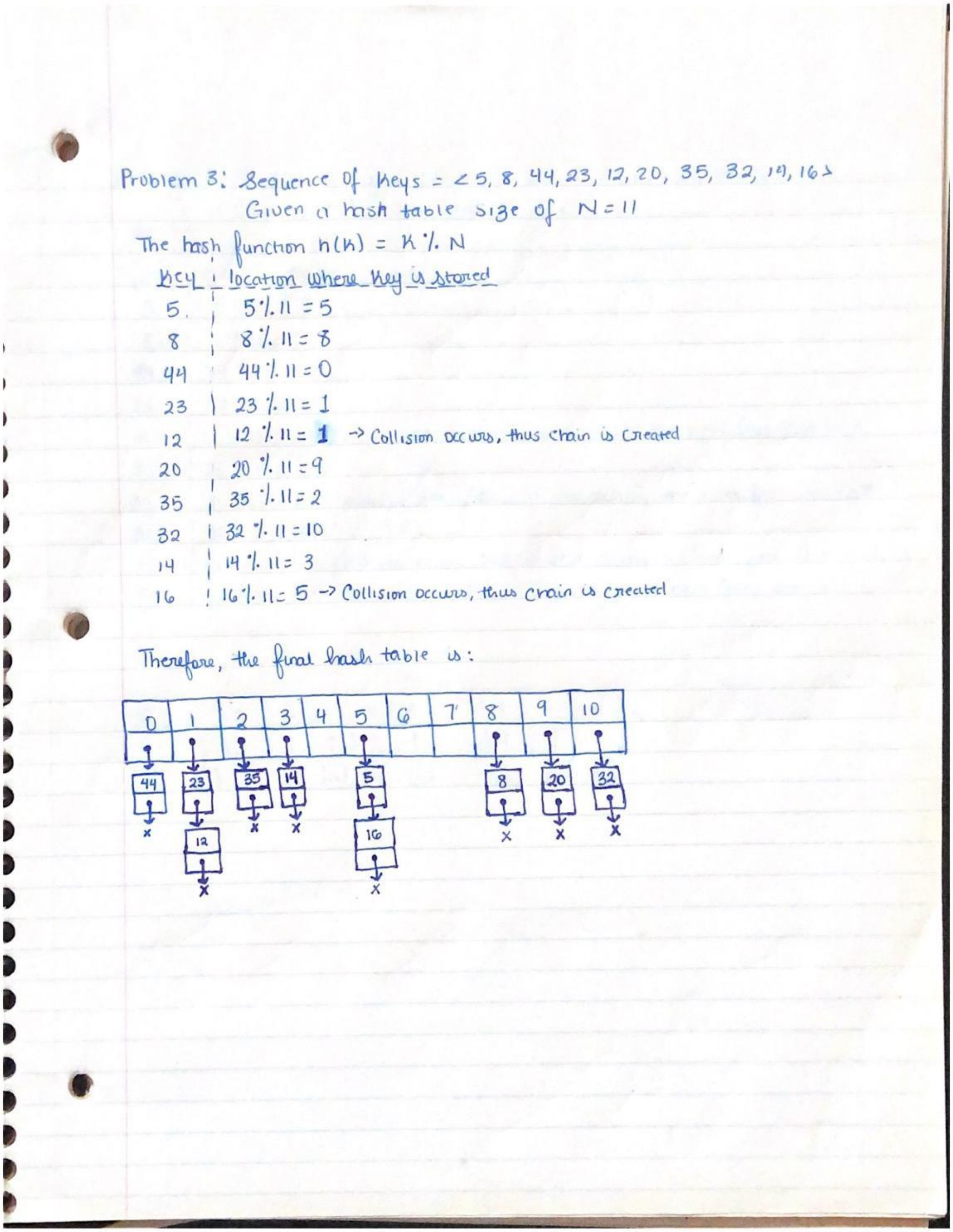
Step 1: Remove the root entry node and replace it with the last node. Root entry "5" is removed and the last entry "34" will be moved up to the new root.

Step 2: Since 34 is not the smallest. Swap the root with the smallest value on either of the children. "22" and "15" are the two children of "34". Compare 15 with 34 since 15 < 22. Swap them since 15 < 34.

Step 3: Since 34 is not the smallest. Swap the root with the smallest value on either of the children. "28" and "37" are the two children of "34". Since 28 < 37, Compare them and Swap 28 and 34 since 28 < 34.

* Since 41 > 34, there's no more swapping needs to be done. Therefore, this is the final step.

Problem 3:



Problem 3: Sequence of keys = <5, 8, 44, 23, 12, 20, 35, 32, 14, 16>
Given a hash table size of N = 11

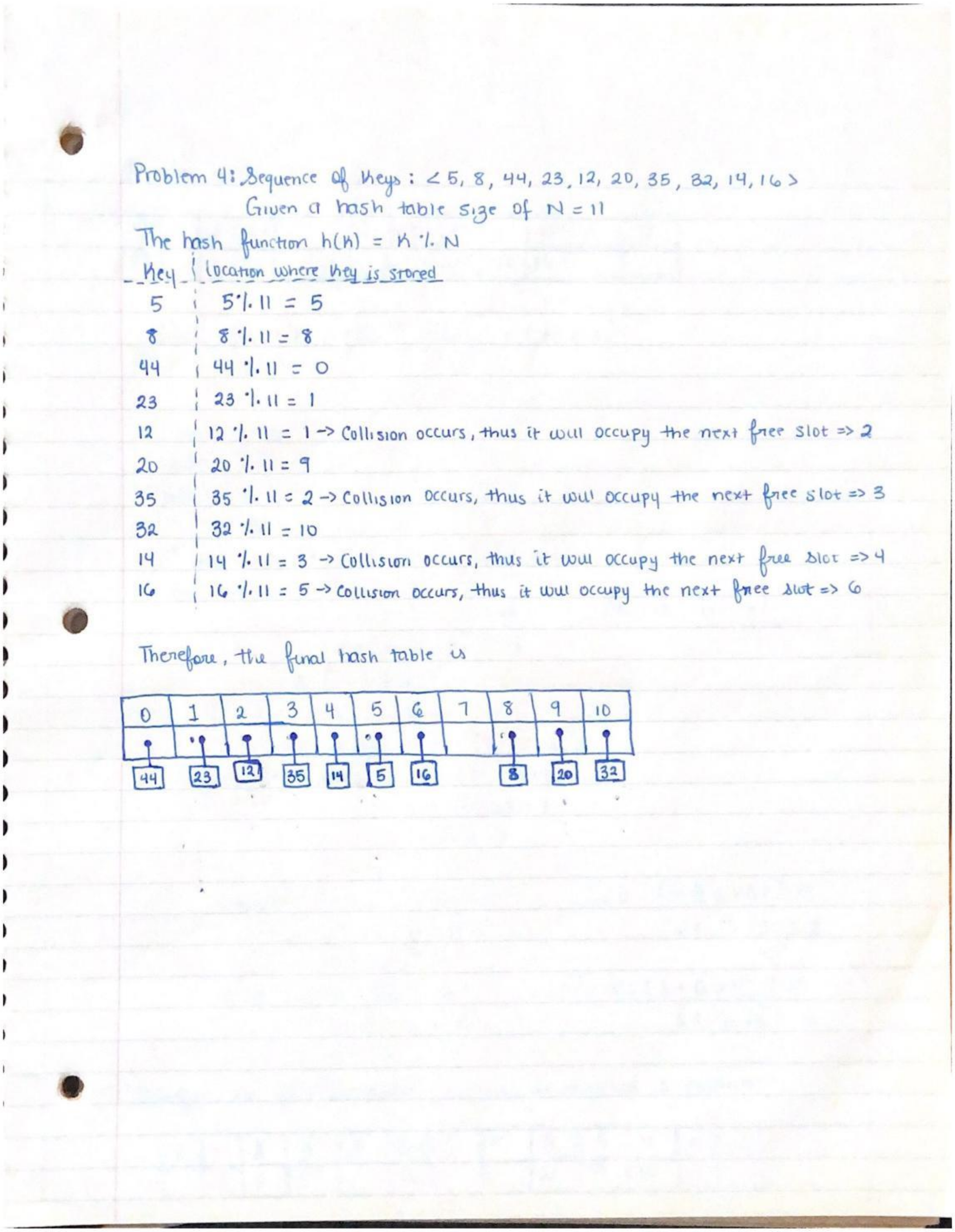
The hash function h(k) = k % N

key	location where key is stored
5.	5 % 11 = 5
8	8 % 11 = 8
44	44 % 11 = 0
23	23 % 11 = 1
12	12 % 11 = 1 -> Collision occurs, thus chain is created
20	20 % 11 = 9
35	35 % 11 = 2
32	32 % 11 = 10
14	14 % 11 = 3
16	16 % 11 = 5 -> Collision occurs, thus chain is created

Therefore, the final hash table is:

0	1	2	3	4	5	6	7	8	9	10
44 x	23 12 x	35 x	14 x		5 16 x			8 x	20 x	32 x

Problem 4:



Problem 4: Sequence of keys : < 5, 8, 44, 23, 12, 20, 35, 32, 14, 16 >
Given a hash table size of N = 11

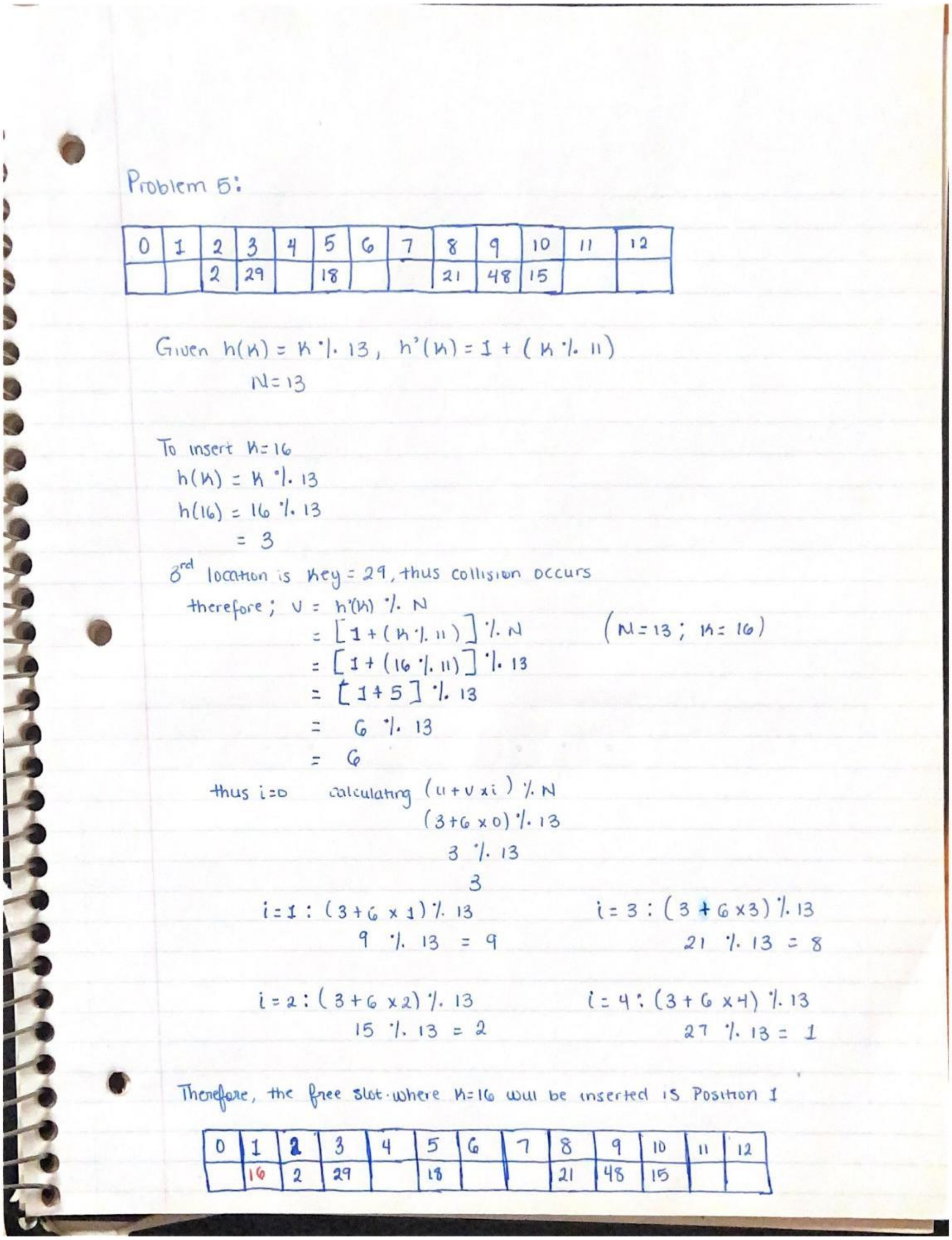
The hash function h(k) = k % N

Key	location where key is stored
5	5 % 11 = 5
8	8 % 11 = 8
44	44 % 11 = 0
23	23 % 11 = 1
12	12 % 11 = 1 -> Collision occurs, thus it will occupy the next free slot => 2
20	20 % 11 = 9
35	35 % 11 = 2 -> Collision occurs, thus it will occupy the next free slot => 3
32	32 % 11 = 10
14	14 % 11 = 3 -> Collision occurs, thus it will occupy the next free slot => 4
16	16 % 11 = 5 -> Collision occurs, thus it will occupy the next free slot => 6

Therefore, the final hash table is

0	1	2	3	4	5	6	7	8	9	10
44	23	12	35	14	5	16		8	20	32

Problem 5:



Problem 5:

0	1	2	3	4	5	6	7	8	9	10	11	12
		2	29		18			21	48	15		

Given $h(k) = k \% 13$, $h'(k) = 1 + (k \% 11)$

$N = 13$

To insert $k = 16$

$h(k) = k \% 13$

$h(16) = 16 \% 13$

$= 3$

3^{rd} location is key $= 29$, thus collision occurs

therefore; $v = h'(k) \% N$

$= [1 + (k \% 11)] \% N$ $(N = 13;\ k = 16)$

$= [1 + (16 \% 11)] \% 13$

$= [1 + 5] \% 13$

$= 6 \% 13$

$= 6$

thus $i = 0$ calculating $(u + v \times i) \% N$

$(3 + 6 \times 0) \% 13$

$3 \% 13$

3

$i = 1: (3 + 6 \times 1) \% 13$

$9 \% 13 = 9$

$i = 2: (3 + 6 \times 2) \% 13$

$15 \% 13 = 2$

$i = 3: (3 + 6 \times 3) \% 13$

$21 \% 13 = 8$

$i = 4: (3 + 6 \times 4) \% 13$

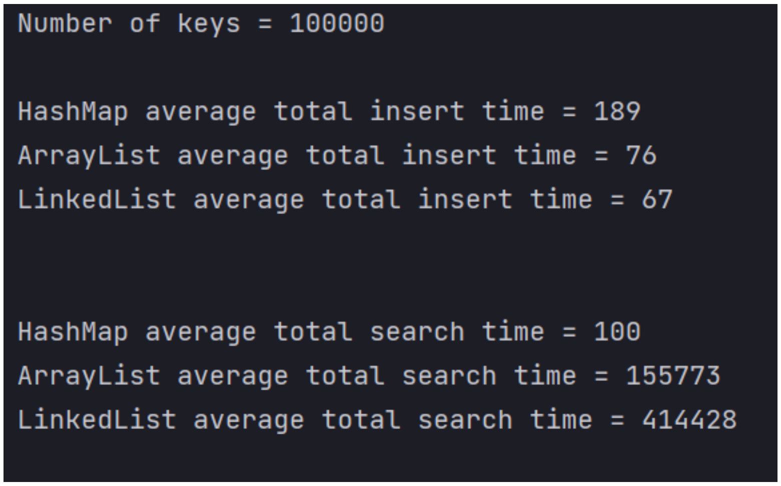
$27 \% 13 = 1$

Therefore, the free slot where $k = 16$ will be inserted is Position 1

0	1	2	3	4	5	6	7	8	9	10	11	12
	16	2	29		18			21	48	15		

Problem 6:

So, the result below shows that the HashMap has the largest insert time but it's the slowest in terms of its searching time. Although LinkedList is faster than the ArrayList and HashMap for the insert time, the searching time is almost 4000 times of HashMap. When comparing the total time, we can find that the HashMap can be used to store content with a large among of data. In closing, when the numbers of elements are not very large, the data structures will come out as a smaller difference. The search time for ArrayList or LinkedList is O(n) on average whereas searching a HashMap is O(1) on average.



```
Number of keys = 100000

HashMap average total insert time = 189
ArrayList average total insert time = 76
LinkedList average total insert time = 67


HashMap average total search time = 100
ArrayList average total search time = 155773
LinkedList average total search time = 414428
```