Hashing and Hash Tables

1. Suppose our hash function simply returns 17 every time it is called. Is such a hash function legal? If so, describe the effect of using it. If not, explain why.

2. Describe an algorithm to delete an element in a hash table using linear probing.

3. Describe the algorithm used to delete an element in a hash table using separate chaining.

4. Insert the keys `E A S Y Q U T I O N` in that order into an initially empty table of $M = 5$ lists, using separate chaining. Use the hash function $11k\%M$ to transform the kth letter of the alphabet into a table index. You do not need to resize the hash table as a part of your solution. Show only the final hash table after all insertions have been completed.

5. Insert the keys `E A S Y Q U T I O N` in that order into an initially empty table of size $M = 16$ using linear probing. Use the hash function $11k\%M$ to transform the kth letter of the alphabet into a table index. You do not need to resize the hash table as a part of your solution. Show only the final hash table after all insertions have been completed.

6 – 2	5 – 2	5 – 10
2 – 0	2 – 3	4 – 1
3 – 6	8 – 1	1 – 11
11 – 8	7 – 11	0 – 6
8 – 4	7 – 8	4 – 8
10 – 3	3 – 10	

Figure 1: A set of vertices connected within a graph

Undirected Graphs

6. When representing a graph in memory, when is it better to use an adjacency list? When is it better to use an adjacency matrix?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

7. Given the list of vertices and edges in Figure 1, draw the resulting graph.

8. Given the list of vertices and edges in Figure 1, write out the resulting adjacency list once the graph is loaded into memory.

9. Does breadth first search tell us anything about the distance from node v to node w when neither is at the root of the search?

10. A colleague suggests you to use a stack instead of a queue when running breadth first search. Would your colleague's suggestion still compute shortest paths in a non-edge-weighted undirected graph? Why or why not?