1. LinkedLists

   Consider the following python code:

```
class LinkedList():
   __slots__ = ( 'head', 'size' )

class Empty():
   __slots__ = ()

class NonEmpty():
   __slots__ = ( 'data', 'next' )
```

   (a) Write a function called 'insert' which takes three parameters: the linked list, a value to insert and the index in the linked list to insert it at.

   (b) What is the time complexity (Big-O) of this function?

2. Strings:

   (a) Write the Python code for a function called 'strcmp' to determine if two strings are equal. Do not simply compare the strings with the == command. Read the two strings from a file, one per line.

   (b) What is the time complexity of part (a)?

   (c) What are the values of the following operations?

```
myStr = "CS Final Review"
```

   i. myStr[0]
   ii. myStr[4]
   iii. myStr[15]
   iv. myStr[−1]
   v. myStr[: 6]
   vi. myStr[−4 :]
   vii. myStr[3 : 10 : 3]
   viii. myStr[9] + myStr[12 : 8 : −1]

3. Sorting:

   (a) Run insertion-sort on this list, $[3, 2, 4, 5, 1]$

4. Trees:

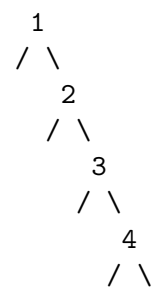
   (a) Construct a binary search tree, based on the list $[2, 4, 3, 1, 5]$.

   (b) What's the height of the tree?

   (c) How many leaf nodes are there?

   (d) What is the Big-O for finding a particular node in a perfectly balanced tree?

   (e) What is the Big-O for finding a particular node in the following poorly balanced tree?



5. Tuples, Lists:

   (a) What is wrong with the following?

```
data = [1,2,3]
data[0] = 3
data[-1] = 1
print(data)

data = (1,2,3)
data[0] = 3
data[-1] = 1
print(data)
```

   (b) What is the type and value of myVar in the following?

```
def myFunc(x):
   y = x + x
   z = x * x
   return y, z

myVar = myFunc( 5 )
```

   (c) What are the types and values of foo and bar?

```
foo, bar = myFunc( 5 )
```

6. Recursion

Write a function cross that takes two parameters S (length) and N (number of iterations) and draws the following:

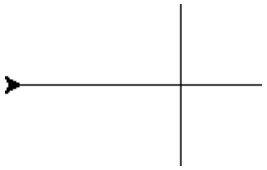
For N = 1, a line of length S

For N = 2, a line of length S and three lines of length S/2, two at right angles to the line and one in the same direction

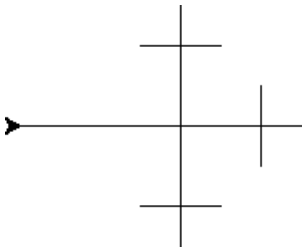
For N = 3, a line of length S, three lines of length S/2, two at right angles to the line and one in the same direction, and three more lines at each of the end points of these three lines, two at right angles and one in the same direction, as shown below:

N = 1

N = 2



N = 3



7. Binary Search

   Tell which values would be checked if the following binary searches were done on the following list:

```
lst = [ 0, 1, 4, 6, 7, 18, 20, 34, 39, 42, 50, 72, 77, 80, 99 ]
```

   (a) binarySearch(lst, 18)

   (b) binarySearch(lst, 34)

   (c) binarySearch(lst, 99)

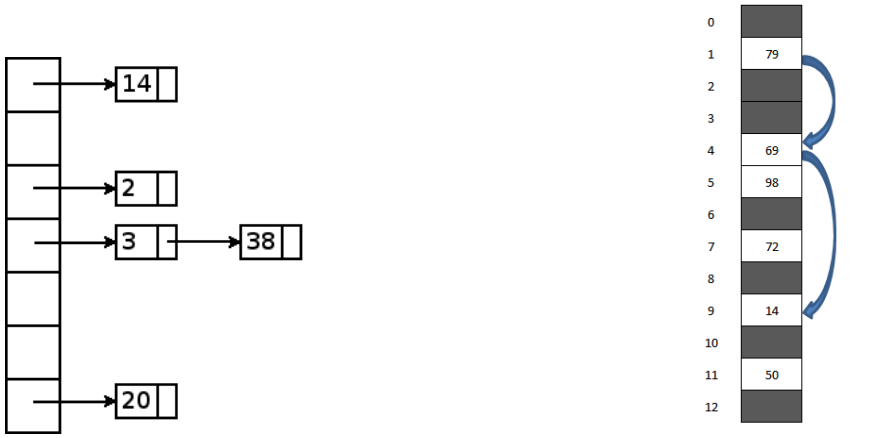
   (d) binarySearch(lst, 77)

   (e) binarySearch(lst, 999)

8. Classes and Maker Functions

   (a) Design a class called 'Pet' which can store the name, age, and species of the pet

   (b) Give the maker function for this class



(a) Hashtable A

(b) Hashtable B

Figure 1: Question 10b

9. Hashing

(a) Explain why the following hash function is not very useful. Recommend a solution.

```
def hashFn(s):
    """hashFn: string, int -> int"""
    value = ord(s[0]) - ord('a')
    return value % 4
```

(b) Label Hashtable A and Hashtable B with the type of chaining being used.

10. Sort complexities

    (a) List the best and worst case time complexities for the following sorting algorithms

        i. Mergesort

        ii. Quicksort

        iii. Heapsort

    (b) Sort the following list using quicksort. Show a substitution trace.
    [44, 2, 6, 99, 31, 63, 24, 1]

11. Heaps:

    (a) Show, using an array, the min-heap created by inserting elements in following order: [5, 9, 11, 7, 2, 8, 19, 21, 1]

    (b) Show, using an array, the resulting min-heap using the min-heap from a, after 2 removes.

12. Complete the following code for the backtracking solve algorithm. Be sure to include pruning in your solution.

```
def solve(board):
    """Backtracking algorithm."""
    if isGoal(board):
        return board
    else:
        for neighbor in generateNeighbors(board):









        return None
```