Question 1:

Linear search and Binary search aim to find the location of a specific value within a list of values (sorted list for binary search). The next level of complexity is to find two values from a list that together satisfy some requirement.

Propose an algorithm (in Python) to search for a pair of values in an unsorted array of *n* integers that are closest to one another. Closeness is defined as the absolute value of the difference between the two integers. Your algorithm should not first sort the list. [10 points]

```
def closest_pair(arr):
   n = len(arr)    pair = (0,0)
min_diff = float("inf")    for
x in range(n):       for y in
range(x+1, n):
         diff = abs(arr[x] - arr[y])
if diff < min_diff:
min_diff = diff              pair =
(arr[x], arr[y])
   return pair
```

Next, propose a separate algorithm (in Python) for a sorted list of integers to achieve the same goal. [10 points]

```
def closest_pair_sorted(arr):
   n = len(arr)    pair =
(0,0)    min_diff =
float('inf')    for x in
range(n-1):
      diff = abs(arr[x] - arr[x+1])
if diff < min_diff:             min_diff
= diff             pair = (arr[x],
arr[x+1])
   return pair
```

Briefly discuss which algorithm is more efficient in terms of the number of comparisons performed. A formal analysis is not necessary. You can simply state your choice and then justify it. [5 points]

The second algorithm is more efficient in terms of the number of comparisons performed, although it is limited to sorted lists. The first algorithm has to check each number against one another, whereas the second one only needs to check each number with the one in front of it.

Question 2:

Implement in Python an algorithm for a level order traversal of a binary tree. The algorithm should print each level of the binary tree to the screen starting with the lowest/deepest level on the first line. The last line of output should be the root of the tree. Assume your algorithm is passed the root node of an existing binary tree whose structure is based on the following BinTreeNode class. You may use other data structures in the Python foundation library in your implementation, but you may not use an existing implementation of a Binary Tree or an existing level order traversal algorithm from any source.

Construct a non-complete binary tree[1] of at least 5 levels. Call your level order traversal algorithm and show that the output is correct. [20 points]

```
class BinTreeNode:         def __init__(self, value=0,
left=None, right=None):              self.value = value
self.left  = left
        self.right = right
```

```
class BinTreeNode:    def __init__(self, value=0,
left=None, right=None):
    self.value = value
self.left = left
    self.right = right

def print_tree(root):

  def tree_height(node):
if not node:          return 0
    left_height = tree_height(node.left)
right_height = tree_height(node.right)       return
max(left_height, right_height) + 1

  height = tree_height(root)
  tree_levels = [[] for _ in range(height)]

  def tree_traversal(node, level):
if not node:
       return
tree_levels[level].append(node.value)        if
node.left:
       tree_traversal(node.left, level - 1)       if
node.right:          tree_traversal(node.right,
level - 1)

  tree_traversal(root, height - 1)

  for level in reversed(tree_levels):
print(level)

if __name__ == "__main__":    root =
BinTreeNode(1)    root.left =
BinTreeNode(2)    root.right =
BinTreeNode(3)    root.left.left =
BinTreeNode(4)    root.left.right =
BinTreeNode(5)    root.right.right =
BinTreeNode(6)    root.left.left.left =
BinTreeNode(7)    root.left.right.right =
BinTreeNode(8)    root.right.right.left =
BinTreeNode(9)
```

[1] A complete binary tree is a binary tree where every level except possibly the last level is full, meaning no additional nodes could fit on that level.

```
root.left.left.left.left = BinTreeNode(10)

print_tree(root)
```

Question 3:

Fill out your profile on Slack including a clear head shot. This will help the TAs and I, as well as you all, associate names with faces quicker. Paste a screen shot of your completed Slack profile below. [5 points]