Question 1. [5 MARKS]

Let A and B be the Assignment 1 part 2 'sparse matrix' representations of the following matrices (the 'default' value is set to 0):

```
A                    B
-3   0  -5   0       3   0   5   0
-4   1   0   0       4   1   0   0
 0   0   0   0       0   0   0   7
 6   1   3   0       6   1   3   0
```

Part (a) [2 MARKS]

Draw a diagram to represent how matrix A would be represented in memory.

Part (b) [1 MARK]

Draw a diagram to illustrate how the object returned by `A.get_col(2)` would be represented in memory.

Matrices repeated from previous page:

A

```
-3   0  -5   0
-4   1   0   0
 0   0   0   0
 6   1   3   0
```

B

```
3   0   5   0
4   1   0   0
0   0   0   7
6   1   3   0
```

Part (c) [2 MARKS]

Draw a diagram to illustrate how the object returned by `A.add_matrix(B)` would be represented in memory.

Question 2. [15 MARKS]

A list of integers L is said to be `nearly sorted` iff there is at most one integer value i such that: `0 < i < len(L) and L[i-1] > L[i]`.

Examples of lists that are nearly sorted include:
`[]`, `[6]`, `[1,2]`, `[2,1]`, `[1,1,2,3,5,8]` and `[9,6,7,11,11]`.
Examples of lists that are not nearly sorted include:
`[3,2,1]` and `[7,2,6,3]`.

Nick wrote two functions which, given a list of integers L, return whether L is nearly sorted. One function is called `is_nearly_sorted` and uses a loop, while the other function is called `is_nearly_sorted_rec` and uses a recursive helper function called `is_nearly_sorted_helper`.

Then ... (you guessed it) ... the CODE MANGLER struck!

- The function headers remained.
- All comments were erased.
- All the code in `is_nearly_sorted_rec` was also erased.
- The remaining code was unindented and mixed up together. Then duplicate lines were removed. (i.e., each line that remains may have been used more than once).

The remaining code can be found on the back of this page (an additional copy can be found on the final page of this exam, which can be detached). Your job is to restore the code to its original state in the space provided on the following page.

```
d = 1
d = d - 1
else:
i = 1
i = i + 1
if L[i-1] > L[i]:
if i < len(L):
result = (d >= 0)
result = True
result = is_nearly_sorted_helper(L, i+1, d)
result = is_nearly_sorted_helper(L, i+1, d-1)
return result
while (i < len(L) and result):
```

```
def is_nearly_sorted(L):
```

```
def is_nearly_sorted_helper(L, i, d):
```

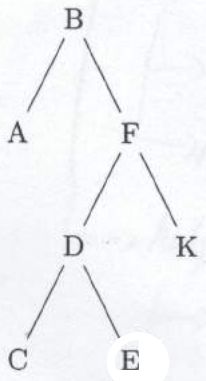
```
def is_nearly_sorted_rec(L):
```

```
class BinTreeNode:
    """
    A node in a binary tree.
    """

    def __init__(self, data, left=None, right=None):
        """ (BinTreeNode, str, BinTreeNode, BinTreeNode) -> NoneType

        Initialize a new BinTreeNode with data, left and right children.
        """

        self.data = data
        self.left = left
        self.right = right

def helper(root, n, level=0):
    """ (BinTreeNode, int) -> (BinTreeNode, int)

    REQ: n > 0.
    """

    if root == None:
        (node, k) = (None, n)
    else:
        (node, k) = (root, n-1)
        if k > 0:
            (node, k) = helper(root.left, k)
        if k > 0:
            (node, k) = helper(root.right, k)
    return (node, k)

def funky(root, n):
    """ (BinTreeNode, int) -> str

    REQ: n > 0.
    """

    (node, k) = helper(root, n)
    if k == 0:
        result = node.data
    else:
        result = None
    return result
```

Question 3. [15 MARKS]

Nick wrote a really cool function that does something interesting with binary trees. The function, along with a helper function it uses is given on the previous page (an additional copy is given on the detachable page on the back of this exam). Unfortunately, Nick didn't heed his own advice about commenting, and now he can't actually remember what his function does.

Let `bt_root` be the root node of the following tree:

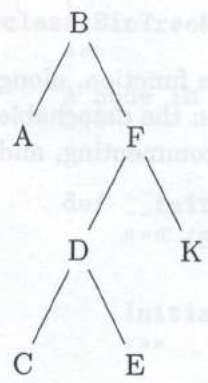


Part (a) [4 MARKS]

Provide a trace of `funky(bt_root, 1)` including what it returns.

Part (b) [4 MARKS]

Provide a trace of `funky(bt_root, 4)`, including what it returns.


B
A
F
D
K
C
E

Part (c) [4 MARKS]

Provide a trace of `funky(bt_root, 7)`, including what it returns.

Part (d) [3 MARKS]

Provide a description for `funky`