A. Short Answer:

(1) What are the values of the following expressions?

\[
[6, 9, 2, 5, 4, 3, 1, 8][(2+1):(10-13)]
\]

\[
"banana"[1].upper() \text{ in } "ABCDEFG"
\]

\[
[[1, 0, 1], [0, 1, 0], [2]][1][0]
\]

(2) Explain what the \texttt{is} operator does.

(3) Define the term \texttt{mutability} as it relates to data structures.

(4) What is the purpose of the \texttt{call stack}?
B. Comprehension:

(1) What does the code to the right print?

```python
L = [1,2,3,4,5]
L.append(L[3]+3)
L.insert(2,L[2])
L[L[0]+L[1]] = L[2+3]
L[L.pop(2)] = 3
print(L)
```

(2) What does the code to the right print?

```python
D={'a':'b','c':'d','b':'e','e':'f'}
L = []
for k in D:
    if D[k] in D:
        L.append(k)
    else:
        L.append(D[k])
for i in range(len(L)-1):
    D[L[i]] = L[i+1]
print(D)
```

(3) What does the code to the right print?

```python
L = [[6,8,2],[7,4,2],[2,1,0],[3,4,7]]
for i in range(0,len(L)):
    for j in range(0,len(L[i])):
        if L[i][j]%3==0:
            L[i][j] += 1
        elif i>0 and L[i][j] > L[i-1][j]:
            L[i][j] -= L[i-1][j]
        elif i<len(L)-1 and L[i+1][j]<=3:
            L[i][j] *= L[i+1][j]
        else:
            L[i][j] //= len(L[i])
for e in L:
    print(e)
```
C. Programming:

(1) Write a function called userWords() that repeatedly asks the user to enter words until they enter the word "stop!". Every word the user enters should be stored in a nested list arranged by word-length such that all the 3-letter words are stored in a list at index 3 (for example). Return the nested list when complete. You may assume no word is larger than 9 letters.

```python
def userWords():
    word ="
    wordList = [[], [], [], [], [], [], [], [], [], []]
    while word!="stop!"
        word = input("Enter a word (stop!) to stop: ")
        wordList[len(word)].append(word)
    return wordList
```

(2) Write a function called qWords() that takes a 2D list of words as argument (e.g., such as the one returned from the previous problem), and returns a 1D list of all the words that start with the letter 'q' from the input list.

```python
def qWords(L):
    out = [
        for lis in L:
            for e in lis:
                if e[0].lower() == 'q':  #.lower() not required
                    out.append(e)
    return out
```
(3) Write a recursive function called `search()` that takes a list and a key as arguments and returns the index of the key in that list if it exists, and -1 if not. E.g. `search([5,3,2,4,1],3) → 1`

```python
def search(lst, key):
    if len(lst) == 0:
        return -1
    elif lst[0] == key:
        return 0
    else:
        result = search(lst[1:], key)  # recurse
        if result == -1:
            # not found below
        else:
            return result + 1  # found below, so count back up
```

(4) Write a function called `copyEvenLines()` that takes two filenames as input and copies every other line from the first file into the second. That is, counting from 0, the 0\textsuperscript{th} line, the 2\textsuperscript{nd} line, the 4\textsuperscript{th} etc… You may assume that the first file exists and that the second does not.

```python
def copyEvenLines(inFile, outFile):
    f = open(inFile, 'r')
    lines = f.read().splitlines()
    f.close()
    f = open(outFile, 'w')
    for i in range(0, len(lines), 2):
        f.write(lines[i])
    f.close()
```
(5) Write a function called `mutateList()` that takes a nested list of integers as argument and performs the following mutations: any even integers get divided by 2, any odd integers less than 10 get multiplied by 10, and any row (inner list) with less than 5 elements should be padded with zeroes to exactly 5 elements.

E.g. `mutateList([[8,5,7,2],[13,12,9,17,5],[10,20]])`  
→ `[[4,50,70,1,0],[13,6,90,17,50],[5,0,0,0,0]]`