Part 1 Time limit: 15 mins

Multiple Choice [36 pts]

The next 2 questions are based on the `takeStep` function below that we wrote in lecture for the Snake Game Case Study. As a reminder, `app.snake` is a list of `(row, col)` tuples representing the row and column of each dot in the snake.

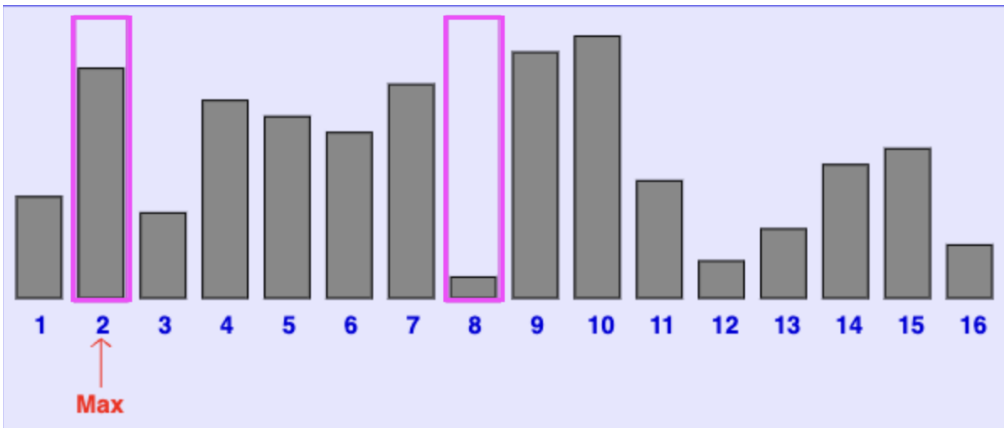
```
def takeStep(app):
    headRow, headCol = app.snake[0]
    drow, dcol = app.direction
    newHeadRow, newHeadCol = headRow + drow, headCol + dcol
    if (isOnBoard(app, newHeadRow, newHeadCol) and
        (newHeadRow, newHeadCol) not in app.snake):
        _________ # BLANK 1
        if (newHeadRow, newHeadCol) == app.foodPosition:
            placeFood(app)
        else:
            _________ # BLANK 2
    else:
        app.gameOver = True
```

1. What code should go in the first blank? [2 points]

A. `app.snake += (newHeadRow, newHeadCol)`
B. `app.snake.insert(0, (newHeadRow, newHeadCol))`
C. `app.snake.append(0, (newHeadRow, newHeadCol))`
D. `app.snake.append((newHeadRow, newHeadCol))`

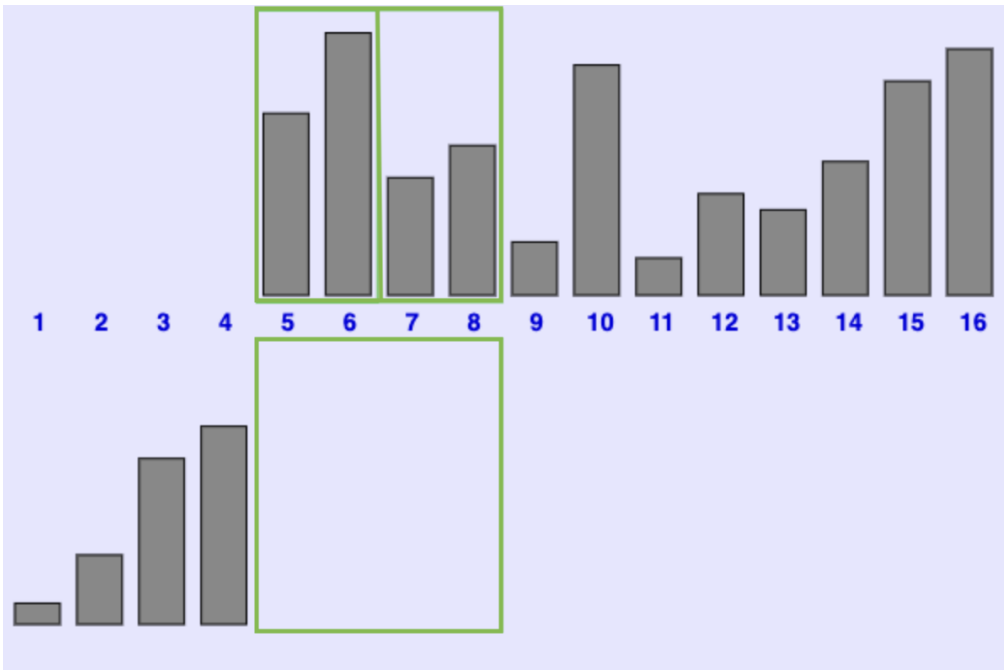
2. What code should go in the second blank? [2 points]

A. `app.snake.remove(0)`
B. `app.snake.remove((headRow, headCol))`
C. `app.snake.pop(0)`
D. `app.snake.pop()`



3. Above is an image of xSortLab running selection sort. The pink highlighted boxes indicate the two elements currently being compared. Given this state, what are the indices of the next two values to be *swapped*?

A. 2 and 1
B. 2 and 8
C. 2 and 16
D. 10 and 1
E. 10 and 8
F. 10 and 16



4. Above is an image of xSortLab running merge sort. The green highlighted blocks are the two blocks that are about to be merged.Given this state, what are the indices of the next two values to be *compared*?

A. 5 and 6
B. 5 and 7
C. 5 and 9
D. 7 and 8
E. 7 and 9

5. How many passes does selection sort do to sort a list? Assume the list has length N.

A. O(1)
B. O(logN)
C. O(N)
D. O(NlogN)
E. $O(N^2)$

6. How many steps does each pass in selection sort do? Assume the list has length N.

A. O(1)
B. O(logN)
C. O(N)
D. O(NlogN)
E. $O(N^2)$

7. How many passes does merge sort do to sort a list? Assume the list has length N.

A. O(1)
B. O(logN)
C. O(N)
D. O(NlogN)
E. $O(N^2)$

8. How many steps does each pass in merge sort do? Assume the list has length N.

A. O(1)
B. O(logN)
C. O(N)
D. O(NlogN)
E. $O(N^2)$

9. What is the efficiency of linear search on a list of length N?

A. O(1)
B. O(logN)
C. O(N)
D. O(NlogN)
E. $O(N^2)$

10. What is the efficiency of binary search on a list of length N? You may assume it is being run on a valid input list.

A. O(1)
B. O(logN)
C. O(N)
D. O(NlogN)
E. $O(N^2)$

11. If we wish to find whether an item exists in a very large unsorted list of length N, which of the following options would we expect to be fastest in the worst case?

A. Sorting the list with selection sort, then performing a binary search
B. Sorting the list with merge sort, then performing a binary search
C. Sorting the list with merge sort, then performing a linear search
D. Performing linear search on the unsorted list

12. What does O(555NlogN + N2 - 10000N) simplify to?**

A. O(1)
B. O(logN)
C. O(N)
D. O(NlogN)
E. $O(N^2)$
F. $O(N^3)$

13. Say on a given computer, a function that runs in O(N2) takes 5 seconds to run on a list with 100 values. Roughly how long would we expect it to take to run on the same computer for a list with 300 values?**

A. 5 seconds
B. 10 seconds
C. 15 seconds
D. 30 seconds
E. 45 seconds
F. 90 seconds

14. What is the efficiency of the following function? Assume the list L has length N.

```
def f(L):
    M = []
    for v in L:
        M.insert(0, v)
    for v in L:
        M.append(v)
    return M
```

A. O(1)
B. O(logN)
C. O(N)
D. O(NlogN)
E. $O(N^2)$
F. $O(N^3)$
G. $O(2^N)$

15. What is the efficiency of the following function? Assume the list L has length N.

```
def f(L):
    s = set(L)
    for i in range(len(L)):
        if i in s:
            s.add(L[i])
    return s
```

A. O(1)
B. O(logN)
C. O(N)
D. O(NlogN)
E. $O(N^2)$
F. $O(N^3)$
G. $O(2^N)$

Code Tracing [15 pts]

What does the following code print?

```
def ct(d):
    t = ''
    for i in range(4):
        s = d.get(i, set())
        for v in s:
            t += v
        t += ':'
        if (i > 1) and (i+1 not in d):
            d[i+1] = s & {'c', 'd'}
    return t

d = {0 : {'a', 'c'},
     2 : {'a', 'c', 'e'}}
print(ct(d))
```

Answer:

Reasoning Over Code [15 pts]

Find an argument for the function `roc(d)` that makes it return `True`.

```
def roc(d):
    res = dict()
    for key in d:
        val = d[key]
        if val in res:
            res[val].add(key)
        else:
            res[val] = {key}
    return res == { 5 : {'a', 'c'}, 7 : {'b'}}
```

Answer: