Names: ________________________________ Worksheet #3

CS Logins: ____________________________ Expanding Stacks and Queues

Activity 1: Pseudocode for a Capped-capacity Stack

Write pseudocode for the functions `isEmpty()`, `push(obj)`, and `pop()` for a capped-capacity stack. Assume your stack has the following constructor and `size()` functions. Write the big-O runtime on each operations.

```
Stack():                         O(      )          function push(obj):      O(     )
    data = array of size 20
    count = 0

function size():                 O(1)
    return count

function isEmpty():              O(       )         function pop():          O(     )
```

What should happen if the user tries to push to a stack that is at full capacity? What about when someone tries to pop from an empty stack?

Activity 2: Expanding Stack - Analysis of Incremental Strategy

Based on the calculations in lecture of the number of operations per push for 5, 10, and 15 pushes, using an incremental expansion strategy where c = 5, what would be the average number of operations per push for **20 pushes**?

Names: ________________________________ Worksheet #4

CS Logins: ____________________________ Expanding Stacks and Queues

Activity 1: Pseudocode for a Capped-capacity Stack

Write pseudocode for the functions `isEmpty()`, `push(obj)`, and `pop()` for a capped-capacity stack. Assume your stack has the following constructor and `size()` functions. Write the big-O runtime on each operations.

```
Stack():                         O(      )          function push(obj):      O(     )
    data = array of size 20
    count = 0

function size():           O(1)
    return count

function isEmpty():              O(       )         function pop():          O(     )
```

What should happen if the user tries to push to a stack that is at full capacity? What about when someone tries to pop from an empty stack?

Activity 2: Expanding Stack - Analysis of Incremental Strategy

Based on the calculations in lecture of the number of operations per push for 5, 10, and 15 pushes, using an incremental expansion strategy where c = 5, what would be the average number of operations per push for **20 pushes**?

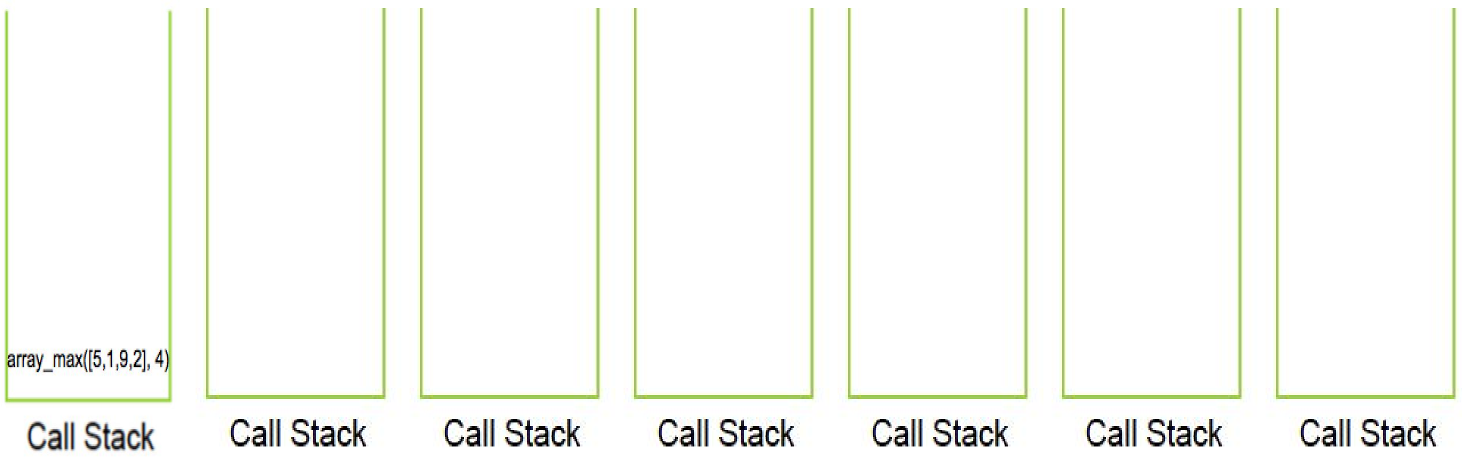
Activity 3: Recursive array_max

Draw out the call stack for each recursion of array_max([5, 1, 9, 2], 4). When you reach the base case and the function returns, write the return value. Continue to write the return value as you pop calls off the stack. Put “N/A” for the non-base-case “return:” values. The first one is done for you!

```
# Returns the maximum value of the first n elements in the array
# Example: array_max([5,1,9,2], 4) → 9

def array_max(array, n):
   if n == 1:
      return array[0]
   else:
      return max(array[n-1], array_max(array, n-1))
```

1.Returns: N/A 2. Returns: 3. Returns: 4. Returns: 5. Returns: 6. Returns: 7. Returns:



Activity 3: Recursive array_max

Draw out the call stack for each recursion of array_max([5, 1, 9, 2], 4). When you reach the base case and the function returns, write the return value. Continue to write the return value as you pop calls off the stack. Put “N/A” for the non-base-case “return:” values. The first one is done for you!

```
# Returns the maximum value of the first n elements in the array
# Example: array_max([5,1,9,2], 4) → 9

def array_max(array, n):
   if n == 1:
      return array[0]
   else:
      return max(array[n-1], array_max(array, n-1))
```

1.Returns: N/A 2. Returns: 3. Returns: 4. Returns: 5. Returns: 6. Returns: 7. Returns: