Algorithms HW3 *Summer 19'* Tae Myles

1. A = [6, 8, 9, 10, 12, 16, 15, 13, 14, 19, 18, 17]
   A = [6, 8, 9, 10, 12, 16, 15, 13, 14, 19, 18, 17, 6]
   A = [6, 8, 9, 10, 12, 6, 15, 13, 14, 19, 18, 17, 16]
   A = [6, 8, 6, 10, 12, 9, 15, 13, 14, 19, 18, 17, 16]

2. a) On another PDF file

   b) <u>Explain what the first base case that the algorithm checks for is, in plain English:</u>
      It checks to see if initial index is equal to the last index. If its equal, we know that there is only one element, hence it's the kth smallest number.

   <u>List the steps that the algorithm will execute if the input happens to be this base case:</u>
      1. Check if p = r
      2. return A[p]

   <u>Complete the recurrence relation using actual constants:</u>
      T(first base case) = 7

   <u>Explain what the second base case that the algorithm checks for is, in plain English:</u>
      It checks for the kth smallest number, it does this by checking if k is equal to the Pivot. If it is, it returns that kth smallest number.

   <u>List the steps that the algorithm will execute if the input happens to be this base case:</u>
      1. Check if p = r, which fails.
      2. Partition the array and set that equal to q.
      3. Let pivotDistance equal to q-p+1.
      4. Check if k is equal to pivotDistance. If so, return the kth smallest number.
      5. Return A[q].

   <u>Complete the recurrence relation using actual constants (assume complexity of Partition to be 20n):</u> T(Second base case) = 20n + 113

   <u>List the steps that the algorithm will execute if the input is not a base case:</u>
      1. Check if p = r, which fails
      2. Partition the array and set that equal to q.
      3. Let pivotDistance equal to q-p+1.
      4. Check if k is equal to pivotDistance. If so, return the kth smallest number.
      5. Check if k < pivotDistance. If so, return Quickselect(A,p,q-1,k).
      6. If previous statement is false then return Quickselect(A,q+1,r,k-pivotDistance)

   <u>Complete the recurrence relation using actual constants (assume complexity of Partition to be 20n and the worst case input size for the recursive call):</u>
   T(n) = (n-1)(20n + T(n-1) + 8) + 7

How will the above recurrence change if you instead assume the best case input size for the recursive call):
T(n) = 7

3. A: [19, 6, 10, 7, 16, 17, 13, 14, 12, 9]
C: [0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1]
C: [0, 0, 0, 0, 0, 0, 1, 2, 2, 3, 4, 4, 5, 6, 7, 7, 8, 9, 9, 10]
B: [6, 7, 9, 10, 12, 13, 14, 16, 17, 19]

4. Original: [456**7**, 321**0**, 234**5**, 432**1**, 567**8**]
Pass 1: [32**1**0, 43**2**1, 23**4**5, 45**6**7, 56**7**8]
Pass 2: [3**2**10, 4**3**21, 2**3**45, 4**5**67, 5**6**78]
Pass 3: [**3**210, **4**321, **2**345, **4**567, **5**678]
Pass 4: [2345, 3210, 4321, 4567, 5678]

5. Bucket0: [0...1/15)
Bucket1: [1/15...2/15)
Bucket2: [2/15...3/15)
Bucket3: [3/15...4/15)
Bucket4: [4/15...5/15)
Bucket5: [5/15...6/15)
Bucket6: [6/15…7/15)
Bucket7: [7/15…8/15)
Bucket8: [8/15…9/15)
Bucket9: [9/15…10/15)
Bucket10: [10/15...11/15)
Bucket11: [11/15…12/15)
Bucket12: [12/15…13/15)
Bucket13: [13/15…14/15)
Bucket14: [14/15…15/15)

Bucket 0: [0…1/n)
Bucket 1: [1/n…2/n)
Bucket n-2: [n-2/n…n-1/n)
Bucket n-1: [n-1/n…n/n)

6. A)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	1	14	3	3	3	1	1	8	3	3	3	3	1	14	18	16	19	20	1

B)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-3	1	14	3	3	3	1	1	8	3	3	3	3	1	14	1	16	16	16	16

C)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	1	-20	3	3	3	1	1	8	3	3	3	3	3	14	3	16	16	16	16

D)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	1	14	3	3	3	1	1	8	3	3	3	3	1	14	1	1	1	1	1

7. On another PDF file