1. (10 points) For parts (1a) and (1b), justify your answers in terms of deterministic QuickSort, and for part (1c), refer to Randomized QuickSort. In both cases, refer to the versions of the algorithms given in lecture (you can refer to the moodle lecture notes).

   (a) What is the asymptotic running time of QuickSort when every element of the input A is identical, i.e., for $1 \leq i, j \leq n$, $A[i] = A[j]$?

   (b) Let the input array $A = [9, 7, 5, 11, 12, 2, 14, 3, 10, 6]$. What is the number of times a comparison is made to the element with value 3?

   (c) How many calls are made to `random-int` in (i) the worst case and (ii) the best case? Give your answers in asymptotic notation.

(a):

When every element of the input A is identical. We will get the worst-case running time which would be: $\Theta(n)^2 \Longleftarrow$
The condition we check is never true when the input is identical. So we have to do the maximum number of partitions.

(b):

We will get 3 comparisons. This is based on the algorithm in CLRSe3, which chooses the right most element as the pivot. Doing another algorithm will change our result.

Using the Logarithmic behavior in quick-sort, and using the 10 elements of our array gives us: $log_2(10) = 3.32 = \approx 3$

$A = [9, 7, 5, 11, 12, 2, 14, 3, 10, 6]$ (Our pivot is 6)
$A = [5, 7, 9, 11, 12, 2, 14, 3, 10, 6]$
$A = [5, 2, 9, 11, 12, 7, 14, 3, 10, 6]$ (1 comparison)
$A = [5, 2, 3, 11, 12, 7, 14, 9, 10, 6]$
$A = [5, 2, 3, 6, 12, 7, 14, 9, 10, 11]$
$A = [5, 2, 3]$
$A = [2, 3, 5]$ (2 comparison)

$= 3$ total comparisons $\Longleftarrow$

(c):

When it comes to partition, each call to `random-int` (n) chooses pivot (n). After each pivot is chosen, it will no longer continue partitions.
If there n elements in our list, there can be at most n pivots and we have n−1 calls to Partition and n−1 calls to `random-int`.

Hence $\Theta(n)$ calls are made to `random-int` in the best and worst cases. $\Longleftarrow$

2. (20 points) Solve the following recurrence relations using any of the following methods: unrolling, substitution, or recurrence tree (include tree diagram). For each case, show your work.

   (a) $T(n) = T(n-2) + C$ if $n > 1$, and $T(n) = C$ otherwise

   (b) $T(n) = 3T(n-1) + 1$ if $n > 1$, and $T(1) = 3$

   (c) $T(n) = T(n-1) + 2^n$ if $n > 1$, and $T(1) = 2$

   (d) $T(n) = T(\sqrt{n}) + 1$ if $n \geq 2$, and $T(n) = 0$ otherwise

(a): $T(n) = T(n-2) + C$ if $n > 1$, and $T(n) = C$ otherwise

T(n) = T(n-2-2) + C + C
= T(n-2) = T(n-2-2) + C (for n-2)
= T(n-4) + 2C
= T(n-6) + 3C
= T(n-2k) + kC (after k times) $\Longleftarrow$

If n-2k = 1 then, n = 1+2k and k = (n-1)/2

T(n) = T(1) + (n-1)C/2
=1 + $\frac{Cn}{2} - \frac{1}{2} = O(n)$

(b): $T(n) = 3T(n-1) + 1$ if $n > 1$, and $T(1) = 3$

$T(n) = 3T(n-1) + C$
$= 9(3\text{T(n-2)+C})+\text{C}=9\text{T(n-2)}+3\text{C}+\text{C}$
$= 3^k T(n-k) + \sum_{i=0}^{k} C^i \Longleftarrow$

(c): $T(n) = T(n-1) + 2^n$ if $n > 1$, and $T(1) = 2$

$T(n) = T(n-1) + 2^n$
$= T(n-1) + 2^{n-1} + 2^n$
$= T(n-1) + 2^{n-2} + 2^{n-1} + 2^n$
$= 2(2^n - 1) \Longleftarrow$

$T(n) = T(1) + 2^2 + ... + 2^n$

(d): $T(n) = T(\sqrt{n}) + 1$ if $n \geq 2$, and $T(n) =$ otherwise

$T(n) = T(\sqrt{n}) + 1$
$= T(n^{\frac{1}{4}}) + 2$
$= T(n^{\frac{1}{8}}) + 3$
$= T(n^{\frac{1}{2^j}}) + j \Longleftarrow$

3. (20 points) Use the Master Theorem to solve the following recurrence relations. For each recurrence, either give the asympotic solution using the Master Theorem (state which case), or else state the Master Theorem doesn't apply.

   (a) $T(n) = T(\frac{3n}{4}) + 2$

   (b) $T(n) = 3T(\frac{n}{4}) + nlgn$

   (c) $T(n) = 8T(\frac{n}{3}) + 2^n$

   (d) $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + n^2$

   (e) $T(n) = 100T(\frac{n}{42}) + \lg n$

(a): $T(n) = T(\frac{3n}{4}) + 2$

f(n) = 2
a = 1
b = 4/3

$\lim_{n\to\infty} \Theta(n^{log(\frac{4}{3})(1))log^{0+1}(n)})$ $T(n) = \Theta(log(n))$ $\Longleftarrow$

(b): $T(n) = 3T(\frac{n}{4}) + nlgn$

f(n) = nlgn
a = 3
b = 4

$\lim_{n\to\infty} \frac{nlgn}{n^{log_4(3)}} = \infty$ $T(n) = \Theta(nlgn)$ $\Longleftarrow$

(c): $T(n) = 8T(\frac{n}{3}) + 2^n$

f(n) = 2^n
a = 8
b = 3

$\lim_{n\to\infty} \frac{2^n}{n^{log_3(8)}} = \infty$ $T(n) = \Theta(2^n)$ $\Longleftarrow$

(d): $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + n^2$

M.T. doesnt apply to this format.
Recursion tree gives guess T(n) = $\Theta(n^2)$ $\Longleftarrow$

(e): $T(n) = 100T(\frac{n}{42}) + \lg n$

f(n) = lgn
a = 100
b = 42

$\lim_{n\to\infty} \frac{lgn}{n^{log_{42}(100)}} = 0$ T(n) = $\Theta(log_{42}(100))$ $\Longleftarrow$

4. (30 points) Professor Trelawney has acquired n enchanted crystal balls, of dubious origin and dubious reliability. Trelawney needs your help to identify which crystal balls are accurate and which are inaccurate. She has constructed a strange contraption that fits over two crystal balls at a time to perform a test. When the contraption is activated, each crystal ball glows one of two colors depending on whether the other crystal ball is accurate or not. An accurate crystal ball always glows correctly according to whether the other crystal ball is accurate or not, but the glow of an inaccurate crystal ball glows the opposite color of what the other crystal ball is (i.e. If the other crystal ball is accurate, it will glow red. If the other crystal ball is inaccurate it will glow green). You quickly notice that there are two possible test outcomes:

crystal ball i glows	crystal ball j glows		
red	red	$\Longrightarrow$	at least one is inaccurate
green	green	$\Longrightarrow$	both are accurate, or both inaccurate

   (a) Prove that if $n/2$ or more crystal balls are inaccurate, Professor Trelawney cannot necessarily determine which crystal balls are tainted using any strategy based on this kind of pairwise test.

   (b) Consider the problem of finding a single good crystal ball from among the n crystal balls, and suppose Professor Trelawney knows that more than $n/2$ of the crystal balls are accurate, but not which ones. Prove that $\lfloor n/2 \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size.

   (c) Now, under the same assumptions as part (4b), prove that all of the accurate crystal balls can be identified with $\Theta(n)$ pairwise tests. Give and solve the recurrence that describes the number of tests.

(a):

Let g be the number of the green glows and n-g be the number of the red .
Also,assume n-g $\geq$ g. From this assumption we have that we can always find a set g of green and a set B of red of equal size g.
If we assume that the set b=(n-g) of red makes attempts to fool the professor, so for any test made by the professor, its oppisite, so the reds list themselves as green and vice versa g as red. Notice that the chips in g report correct answers and then exhibit a symmetric behaviour: they declare themselves as green and vice vers with b as red. So no matter whichever strategy is used by the professor, the behaviour of the chips in g is indistinguishable from the behaviour of the chips in b. This does not allow the professor to determine which chips are good.

(c):

Using the last sentence in part (b), we can use that to complete part (c) we can use $\frac{n}{2}$ to reduce the size in half, in addition with $\frac{n}{2}$ pairwise tests, we can solve and represent the recurrence:

$$\begin{aligned} \mathrm{T(n)} &= \mathrm{T}(\tfrac{n}{2})+\tfrac{n}{2} \\ &= \mathrm{T}(\tfrac{n}{4})+\tfrac{n}{4}+\tfrac{n}{2} \\ &= \mathrm{T}(\tfrac{n}{8})+\tfrac{n}{8}+\tfrac{n}{4}+\tfrac{n}{2} \\ &= \mathrm{n}\textstyle\sum_{i=1}^{lg(n)} \frac{1}{2^i} < n = O(n) \qquad \Longleftarrow \end{aligned}$$

5. (10 points) Harry needs your help breaking into a dwarven lock box. The lock box projects an array A consisting of n integers $A[1], A[2], \ldots, A[n]$ and has you enter in a two-dimensional $n \times n$ array B – to open the box – in which $B[i, j]$ (for $i < j$) contains the sum of array elements $A[i]$ through $A[j]$, i.e., $B[i, j] = A[i] + A[i + 1] + \cdots + A[j]$. (The value of array element $B[i, j]$ is left unspecified whenever $i \geq j$, so it doesn't matter what the output is for these values.)

   Harry suggests the following simple algorithm to solve this problem:

```
dwarvenLockBox(A) {
   for i = 1 to n {
      for j = i+1 to n {
         s = sum(A[i..j])       // look very closely here
         B[i,j] = s
}}}
```

   (a) For some function g that you should choose, give a bound of the form $\Omega(g(n))$ on the running time of this algorithm on an input of size n (i.e., a bound on the number of operations performed by the algorithm).

   (b) For this same function g, show that the running time of the algorithm on an input of size n is also $O(g(n))$. (This shows an asymptotically tight bound of $\Theta(g(n))$ on the running time.)

(a):

For some function g running time, we consider all the execution cases of the algorithm, we add up the array A[i] through A [j], we know this will take at least $\frac{n}{2}$ operations for each pair and since we know the total number of our pairs is $(\frac{n}{4})^2$. We know our algorithm must perform at least $\frac{n}{2} \cdot (\frac{n}{4})^2 = \frac{n^3}{32}$ operations.

Hence $\Omega(g(n^3))$ $\Longleftarrow$

(b):

In order to prove $g(n) = n^3$ we must consider that each line of code that is adding up A[i] through A[j] At most it will be executed n^2 times because both our inner and outer loops in the the code run n iterations in both loops. We also know adding up the entries O(j-i+1) operations and storing is at most $n^2 \cdot O(n) = O(n^3)$ Hence $O(n^3)$ as desired. $\Longleftarrow$

6. (20 points) Consider the following strategy for choosing a pivot element for the `Partition` subroutine of QuickSort, applied to an array A.

   - Let n be the number of elements of the array A.
   - If $n \leq 24$, perform an Insertion Sort of A and return.
   - Otherwise:
     - Choose $2\lfloor n^{(1/2)} \rfloor$ elements at random from n; let S be the new list with the chosen elements.
     - Sort the list S using Insertion Sort and use the median m of S as a pivot element.
     - Partition using m as a pivot.
     - Carry out QuickSort recursively on the two parts.

   (a) How much time does it take to sort S and find its median? Give a Θ bound.

   (b) If the element m obtained as the median of S is used as the pivot, what can we say about the sizes of the two partitions of the array A?

   (c) Write a recurrence relation for the worst case running time of QuickSort with this pivoting strategy.

(a):

If the number of elements in the array is size n, insertion sort will take $O(n^2)$ time to sort, since we have an array size of $2(n^{\frac{1}{2}})$, it requrires $\Theta(2(n^{\frac{1}{2}})^2$ time to sort. Which means it renames $\Theta(n)$ to sort the array S. Then it will take $2(n^{\frac{1}{2}})$ time to find the median using this array because its a constant so the time complexity:

$\Theta(n) + \Theta(2(n^{\frac{1}{2}}) = \Theta(n) \quad \Longleftarrow$

(b):

The size of the partitioned array depends on the position of the pivot in the Sort list of numbers. If the pivot element is close to the middle of the list, the partitioned is calculated by the medium of two into that one have random elements in the list. This will give an approximate middle value of the original list and the two partitions will be equal in size.

(c):

not sure.