Name: Yong Zhou
HW: One

1, Stable Matching
(a) Answer: There is a switch that would improve the partner of a woman by exchanging her preferences. See the example below (M as Man, W as Women):

- Example: The original table (1) below:

	1st	2nd	3rd
M	W	W2	W1
M1	W2	W	W1
M2	W	W2	W1

Table (2)

	1st	2nd	3rd
W	M1	M	M2
W1	M	M1	M2
W2	M	M1	M2

The stable matching will be the pairs of M and W; M1 and W1; M2 and W2.

Table (3)

	1st	2nd	3rd
M	W	W2	W1
M1	W	W2	W1
M2	W	W2	W1

.
Correctness:
- From the original table (1) and (2). M, M1, M2 find preference first. The final stable matching is (M, W), (M1, W2), (M2, W1). As the question asked, we randomly exchange one of the M or M1 or M2 s' preference list. Let us say in M1 (as table 3) preference W2 and W, after the M1 lies and say it's prefers W over W2(actually not), then M1 get the result of (M, W2), (M1, W), (M2, W1)

(b)
Algorithm:
// M propose first,
- Initially all M and W are free.
  - Let M propose first.

By G-S algorithm that return a set of pairs S. The set S is a stable matching from textbook by 1.6(page 8).

//Let W propose first.
By G-S algorithm that return a set of pairs T. The set T is a stable matching.
If S equals T:
    Return unique stable matching

- **Proof: not uniqueness if S not equals T**, by textbook (1.9) page21: there is no instability with respect
  to the return matching S, from G-S algorithm. This stable matching is not unique.

- **Proof: uniqueness if S equals T.**
    - By textbook (1.7) page 11. For the men propose first, the G-S algorithm is ideal. For a woman W, we say that M is a valid partner if there is a stable matching that contains the pair (M, W). We say that M is the worst valid partner of W if M is a valid partner of W, and no man whom W ranks lower than M is a valid partner of hers. Vice versa for Man if Women propose first. Then, to the both set S and set T. if the best valid set S equals the Worst valid set T. Then, the stable matching to be the unique stable matching.
    - Assume it will end up with other possible stable matching beside on S and T: then textbook (1.8: In the stable matching set S, each woman is paired with her worst valid partner will not valid). Unless the proof of 1.8, not correct, the assume that other possible stable matching beside S and T.

Complexity: This algorithm run two table in different while loop is 2O (n^2), and compare the two sets is 1. The total running time: 2O (n^2) + 1 = O (n^2)

2,
(a) II and V are true.

(b) The number of calling in Merge-Sort is 2N – 1 = O(n).

Prove by strong induction:
- **Prove: the array is recursively divided in two halves, which is n**

Initialize p is length of the array, r = 1. we proved r – p = k +1 could be valid, call 2n - 1
- **Base case**
    - Size of array A = 2, Find the middle of array divide into two halves, q = (0 + 2)/2; two halves of the array length = 1. Merge-Sort run itself and two halves. Stop to call Merge-Sort (). Run merge-Sort () 2n - 1 = 3
    - Size of array A = 3, Find the middle of array divide into two halves, q = (0 + 3)/2; one half of the array length = 1. Another half of the array length = 2. Repeat the first stop above when size of array A = 2. Total run merge-Sort () 2 + 3 times. Stop to call Merge-Sort (); 2n – 1 = 5;

- **Induction hypothesis**
    - We assume that when length of array is n, which is r - p

- **Induction step**
  - Array length is arbitrary integer n + 1 is an even number, total call is two halves = initial call + 2 * F($\frac{n+1}{2}$); 1 <= $\frac{n+1}{2}$ <= k; Then going further by assumption, 1 + 2(2($\frac{n+1}{2}$) - 1) = 1 + 2n
  - Array length is arbitrary integer n + 1 is an odd number, total call is two halves = initial call + F($\frac{(n+1)+1}{2}$) + F((n+1) - $\frac{(n+1)+1}{2}$) = 1 + F ($\frac{n+1}{2}$) + F ($\frac{n}{2}$); going further by assumption, 1 + P($\frac{n+2}{2}$) + P($\frac{n}{2}$) = 1 + (2($\frac{n+2}{2}$) - 1) + (2 ($\frac{n}{2}$) -1) = 2n +1 ;

- **Apply the induction principle to prove the claim you wanted.**
  - Therefore, the claim that n + 1 is valid. The number of the total call is 2n – 1, p is the length of the array, r is 1, which is 2(r-p) -1
- **Complexity**: O(nlogn)

3, The list of functions in decreasing asymptotic order is:

- $2^{2^{n+1}} > 2^{2^n} > n2^n > 2^{lg^{1.001}*n} > n^{\lg\lg n} > n^{100} = 2^{100\lg n} > n^3 > n^2 > 4^{\lg n} > \binom{n}{2} + nlgn >$
  n lg n > lg(n!) > n > $\sqrt{n} > \left(\sqrt{2}\right)^{\lg n} > lg^2\, n >$ lg n > $\log_3 5n > n^{1/\lg n}$

Correctness: We can see the simplified version of the given numbers.

- $2^{2^{n+1}} = 4^{2^n} > 2^{2^n} > n2^n = 2^n * 2^{\lg n} > 2^{lg^{1.001}*n} = 2^{(logn)\lg(.001)n} = n^{lg^{0.001}*n} > n^{\lg\lg n} = 2^{(lgn)lglgn} > n^{100} = 2^{lgn^{100}} = 2^{100\lg n} > n^3 > n^2 = 4^{\lg n} = 2^{2*\lg n} = 4^{\lg n} = 2^{2\lg n} = n^2 > \binom{n}{2} + nlgn = \frac{n(n-1)}{2}$ + n lgn = $\Theta(n^2)$ > lg(n!) > n > $\sqrt{n} > \left(\sqrt{2}\right)^{\lg n} = n^{\frac{1}{2}} > lg^2\, n >$ lg n>$\log_3 5n = log_3^5 + \log_3 n > n^{1/\lg n} = 2$

- Log(n!) = log(1*2*3*…*n) = sum (I to n) log(i) <= n* log2(n) (largest term)

Observation and calculation:
- a^n as n going to infinity, the whole number will go to infinity.
- Lg(n!) = lg(1*2*…*n) so $\frac{n}{2}$lg(n/2) <= lg(n!) <= nlgn in
- log_a^b, a is integer, when b gets bigger, number itself will be bigger; as a getting larger, number will become smaller.