Statement of Integrity: I, Derek Zhu, state that I completed this assignment with integrity and by myself.

Problem 1:

There are $0 < m <= 100$ number of lists, the run time for each line are:
Line #1: 0
Line #2: 1
Line #3: m
Line #4: m
Line #5: $m * O(n^2)$
Line #6: $O(n)$
Line #7: 1

Since the m is a constant value less or equal to 100 as given, then the overall run time is:
$T(n) = 2 + 2m + O(n) + m * O(n^2) = const1 + const2 * O(n^2) = O(n^2)$

Problem 2:

Use the contradiction proof method: Assume PermuteWithAll produce a uniform random permutation, then the possibility for each permutation will be the same value which is $1/n!$, since there are only $n!$ possible combinations from the N-elements array. By swapping an random element from anywhere in the array, PermuteWithAll produces n^n total combinations from the N-elements array, that means it produce $(n^n - n!)$ more than the possible combinations. If the $(n^n - n!)$ are uniform randomly distributes among the $n!$ possible combinations, then $(n^n - n!)$ must be dividable by $n!$, which is not necessarily true. For example, $4^4 = 256,\ 4! = 24$, $256 - 24 = 232$ is not dividable by 24. Hence, there are $(n^n - n!)$ more combinationa cannot be uniform randomly distributes among the $n!$ possible combinations, the possibility for each permutation will NOT be the same value. The orignal assumption leads to a contradiction.

Hence the answer is NO.

Problem 3:

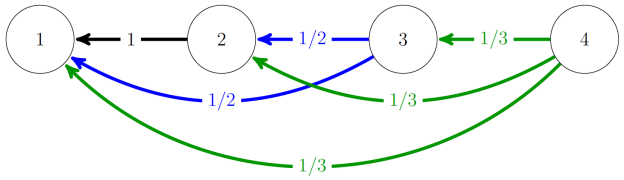

Figure 1: A mobile ad-hoc network. As new nodes appear, they choose any single previous node to connect to. When 2 joins, it has only 1 to connect to. When 3 joins, it can pick from 1 or 2 with equal probability, and so on.

The network begins with a single node v_1; therefore, the expected number of incoming link to v_1 is 0:
$E(v1) = 0$

When node v_2 joins, it has a possibility $1/1 = 100\%$ of choosing v_1 to create a link to it;therefore, the expected number of incoming link to v_1 and v_2 so far are:
$E(v1) = 1/1 = 1$
$E(v2) = 0$

When node v_3 joins, it select uniformly at random from existing nodes v_1,v_2; it has a possibility $1/2 = 0.5$ of choosing v_1, and also it has a possibility $1/2 = 0.5$ of choosing v_2 to create link to it. Therefore, the expected number of incoming link to v_1 ,v_2 , v_3 so far are:
$E(v1) = 1/1 + 1/2$
$E(v2) = 1/2$
$E(v3) = 0$

when node v_n joins, it select uniformly at random from existing nodes v_1,v_2, ..., $v_{(n-1)}$; it has a possibility $1/(n-1)$ of choosing any node from existing nodes v_1,v_2, ..., $v_{(n-1)}$ to create to it. Therefore, the expected number of incoming link to v_1 ,v_2 , v_3 …, v_j ,... v_n so far are:
$E(v1) = 1/1 + 1/2 + 1/3 + 1/4 + ..+ 1/(n-1)$
$E(v2) = 1/2 + 1/3 + 1/4 + ..+ 1/(n-1)$
$E(v3) = 1/3 + 1/4 + ..+ 1/(n-1)$
…
$E(vj) = 1/j + 1/(j+1) + 1/(j+2) +...+ 1/(n-1)$
…
$E(v(n-1)) = 1/(n-1)$
$E(vn) = 0$

If define harmonic series as $H(x) = 1/1 + 1/2 + 1/3 +... + 1/x$
Then the exact formula in terms n and j is:
$E(vj) = H(n-1) - H(j-1)$, *where harmonic* $H(x) = 1/1 + 1/2 + 1/3 +...+ 1/x$

Let's prove $E(vj) = \Theta(ln(n))$:

By looking the "cheating sheet" suggested or CLRS text book page 1142 formula A.10, we have:
$ln(1 + x) \leq H(x) \leq 1 + lnx$

then:
$ln(n) \leq H(n-1) \leq 1 + ln(n-1)$
$ln(j) \leq H(j-1) \leq 1 + ln(j-1)$

Or:
$-1 - ln(j-1) \leq -H(j-1) \leq -ln(j)$

Hence:
$ln(n) - 1 - ln(j-1) \leq H(n-1) - H(j-1) \leq 1 + ln(n-1) - ln(j)$
$ln(n) - 1 - ln(j-1) \leq E(vj) \leq 1 + ln(n-1) - ln(j)$
$ln(n) + c1 \leq E(vj) \leq ln(n-1) + c2$
Where: $where\ c1 = -1 - ln(j-1);\ c2 = 1 - ln(j)$
Since $ln(n) + c1 \leq E(vj)$, hence: $E(vj) = \Omega(ln(n))$
Since $E(vj) \leq ln(n-1) + c2$, hence: $E(vj) = O(ln(n-1)) = O(ln(n))$

Therefore: $E(vj) = \Theta(ln(n))$

Problem 4:

To keep track of the cards, we assign a tuple to each card with (index, value), for example, (2,v) menas the card is at 2nd position in the single stack with value v. we initialze the 21 cards single stack as:

stack={ (1,v1), (2,v2), (3,v3),
(4,v4), (5,v5), (6,v6),
(7,v7), (8,v8), (9,v9),
(10,v10), (11,v11), (12,v12),
(13,v13), (14,v14), (15,v15),
(16,v16), (17,v17), (18,v18),
(19,v19), (20,v20), (21,v21) }

(a) The **1st card** in the chosen pile is the 8th card of the single stack, which is card (8,v8). When deal out the three piles in the next phase, the card (8,v8) will be in the 3rd position of the pile, since there are floor((8-1)/3)=2 cards ahead in the same pile. Hence **it advances two position** from the first position to the 3rd position in the pile.Here is the piles status:

$p1 = (1, v1), (4, v4), (7, v7), (10, v10), (13, v13), (16, v16), (19, v19)$
$p2 = (2, v2), (5, v5), (8, v8), (11, v11), (14, v14), (17, v17), (20, v20)$
$p3 = (3, v3), (6, v6), (9, v9), (12, v12), (15, v15), (18, v18), (21, v21)$

(b) The **2nd card** in the chosen pile is the 9th card of the single stack, which is card (9,v9). When deal out the three piles in the next phase, the card will be in the 3rd position of the pile, since there are floor((9-1)/3)=2 cards ahead in the same pile. Hence **it advances one position** from the sencod position to the 3rd position in the pile. Here is the piles status:

$p1 = (1, v1), (4, v4), (7, v7), (10, v10), (13, v13), (16, v16), (19, v19)$
$p2 = (2, v2), (5, v5), (8, v8), (11, v11), (14, v14), (17, v17), (20, v20)$
$p3 = (3, v3), (6, v6), (9, v9), (12, v12), (15, v15), (18, v18), (21, v21)$

The **3nd card** in the chosen pile is the 10th card of the single stack, which is card (10,v10). When deal out the three piles in the next phase, the card will be in the 4th position of the pile, since there are floor((10-1)/3)=3 cards ahead in the same pile. Hence **it advances one position** from the 3rd position to the 4th position in the pile. Here is the piles status:

$p1 = (1, v1), (4, v4), (7, v7), (10, v10), (13, v13), (16, v16), (19, v19)$
$p2 = (2, v2), (5, v5), (8, v8), (11, v11), (14, v14), (17, v17), (20, v20)$
$p3 = (3, v3), (6, v6), (9, v9), (12, v12), (15, v15), (18, v18), (21, v21)$

(c) The chosen 11st card of the single stack is actually the **4th card** of the chosen pile : as we see from the step above, the 4th card of a chosen pile will stay in 4th position after deal out the three piles in the next phase, since there will be 10/3=3 cards ahead it in the pile. Since the 4th card of a chosen pile will be again the 11th card of the new stack after deal out, therefore, no matter how

many phase deal out, it will still be the 11th card in the single stack (the 4th position of the chosen pile). Hence **it advances zero position** in the pile

(d) If the chosen card is the **5th card** in the chosen pile, it will be in 7+5= 12th position of the single stack after sandwich the chosen pile in the middle. When deal out the three piles in the next phase, the card (12,v12) will be in the 4th position of the pile, since there are floor((12-1)/3)=3 cards ahead in the same pile. Hence it go backward one position from the 5th position to the 4th position in the pile. Here is the piles status:

$p1 = (1, v1), (4, v4), (7, v7), (10, v10), (13, v13), (16, v16), (19, v19)$
$p2 = (2, v2), (5, v5), (8, v8), (11, v11), (14, v14), (17, v17), (20, v20)$
$p3 = (3, v3), (6, v6), (9, v9), (12, v12), (15, v15), (18, v18), (21, v21)$

It will stay in the 4th position (the middle position) in it's pile for all next phase, according to step C above.

If the chosen card is the **6th card** in the chosen pile, it will be in 7+6= 13th position of the single stack after sandwich the chosen pile in the middle. When deal out the three piles in the next phase, the card (13,v13) will be in the 5th position of the pile, since there are floor((13-1)/3)=4 cards ahead in the same pile. Hence it go backward one position from the 6th position to the 5th position in the pile. Here is the piles status:

$p1 = (1, v1), (4, v4), (7, v7), (10, v10), (13, v13), (16, v16), (19, v19)$
$p2 = (2, v2), (5, v5), (8, v8), (11, v11), (14, v14), (17, v17), (20, v20)$
$p3 = (3, v3), (6, v6), (9, v9), (12, v12), (15, v15), (18, v18), (21, v21)$

It will go backward one position to from the 5th to the 4th position in it's pile for the next phase, according to step above regarding the 5th card, then stay at the 4th (the middle) position in it's pile forever for any further phases according to the step C above.

If the chosen card is the **7th card** in the chosen pile, it will be in 7+7= 14th position of the single stack after sandwich the chosen pile in the middle. When deal out the three piles in the next phase, the card (14,v14) will be in the 5th position of the pile, since there are floor((14-1)/3)=4 cards ahead in the same pile. Hence it go backward two position from the 7th position to the 5th position in the pile. Here is the piles status:

$p1 = (1, v1), (4, v4), (7, v7), (10, v10), (13, v13), (16, v16), (19, v19)$
$p2 = (2, v2), (5, v5), (8, v8), (11, v11), (14, v14), (17, v17), (20, v20)$
$p3 = (3, v3), (6, v6), (9, v9), (12, v12), (15, v15), (18, v18), (21, v21)$

It will go backward one position to from the 5th to the 4th position in it's pile for the next phase, according to step above regarding the 5th card, then stay at the 4th (the middle) position forever for any further phases according to the step C above.

(e) A card is chosen from the single stack, then deal out into three files in the first round of deals,then sandwich the chosen pile in the middle.

Let p(i) be the possibility that the ith card is the chosen card in the chosen pile, then:
$p(i) = 1/7,\ when\ i \in [1, 2, 3, 4, 5, 67]$

Let x(i) be a random indicator available, we define it as:
x(i) = 1 if the ith card in a chosen pile move to the 4th position after deal out in the next phases, otherwise 0;

According to the steps a~d above,
For the 1st card , it advances two position from the 1st position to the 3rd position,
henece: x(1) = 0
For the 2nd card, it advances one position from the 2nd position to the 3rd position,
henece: x(2) =0
For the 3rd card, it advances one position from the 3rd position to the 4th position,
henece: x(3) =1
For the 4th card, it advances zero position, henece: x(4) =1
For the 5th card, it go backward one position from the 5th position to the 4th position,
henece: x(5) =1
For the 6th card, it go backward one position from the 6th position to the 5th position,
henece: x(6) =0
For the 7th card, it go backward two position from the 7th position to the 5th position,
henece: x(7) =0

The expected possibility of a chosen card will be in the 4th position after two round deal is:

$$\sum_{i=1}^{i=7} p(i)x(i) = 0 + 0 + (1/7) * 1 + (1/7) * 1 + (1/7) * 1 + 0 + 0 = 3/7$$

That means about $3/7 \approx 43\%$ or about half of time, you only need to go through two round of deals that the chosen card will be at the 4th position of the chosen pile, which is the 7+4=11th position at the single stack.

Problem 5:

Idea:

given k sorted lists with n total number of elements, we will use heaps for k-way merging, here are the steps:

Initialize the heap “min-heap” by pop (remove) the first element of each sorted list and insert it to min-heap;

Initialize an empty array “result” to store result;

Repeat until “min-heap” is empty:

- Heapify “min-heap”;
- Extract the minimum value of “min-heap” and add it to “result”;
- Pop (remove) the first element (if any) of the list which that minimum value was taken from into “min-heap”

Return “result”

Pseudocode:

```
MERGE_K_SORTED_LIST_ARRAY(A, k)
        //input: k is an positive integer
        //input: A is an array of k elements, each element is a sorted list.
1       minHeap = empty min heap
2       result= empty array
        // Initialize the “min-heap” with the 1st element of each sorted list:
3       For i=1 to k
4               If  A[i] is not empty
5                       x.key=A[i].pop()
6                       x.from = i
7                       minHeap.insert_and_heapify(x)
        // process the min-heap until it is empty:
8       While minHeap is not empty
9               x= minHeap.pop_and_heapify()
10              result.append(x.key)
                //get next element from the same list:
11              If  A[x.from] is not empty
12                      x.key=A[x.from].pop()
13                      minHeap.insert_and_heapify(x)
14      return result
```

Correctness:

(1) Initialize the heap “min-heap” by pop (remove) the first element of each sorted list and insert it to min-heap: this ensures that the min-heap contains current smallest value among all the sorted lists;

(2) Heapify “min-heap”: this ensures that the min-heap root contains the current smallest value among all nodes of the min-heap;

(3) Extract the minimum value of "min-heap" and append it to "result": this ensures that the current smallest value is appended to the "result" list;
(4) Pop (remove) the first element (if any) of the list which that minimum value was taken from into "min-heap": again, this ensures that the min-heap contains current smallest value among all the sorted lists;
(5) Repeat until "min-heap" is empty: this ensures that all the values among all sorted lists have been appended to the "result" list in a sorted order.

Runtime:

Line #1: 1
Line #2: 1
Line #3: k
Line #4: k
Line #5: $\Sigma length(i) = n,\ where\ n\ is\ the\ total\ elements\ of\ the\ all\ sorted\ lists$
Line #6: n
Line #7: $nlgk$
Line #8: n
Line #9: $nlgk$
Line #10: n
Line #11: n
Line #12: n
Line #13: $nlgk$
Line #14: 1

Total run time: $T(k, n) = 3nlgk + 6n + 2k + 3 = O(nlgk)$