1. Charn Supparnaya

1. For 1 <= i <= n, let

   X_i = I{customer i gets hat back}

   Let random variable X be the number of customers who gets hat back. We want to compute E[X].

   $$X = \sum_{i=1}^{n} X_i$$

   The probability that customer i gets own hat back is 1/n, which implies $E[X_i] = 1/n$ by basic properties of indicator random variables.

   $$E[X] = E[x_1 + x_2 + \cdots + x_n] = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} \frac{1}{n} = 1$$

`

2. Sophanha Phan

Indicator Random Variable

Exercise 5.25 (P122)

Let X_{ij} be the indicator variable that represents whether the i^{th} and j^{th} element is an inversion.

$$X_{ij} = I \begin{cases} 1, & i^{th} \text{ and } j^{th} \text{ element is an inversion} \\ 0, & i^{th} \text{ and } j^{th} \text{ element is not inversion} \end{cases}$$

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} \qquad \Pr\{i < j\} = \frac{1}{2}$$

$$\begin{aligned}
E[x] &= E\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} x_{ij}\right] \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E[x_{ij}] \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \Pr \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{1}{2} \\
&= \frac{1}{2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} 1 = \frac{1}{2}\sum_{i=1}^{n-1}(n-(i+1-1)) \\
&= \frac{1}{2}\sum_{i=1}^{n-1}(n-i) = \frac{1}{2}\left(\sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i\right) \\
&= \frac{1}{2}\left(n(n-1) - \frac{(n-1)(n-1+1)}{2}\right) = \frac{1}{2}\left(n(n-1) - \frac{n(n-1)}{2}\right) \\
&= \frac{1}{2}\left(\frac{2n(n-1)}{2} - \frac{n(n-1)}{2}\right) = \frac{1}{2}\left(\frac{n(n-1)}{2}\right) = \frac{n(n-1)}{4}
\end{aligned}$$

3. Charn Supparnnaya

3\. a) If elements are equal, when PARTITION returns, q is equal to r and all elements in A[p..q-1] are equal. The recurrence we get is:

$T(n) = T(n - 1) + T(0) + \Theta(n)$

So $T(n) = \Theta(n^2)$

b) Modified PARTITION code:

```
PARTITION' (A, p, r)
    x = A[p]
    i = h = p
    for j = p + 1 to r
        if A[j] < x
            y = A[j]
            A[j] = A[h + 1]
            A[h + 1] = A[i]
            A[i] = y
            i = i + 1
            h = h + 1
        else if A[j] == x
            exchange A[h + 1] with A[j]
            h = h + 1
    return (i, h)
```

c) RANDOMIZED-PARTITION' is same as original RANDOMIZED-PARTITION except for call to PARTITION being changed to PARTITION'.
Modified QUICKSORT code:

```
QUICKSORT' (A, p, r)
    if q < r
        (q, t) = RANDOMIZED-PARTITION' (A, q, r)
        QUICKSORT' (A, p, q - 1)
        QUICKSORT' (A, t + 1, r)
```

d) Put elements equal to the pivot in the same partition as the pivot. This makes problem sizes of QUICKSORT' no larger than those of the original QUICKSORT when all elements are distinct, and even with equal number of elements.

4.
Refer to textbook 9.2