CSC3100 Assignment 1 Report

Xue Zhongkai 122090636

Problem 1

This is a rather simple question, just to sort the coordinates with different rules. To make my codes simpler, I applied `lambda` function in Python, and also refer to this quora page to handle I/O more efficiently.

Problem 2

This is a problem dealing with chained sums within a grid. At first I tried to simplify the summation below intuitively,

$$\sum_{a=1}^{n}\sum_{b=1}^{m}\sum_{c=a}^{n}\sum_{d=b}^{m}\sum_{i=a}^{c}\sum_{j=b}^{d}k_{i,j}$$

To optimize both time and space, I came across prefixSum algorithm and attempted to reduce it to time complexity $O(N^4)$ and space complexity $O(N^2)$:

```
total_sum = 0

for i in range(1, n+1):
    for j in range(1, m+1):
        extended_grid[i][j] = (extended_grid[i-1][j] + extended_grid[i][j-1] \
                               - extended_grid[i-1][j-1] + grid[i-1][j-1]) % 998244353
for a in range(1, n+1):
    for b in range(1, m+1):
        for c in range(a, n+1):
            temp1 = extended_grid[a-1][b-1] % 998244353
            for d in range(b, m+1):
                temp2 = extended_grid[c][b-1] % 998244353
                rect_sum = (extended_grid[c][d] - extended_grid[a-1][d] \
                            - temp2 + temp1) % 998244353
                total_sum = (total_sum + rect_sum) % 998244353
```

However, it still could not pass large-scale test cases for the sake of out-of memory, but this approach has already reached its optimal.

Then I saw the hint, and decided to apply it from the calculation principle. It has been shockingly simple and elegant:

```
long total = 0;
for (long i = 0; i < p; i++) {
  long x = scanner.nextLong();
  long y = scanner.nextLong();
  long k = scanner.nextLong();

  long rectangles = (x * y * (n - x + 1) * (m - y + 1)) % MOD;
  total = (total + (rectangles * k) % MOD) % MOD;
}
```

First we discuss `long rectangles = ((long) x * y * (n - x + 1) * (m - y + 1)) % MOD;`, which calculates the number of all the rectagles containing specific Tyranid squads:

- `x * y`: This calculates all the possible choices of left-up rectangle vertex, from $(1, 1)$ to (x, y).
- `(n - x + 1) * (m - y + 1)`: This calculates that of right-down rectangle vertex, from (x, y) to (n, m) .
- Multiply these two parts together, we get the number of all possible rectangles contaning the squad.

Then we use `total = (total + (rectangles * k) % MOD) % MOD;` to sum up the contribution of squads to the expected number of the Tyranids. For every iteration, `rectangles * k` is added to the total for the contribution of this specific squads.

Astonishingly, it finally reaches time complexity $O(N)$ and space complexity $O(1)$! It passed all test cases and marked the end of this assignment 1.