PDHW - Week 6

Pset 1

1. Shown as below:

```
int combiRec(int n, int m, int** com)
{
        if (n < m) //若 m > n 則代表出錯，按照上題題意要輸出 -1
                return -1;
        else if (n == m) //我們知道 C n 取 m 會等於 1，所以 return 1
                return 1;
        else if (m == 1) //我們知道 C n 取 1 會等於 n，所以 return n
                return n;
        else
        {
                if (com[n - 1][m - 1] == -1) //檢查我們的動態陣列中是否有存以前算過的紀錄，若沒有則遞迴計算並記錄進動態陣列
                {
                        int res = combiRec(n - 1, m, com) + combiRec(n - 1, m - 1, com);
                        com[n - 1][m - 1] = res;
                        return res;
                }
                else //若有紀錄，則直接輸出動態陣列中的紀錄，不重複計算
                        return com[n - 1][m - 1];
        }
}
```

2. Shown as below:

```
// declare an n by m dynamic array
// and initialize all elements in it to -1
int** com = new int*[n];
for (int i = 0; i < n; ++i)
{
```

```
        com[i] = new int[m];
        for (int j = 0; j < m && j <= i; ++j)
        {
                com[i][j] = -1;
        }
}
```

3. Shown a below:

```
int combiRec(int n, int m, int** com) {
        for (int i = 0; i < n; ++i)
        {
                for (int j = 0; j < m && j <= i; ++j)
                {
                        if (i == j)
                                com[i][j] = 1;
                        else if (j == 0)
                                com[i][j] = i + 1;
                        else
                                com[i][j] = com[i - 1][j] + com[i - 1][j - 1];
                }
        }
        return com[n - 1][m - 1];
}
```

4. 首先我們可以看到 (c) 小題的實做會把 C_m^n 的 (就算不是必須的) 每一個可能會需要用到的元素都計算一次，所以這樣過多而不必要的計算會在時間複雜度上帶來負面影響，但相對的因為他是採用 bottom-up 的動態程式規劃，所以不會像 (a) 小題那樣需要重複呼叫自己去計算，而是直接查表，就某方面來說對時間複雜度會有正面影響。

Pset 2

---

2. Shown a below:

```
for (int i = 0; i < r; i++)
        delete [] array[i];
delete [] array;
array = nullptr;
```

```
5 return 0;
```