演算法 Homework 20191003
山下夏輝（Yamashita Natsuki）
R08922160

Mini HW #4

繳交作業

截止時間 星期四， 14:20 **總分** 1 **繳交** 線上輸入或者檔案上傳
檔案類型 pdf **接受繳交時間** 10月17日 14:20 - 10月24日 14:20 7 天

Consider a 0/1 Knapsack Problem where you have ***N*** objects to choose from.

The weight and value of each object are listed in the table below.

Weight	1	3	4	5	8	10	11
Value	3	7	10	12	17	19	21

1. Construct a DP table to fill knapsack with capacity ***W = 15*** (50%)
   (Your DP algorithm must run in ***O(N*W)*** time)
2. Modify the DP algorithm to 0/1 Knapsack problem mentioned in class so that we can find maximum total value with ***O(W)*** space. (50%)

1.

```
KS(n, W)
        If DP[n][W] != undefined
                Return DP[n][W]
        If n == 0 || W == 0
                DP[n][W] = 0
        Else if w[n] > W
                DP[n][W] = KS(n-1, W)
        Else:
                DP[n][W] = Max(KS(n-1, W), KS(n-1, W-w[n])+v[n])
        Return DP[n][W]
```

2.

//reference: https://www.geeksforgeeks.org/space-optimized-dp-solution-0-1-knapsack-problem/

```
KS_SpaceOpt(n, W)
        If DP[n][W] != undefined
                Return DP[n][W]
        If n == 0 || W == 0
                DP[n][W] = 0
        If n%2 = 0
                Else if w[n] > W
                        DP[0][W] = KS(1, W)
                Else
                        DP[0][W] = Max(KS(1, W), KS(1, W-w[n])+v[n])
        Else
                Else if w[n] > W
                        DP[1][W] = KS(0, W)
                Else
                        DP[1][W] = Max(KS(0, W), KS(0, W-w[n])+v[n])
        Return (n%2 = 0) ? DP[1][W] : DP[0][W]
```