Introduction to Algorithm
Exercise #3

I. Environment

i. How to run your program

OS: Windows

Compiler Version: g++

IDE: visual studio 2019

II. Results

i. Method or Solutions

Header file:

Using the header file.

```
#include<iostream>
#include<stdio.h>

using namespace std;
```

Global Variable Declarations:

- Profit: To store profit tables.
- Sum: Calculating temporary total profits for different course combinations with number of days.
- score: Stores the maximum profit for a given number of days.

```
int profit[1000][1000];
int sum[1000][1000];
int score[10000];
```

‘dp’ Function:

- initializes the score array. Then, it calculates the maximum profit when only the first stage courses are considered.
- Computing the maximum profit for each new stage of courses, considering the previous stages.
- Traversing the sum array and updating the score array to reflect the maximum profit.
- The function returns the maximum profit for the given number of days.

```
int dp(int days, int a, int b){
    for(int i=0;i<10000;i++){
        score[i]=0;
    }
    //has b rows(ways)
    for(int i=1;i<=a;i++){
        score[i]=profit[i][1];
    }

    //calculate course 2 to a
    for(int i=2;i<=b;i++){

        //calculate sum between before courses and current course
        for(int j=1;j<=days;j++){
            for(int k=1;k<=a;k++){
                if(score[j]!=0){
                    if(j+k<=days){
                        sum[j][k]=score[j]+profit[k][i];
                    }
                }
            }
        }
```

```
31
32          for(int j=1;j<=days;j++){
33              for(int k=1;k<=days;k++){
34                  if(sum[j][k]!=0){
35                      if(score[j+k]<sum[j][k]){
36                          score[j+k]=sum[j][k];
37                      }
38                  }
39              }
40          }
41          score[i-1]=0;
42
43          for(int j=0;j<1000;j++){
44              for(int k=0;k<1000;k++){
45                  sum[j][k]=0;
46              }
47          }
48
49      }
50      return score[days];
51  }
```

‘main’ Function:

- Reads the number of sets of profit tables.
- For each profit table, it reads its dimensions (a and b), and then populates the profit array.
- Reads the number of query.
- For each query, it reads the number of days to be queried and calls the calculate function to find out the maximum profit, and outputs the result.

```cpp
int main() {
    int sets, a, b, queries;
    cin >> sets;  // Number of profit tables

    while (sets--) {
        cin >> a >> b;  // Dimensions of the profit table

        // Read profit table
        for (int i = 1; i <= a; ++i) {
            for (int j = 1; j <= b; ++j) {
                cin >> profit[i][j];
            }
        }

        cin >> queries;  // Number of days of study queries
        int days;
        for (int i = 0; i < queries; ++i) {
            cin >> days;
            int ans = calculate(days, a, b);
            cout << ans << endl;
        }
    }

    return 0;
}
```

ii. Anything you want to share

In this exercise, I practiced the dynamic programming to solve the problem. In this way, I combine this knowledge of the algorithm that professor teaches in class with the C++ programming. As a result, this is a treasured opportunity to learn the algorithm by hand.