Dynamic Programming

1.1 Objectives

To learn how to select the appropriate problem-solving method for computer science problems based on several categories of problems..

1.2 Requirements

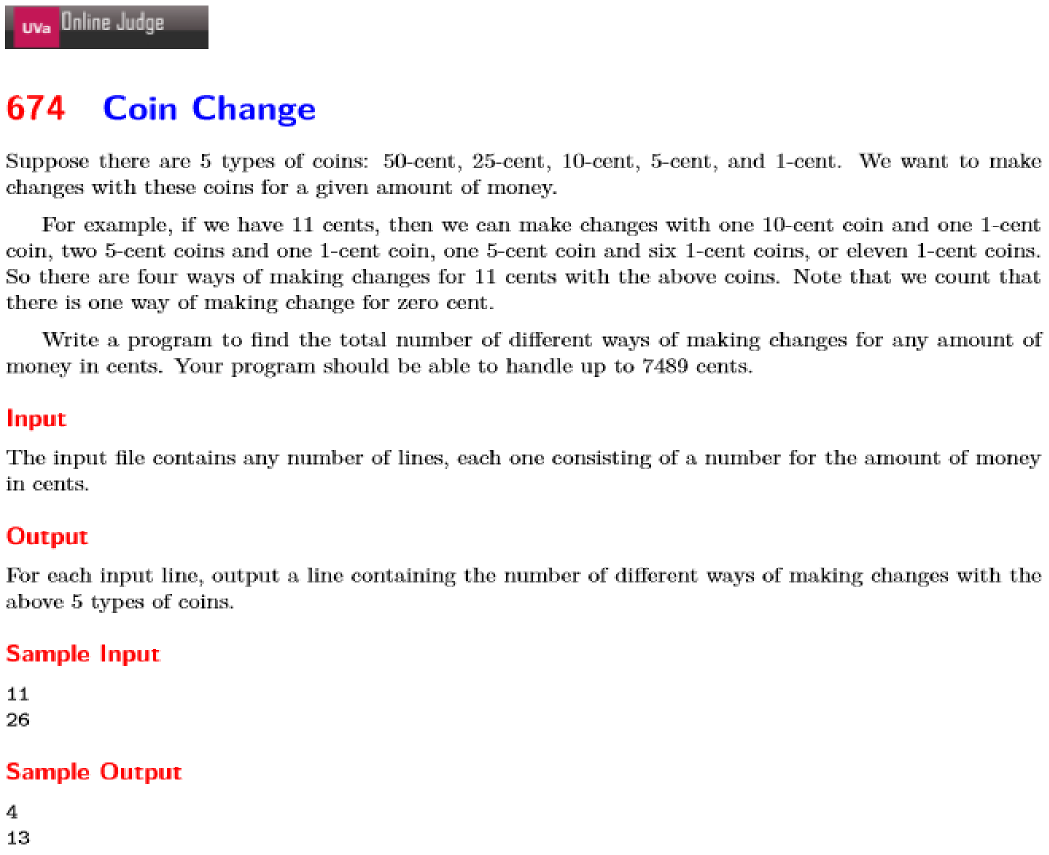
The tasks should be performed at the Lab (Makmal Pengajaran 1, Block B) machines in the faculty
Important:
Select **one leader** for each team of two students. Each team can use two computers but the computers must be side by side.

1.3 Tasks

Prior to the Lab

i. Start reading chapter 3 in the text book.

a) UVa online judge: Coin Change.



UVa Online Judge

674 Coin Change

Suppose there are 5 types of coins: 50-cent, 25-cent, 10-cent, 5-cent, and 1-cent. We want to make changes with these coins for a given amount of money.

For example, if we have 11 cents, then we can make changes with one 10-cent coin and one 1-cent coin, two 5-cent coins and one 1-cent coin, one 5-cent coin and six 1-cent coins, or eleven 1-cent coins. So there are four ways of making changes for 11 cents with the above coins. Note that we count that there is one way of making change for zero cent.

Write a program to find the total number of different ways of making changes for any amount of money in cents. Your program should be able to handle up to 7489 cents.

Input

The input file contains any number of lines, each one consisting of a number for the amount of money in cents.

Output

For each input line, output a line containing the number of different ways of making changes with the above 5 types of coins.

Sample Input

```
11
26
```

Sample Output

```
4
13
```

i. Submit Program A (Coin Change without DP) to PC Square and UVa online judge.

```
/* Coin Change */

#include <cstdio>
#include <cstring>
using namespace std;

int N = 5, V, coinValue[5] = {1, 5, 10, 25, 50}, memo[6][7500];
// N and coinValue are fixed for this problem, max V is 7489

int ways(int type, int value) {
int ans;
  if (value == 0)              return 1;
  if (value < 0 || type == N)  return 0;

  return ans = ways(type + 1, value) + ways(type, value - coinValue[type]);
}

int main() {
 // we only need to initialize this once
  while (scanf("%d", &V) != EOF)
    printf("%d\n", ways(0, V));

  return 0;
}
```

ii. Apply Dynamic Programming to Program A to speed the execution time and name it as Program B. Submit Program B (Coin Change with DP) to PC Square and UVa online judge.

b) Problem_UVA_11450
Add Dynamic Programming element to the solution for Problem: UVa 11450 Wedding Shopping. Submit Program C (Wedding Shopping with DP) via the PC Square and UVA online Judge.

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <sstream>
#include<cstdlib>

#include<iostream>

using namespace std;

int M, C, price[25][25];
//1
int shop(int money, int g) {
  if (money < 0) return -1000000000;
  if (g == C) return M - money;
```

```
 // 2
  int ans = -1;
  for (int model = 1; model <= price[g][0]; model++)
       ans = max(ans, shop(money - price[g][model], g + 1));
  return ans;
 // 3
}

int main() {
  int i, j, TC, score;
  i=0, j=0, TC=0;
  fflush(stdout);

  cin >> TC;
  while (TC--) {
    scanf("%d %d", &M, &C);
    for (i = 0; i < C; i++) {
      scanf("%d", &price[i][0]);
      for (j = 1; j <= price[i][0]; j++) scanf("%d", &price[i][j]);
    }
   // 4
    score = shop(M, 0);
    if (score < 0) printf("no solution\n");
    else           printf("%d\n", score);
} }
```

c) Watch this:
Knapsack: https://www.youtube.com/watch?v=dN_gQYo9Uf8
Try it → Problem UVA 10130 Supersale. This problem is based on knapsack

Evaluation

Items	
Submission	**5**
Achievement	**10**
Total	**15**