Introduction to Competitive Programming

1.1 Objectives

To learn how to write solutions for problems in a programming contest.

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

a. Start reading chapter 1 in the text book.

a) Typing test

Try this typing test at http://www.typingtest.com and follow the instructions there on how to improve your typing skill.
Please submit your results in Ifolio → Tasks → Lab 1 or Openlearning

b) Log in to the PC Square

Log in to the PC Square using the given username and passwords.

c) In order to learn about input routines, submit programs for Case Study 1 until Case Study 5 via PC Square.

i. Case study 1: The number of test cases is given in the first line of the input

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

int main() {
	int TC, a, b;
	scanf("%d", &TC); // number of test cases
	while (TC--) { // shortcut to repeat until 0
		scanf("%d %d", &a, &b); // compute answer


		printf("%d\n", a + b); // on the
	}
}
```

Sample Input

2
1 2
4 2

Sample output

ii. Case study 2: Each line of input contains an integer N where 5 <= N <= 1,000,000. The input is terminated with a line containing 0.

Sample Input
5
10
118
211
0

Sample output:

Code:

```
#include <iostream>
using namespace std;

int main () {
   int n;
   int c;
   int i = 1;

   cin >> n;
   while ( n != 0) {
         c = n + 10;
         cout << "Case #" << i++ << ": " << c << endl;
         cin >> n;
   }

  return 0;
}
```

iii. Case Study 3: Input
The first line of input is T which is the number of cases, 1 <T< 1000. This is followed by the test cases. Each case consists of several lines. The first line contains 2 numbers, first is N which represents the number of relationships, N ≤ 200, and M which represents the number of politicians, 5 ≤M≤ 20. Each of the next N lines contains three integers that represent politician x,

his/her friend y (0 ≤x,y< M) and the strength of relationship z (1 ≤z≤ 4). Politician x=0 represents Khairy and politician x=M-1 represents the top politician

Sample Input:

```
2
3 2
1 1 2
2 3 1
4 2 1
3 5
0 1 2
1 3 4
4 2 1
```

Sample Output:

Code:

```
#include<cstdio>
#include<sstream>
#include<cstdlib>
#include<cctype>
#include<cmath>
#include<algorithm>
#include<set>
#include<queue>
#include<stack>
#include<list>
#include<iostream>
#include<string>
#include<vector>
#include<cstring>
#include<map>
#include<cassert>
#include<climits>
using namespace std;


#define sz size()
#define pb push_back

#define INF 10000000

int main() {
  int t, ncase = 0;
  cin >> t;
  while(t--) {

    int n, m;
    cin >> n >> m;

    while(n--) {
      int a, b, c;
      cin >> a >> b >> c;

    }
    printf("Case #%d:", ++ncase);
    cout <<" "<< m << endl;
  }



  return 0;
}
```

iv. Case study 4:
The input consists of up to `1000` test cases. Each test case describes a spot image of a rainforest area, which starts on a new line with a positive integer **N**, `(1 <= N <= 40)` indicating the dimension of the image. The next **N** lines of each test case delineate the pixelated representation of the image.

Sample Input:
6
100000
010101
000100
011000
000110
011000

Sample Output:

Code:
```
/*
 *  Muthukkaruppan Annamalai
 *  26 August 2013
 *
 *  AlKhawarizmi 2013 Problem Rainforest Canopy - PART OF THE PROGRAMS

//package rainforest_canopy;

import java.util.Scanner;
import java.lang.String;


public class case1 {

    static final int maxL = 40;

    public static void main(String[] args) {
        Scanner cin = new Scanner(System.in);
        short cn = 0;

        while (cin.hasNext()) {
            int nol = cin.nextInt();        // dimension

            cn++;


            if ((nol > 0) && (nol <= maxL)) {
                for (int i = 0; i < nol; i++) {
                    String ln = cin.next();

                System.out.println ("Case #" + cn + ": " + nol);
            }
        }

    }

  }
```

d) Problem_UVA_11450
Explore the solution for Problem: UVa 11450 Wedding Shopping. Type the C++ source code below as fast as possible and submit via the PC Square.

```
#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);
} }
```

e) Problem Sahur and Imsa' (ACM/ICPC 2013)
Getting started. Submit the solution for the following problem via PC Square

Hosted by: Universiti Teknologi PETRONAS
ACM-ICPC MALAYSIA
Al-Khawarizmi National Programming Contest 2013

F	SAHUR & IMSA'	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

Midhat is a Network Security Engineer, based in Sarajevo. He is assigned to do some important consultation projects around the globe in July and August 2013. It happened that Ramadhan (the fasting month for Muslim) falls during these months for the year 2013. Midhat has to travel to several cities – Istanbul, Kuala Lumpur, Tokyo, Melbourne, Sao Paolo and Chicago. Even though it is permissible for Muslims not to fast when travelling, he prefers to continue fasting in Ramadhan. Midhat has no problem on this matter except he needs to make himself awake for *sahur* (early breakfast before dawn) on his own, which he is used to be waked by his mother at home.

Midhat wants to have his *sahur* 45 minutes before *imsa'* (end time for taking *sahur*). He decided to set his travelling alarm clock 45 minutes before imsa', so that he can take his *sahur* in time, make his *Fajr* prayer and ready to work early. Since he's travelling to different parts of the world, the *imsa'* time differs from one city to another. Midhat wants to set his travelling clock to wake him up on time for all the cities he visits. Help Midhat by writing a program that will take one time stamp, in 24-hour notation, and print out a new time stamp, 45 minutes earlier, also in 24-hour notation.

Note: In 24-hour time notation, it starts with 0:00 (midnight) and ends with 23:59 (one minute before midnight). In the input and output we'll ignore the leading zeros and colon for simplicity. So 0:00 will be written as 0 0.

Input

The first line will contain number of test cases, **T**. After that **T** lines will follow, where each line will contain exactly two integers **H** and **M** (0 ≤ **H** ≤ 23, 0 ≤ **M** ≤ 59) separated by a single space, the input time in 24-hour notation. **H** denotes hours and **M** minutes.

Output

For each test case, the output contains a line in the format Case #x: followed by a sequence of integers, where x is the case number (starting from 1) and output one line with exactly two integers, the time 45 minutes before input time.

Sample Input	Sample Output
4	Case #1: 4 15
5 0	Case #2: 9 25
10 10	Case #3: 23 45
0 30	Case #4: 23 2
23 47	

Evaluation

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