Programming Design, Spring 2016
Homework 2

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

Please upload one PDF file for Problem 1 and two CPP files for Problems 2 and 3 to PDOGS at http://pdogs.ntu.im/judge/. Each student must submit her/his individual work. No hard copy. No late submission. The due time of this homework is ***2:00 am, March 7, 2016***. Please answer in either English or Chinese.

Before you start, please read Section 3.6 (about the `if-else` statement) by yourself in the textbook.[1]

Problem 1

(10 points) Consider the following two programs:

```
int a = 0;
cin >> a;
if(a % 2 == 1)
  cout << "a is odd";
else
  cout << "a is even";
```

```
int a = 0;
cin >> a;
if(a % 2 == 1)
  cout << "a is odd";
if(a % 2 == 0)
  cout << "a is even";
```

Explain why both programs can print out the right message to show whether `a` is odd or even. While they are both correct, which one is more efficient? Explain why.

Problem 2

(45 points) A classic ice-breaking game is played in the following rule. $n \in \mathbb{N}$ persons make a circle and call numbers 1, 2, 3, ... in terms. However, if the number one should call is a multiple of $m \in \{2, 3, ..., 9\}$ or has its last digit being m, she/he should clap her/his hands and call nothing. For example, if $m = 4$, the numbers that will be called are 1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 15, 17, 18, 19, 21, ... and so on. Therefore, suppose there are five persons in this game ($n = 5$), the first person will call 1, 6, 11, 21, ..., and the fourth person will call 9, 19, ...

The question is: For a game with n persons and a number m, what will be the sum of all numbers called by the kth person if the game ends at the number T? Let's consider the example in the previous paragraph. If $n = 5$, $m = 4$, $k = 4$, and $T = 25$, then the answer is 28 $(9 + 19)$; if $n = 5$, $m = 4$, $k = 1$, and $T = 30$, then the answer is 65 $(1 + 6 + 11 + 21 + 26)$.

Input/output formats

There are 15 input files. In each file, there are 4 nonzero integers n, m, k, and T in one single line. Two consecutive integers are separated by one white space. We have $n \in \{1, 2, ..., 1000\}$, $m \in \{2, 3, ..., 9\}$, $k \in$ N, $k \leq n$, and $T \in \{1, 2, ..., 100000\}$. Given these input values, your program should print out one integer as the answer according to the above rule.

To read inputs from the files on PDOGS, simply use `cin` as if a user will enter those input values according to the above rules. Then simply output your results using `cout` as if you are required to print out values on your screen according to the above rules. PDOGS will execute your program for 15 times, each time with a different input file. The 15 sets of outputs will be graded separately.

[1]The textbook is *C++ How to Program: Late Objects Version* by Deitel and Deitel, seventh edition.

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use techniques covered in lectures AND the `if-else` statement. NO other techniques are allowed. Finally, you should write relevant comments for your codes.

Grading criteria

- 30 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.
- 15 points will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.

Problem 3

(45 points) Alice, Bob, Cindy, and Dororo are four pirates. One day, they found a lot of equally large diamonds put in $n \in \mathbb{N}$ boxes. While some boxes contained more diamonds than others, all boxes looked the same. While they were happy, they faced the question of dividing the diamonds into four pieces as equally as possible.

They decided to use the following rule. First, close all the boxes and put them in a list. Then they open each box one by one. Once the ith box is opened, the number of diamonds inside x_i is counted. The box and all the diamonds inside are then given to the person who currently owns the least diamonds. If there is a tie, the box should be assigned to the person whose name is ordered first alphabetically among those who owns the least diamonds. The process continues until all boxes are open and assigned.

As an example, suppose there are $n = 15$ boxes, and the numbers of diamonds in the boxes are 4, 3, 7, 15, 4, 3, 14, 26, 8, 10, 2, 12, 16, 20, and 1 (in the order of the boxes in the list). Then according to the rule, these boxes of diamonds will be assigned to them in the following way:

Name	1	2	3	4	Sum
Alice	4	3	14	16	37
Bob	3	4	26	N/A	33
Cindy	7	8	10	20	45
Dororo	15	2	12	1	30

You are asked to find the total numbers of diamonds they obtain at the end and list them in the ascending order. In this example, your should answer 30, 33, 37, and then 45. Note that though Dororo gets the most diamonds when each of them is assigned their first box, at the end he gets the least diamonds. This reflects an old saying: "Don't celebrate too early!"

Input/output formats

There are 15 input files. In each file, there are $n + 1$ nonzero integers n, x_1, x_2, ..., and x_n in one single line. Two consecutive integers are separated by one white space. We have $n \in \{1, 2, ..., 10000\}$ and $x_i \in \{1, 2, ..., 100\}$ for all $i = 1, ..., n$. Given these input values, your program should print out four integers as the answer according to the above rule.

To read inputs from the files on PDOGS, simply use `cin` as if a user will enter those input values according to the above rules. Then simply output your results using `cout` as if you are required to print out values on your screen according to the above rules. PDOGS will execute your program for 15 times, each time with a different input file. The 15 sets of outputs will be graded separately.

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use ANY technique. Finally, you should write relevant comments for your codes.

Grading criteria

- 30 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.
- 15 points will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.