GECG10069(561085) F25

Midterm Examination

FULL SCORES:

100 %

EXAMINATION TIME:

9：00～12：00, total 180 minutes

INSTRUCTIONS:

- You may pick an arbitrary number of problems to solve with a total score of up to 100%.
- Please strictly follow the input and output specifications described in each problem. Any deviation that causes test case errors will result in point deductions, for which the student is fully responsible.
- Name your files by problem number: Problem 1 → p1.cpp, …
- Zip all program files into one archive named with your student ID (e.g., 411511004.zip or 411511004.7z). Submit through **NTU Cool**. If the system is slow, other methods will be announced.
- You have a 10-minute grace period (until 12:10 PM) to submit.
- **Submissions after 12:10 PM will not be accepted.**
- You are allowed to open any notes or books. However, you are **not allowed to copy others' code AND browse on the internet**. Otherwise, you'll get 0% in this examination and be punished by the school regulations.
- Carefully read the statements and requirements of each problem.

PROBLEM 01 (10%)

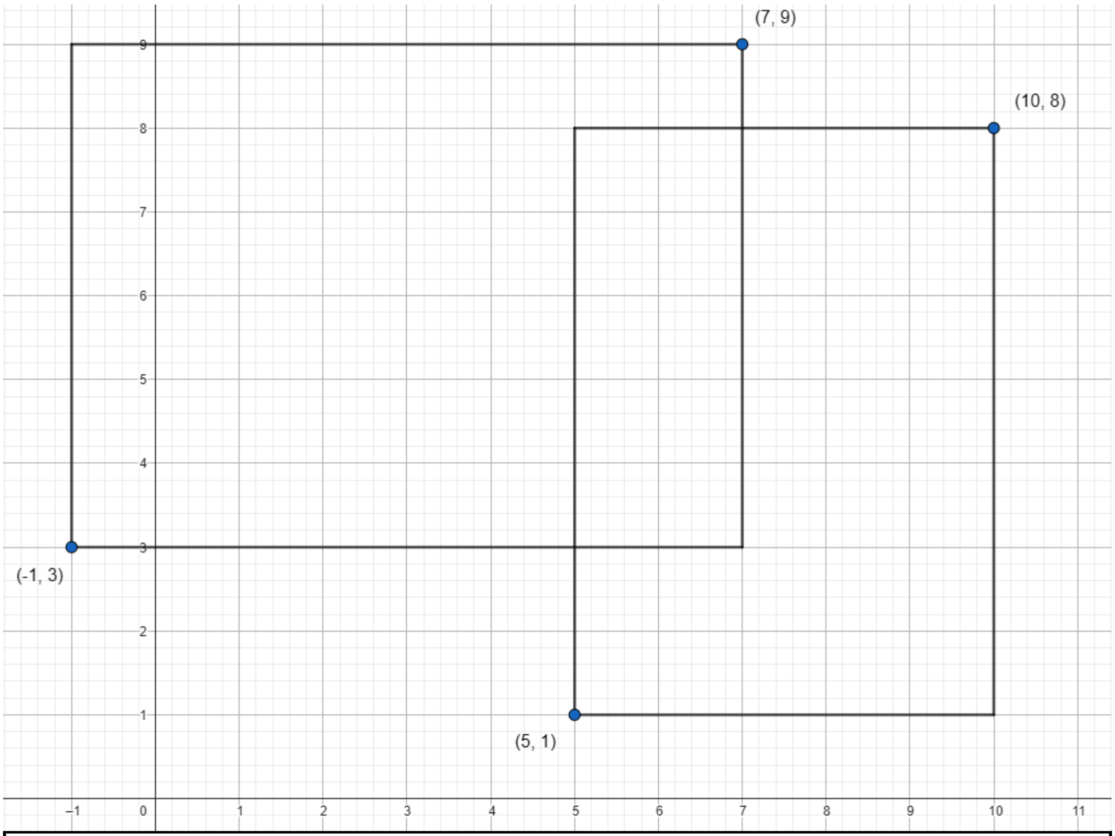
Sum of digits. Calculate the sum of every digit of a positive integer.

4583 → 4 + 5 + 8 + 3 = 20

Sample Input - 1
4813
Sample Output - 1
16
Sample Input - 2
12345
Sample Output - 2
15
Sample Input - 3
7543824
Sample Output - 3
33

PROBLEM 02 (10%)

Given two points of two rectangles, please calculate the overlapping area.
Input format: The First four numbers are the x, y of the bottom left and top right points of the same rectangle. The last four numbers are the x, y of the bottom left and top right points of the other rectangle.
The area will always be greater than 0.



Sample Input - 1
-1 3 7 9 5 1 10 8
Sample Output - 1
10
Sample Input - 2
1 3 8 9 6 7 9 10
Sample Output - 2
4
Sample Input - 3
3 -1 7 9 -1 2 5 11
Sample Output - 3
14

PROBLEM 03 (10%)

Create a C++ program that calculates the least common multiple (LCM) of two integers. The inputs are two integers a and b, and a function named find_LCM will return their LCM.

(The least common multiple of two positive integers is the smallest positive integer that is divisible by both numbers.)
Hint: 最小公倍數

Input:

- Two integer numbers (a & b) separated by a space
- Both a and b are integers, and at least one of them is **nonzero**. (That is, you only need to handle the case where one of the numbers is 0.)

Output:

- The least common multiple (LCM) of a and b. The result of this problem will not exceed the range of an **integer.**

Sample Input - 1
10 12
Sample Output - 1
60
Sample Input - 2
5 7
Sample Output - 2
35
Sample Input - 3
0 30
Sample Output - 3
0

PROBLEM 04 (10%)

Create a program that converts an amount in New Taiwan Dollars (NTD) into U.S. Dollars (USD) and breaks it down into denominations.
The exchange rate is 1 USD = 30 NTD.

Your program should first convert the input NTD amount into USD (using floating-point division), then calculate:

- *a*: number of 100-dollar bills
- *b*: number of 10-dollar bills
- *c*: number of 1-dollar bills
- *d*: number of 10-cent coins
- *e*: remaining NTD after exchange

- Exchange rule (uniqueness requirement):
  Always use the largest denominations first to exchange as much as possible.
  Formally, decompose the USD amount (in cents) greedily in this order:
  - \$100 bills →
  - \$10 bills →
  - \$1 bills →
  - 10-cent coins.

Finally, output the results in the following format:

```
a = …
b = …
c = …
d = …
e = …
```

You can follow this flow:
n = 5000
⇨ The amount in USD (U)= n/30 = 166.66666666……
⇨ a = integer part of U / 100 = 1
⇨ b = integer part of (U – a*100) / 10 = 6
⇨ c = integer part of (U – a*100 – b*10) / 1 = 6
⇨ d = integer part of (U – a*100 – b*10 – c*1) * 10 = 6
⇨ e = round of (U – a*100 – b*10 – c*1 – d*0.1)*30 = 2

Input:

- A single integer *n* representing the amount in New Taiwan Dollars (NTD).
- n is a non-negative integer. Test cases may include values that are not divisible by 30.

Output:

- The number of each denomination (a, b, c, d) and the remaining NTD (e).

Sample Input - 1
5000
Sample Output - 1

a = 1
b = 6
c = 6
d = 6
e = 2

Sample Input - 2
90
Sample Output - 2

a = 0
b = 0
c = 3
d = 0
e = 0

Sample Input - 3
95
Sample Output - 3

a = 0
b = 0

```
c = 3
d = 1
e = 2
```

PROBLEM 05 (10%)

Write a simple time-conversion program.
Given a conversion type and time values, your program converts between total seconds and hours–minutes–seconds (H:M:S) formats.
There are two conversion types:

- Type 1: Convert from total seconds to H:M:S.
- Type 2: Convert from H:M:S to total seconds.

Input Format

The input format depends on the conversion type.
Type 1 — Input should contain two integers.

- The first integer is 1, which represents type 1.
- The second integer represents the total number of seconds.

Type 2 — Input should contain four integers.

- The first integer is 2, which represents type 2.
- The next three integers represent hours, minutes, and seconds, respectively.

Output Format

Type 1:
Output the time in the format HH:MM:SS
(with leading zeros, e.g., 01:01:01, 00:59:30).

Type 2:
Output a single integer representing the total number of seconds.

Notes:

- ➢ Assume all inputs are valid and non-negative
- ➢ Hours are not required to wrap around (Hours can be more than 24)
- ➢ The input fits in 32-bit signed integers
- ➢ Output for Type 1 must include leading zeros

Sample Input - 1
1 3661
Sample Output - 1
01:01:01
Sample Input - 2
2 1 1 1
Sample Output - 2
3661
Sample Input - 3
1 888
Sample Output - 3
00:14:48

PROBLEM 06 (10%)

The number pattern problem prints a square pattern of concentric layers of numbers.The outermost layer starts with the number n, and each inner layer decreases the number by 1 until reaching the center, which is 1.

Input Format

`A single integer n (1 ≤ n ≤ 9).`

Output Format

Print the concentric square pattern of numbers.
The output should be a square of size (2n - 1) × (2n - 1).

Sample Input - 1
2
Sample Output - 1

2 2 2
2 1 2
2 2 2

Sample Input - 2
3
Sample Output - 2

3 3 3 3 3
3 2 2 2 3
3 2 1 2 3
3 2 2 2 3

```
3 3 3 3 3
```

Sample Input - 3

```
5
```

Sample Output - 3

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

PROBLEM 07 (10%)

You're building a delivery fee calculator. The final fee should be calculated based on:

(1) the price of the meal.
(2) current weather conditions (raining or not).
(3) the customer's VIP status.

All fees are integers. Apply the steps in this order:

1. Base fee by meal price

   If meal_price < 200 → base fee = 100

   If 200 ≤ meal_price < 500 → base fee = 50

   If meal_price ≥ 500 → base fee = 20

2. Rain surcharge

   If is_rain = true → add 20

   If is_rain = false → add 0

3. VIP discount

   If is_vip = true → apply 50% off to the current subtotal (integer division)

If is_vip = false → no discount

4. Lower bound

Ensure final fee ≥ 15

Input Format

Three lines:
Line 1: meal_price (integer)
Line 2: is_rain (boolean: true or false, lowercase)
Line 3: is_vip (boolean: true or false, lowercase)

Output Format

One line: the final delivery fee (integer).

Constraints

0 < meal_price < 100000
is_rain ∈ {0, 1}
is_vip ∈ {0, 1}

Sample Input - 1
180 1 0
Sample Output - 1
120
Sample Input - 2
520 1 1
Sample Output - 2
20
Sample Input - 3

250 0 0
Sample Output - 3
50

PROBLEM 08 (10%)

Read an integer seed and an integer R (roll count). Initialize the C RNG with std::srand(seed) and roll a fair six-sided die exactly R times.

Report:

1. the counts of each face 1..6,
2. the sample mean,
3. the unbiased sample variance.

Input Format

Two lines:
Line 1: seed (integer)
Line 2: R (number of rolls, integer)

Output Format

COUNTS: c1 c2 c3 c4 c5 c6

Counts of faces **1..6** over the R rolls (integers).

MEAN ($\bar{x}$): m.mm

Sample mean of the R rolls, **2 decimals** (小數點後 2 位), standard rounding.

$$\bar{x} = \frac{1}{R}\sum_{i=1}^{R} x_i$$

MAX COUNT FACE:

Print the most frequent face.

MIN COUNT FACE:

Print the least frequent face.

For the test cases, seeds are chosen so that no two faces occur the same number of times over R rolls.

Constraints

- $0 < \text{seed} < 1{,}000{,}000$
- $1 \leq R \leq 1{,}000{,}000$

Hint:

- - std::srand(seed)
- - std::rand()
- - std::setprecision(x)
- - std::fixed (remember to #include <iomanip>)

Sample Input - 1
1234 70
Sample Output - 1
COUNTS: 12 11 16 14 10 7 MEAN: 4.00 MAX COUNT FACE: 3 MIN COUNT FACE: 6
Sample Input - 2
228 88000
Sample Output - 2
COUNTS: 14691 14812 14820 14480 14565 14632 MEAN: 3.49 MAX COUNT FACE: 3 MIN: COUNT FACE: 4
Sample Input - 3

7777 7777
Sample Output - 3
COUNTS: 1330 1253 1265 1322 1233 1374 MEAN: 3.51 MAX COUNT FACE: 6 MIN: COUNT FACE: 5

PROBLEM 09 (10%)

You have been hired by the CIA Hotel to write a simple billing program.

Your task is to calculate the total amount for each guest
and **print a formatted table**.

Two monetary values (Price, Total Price)
must be **rounded and displayed with exactly two digits after the decimal point.**

Input :

- The input consists of 3 lines in the following format:

```
<name_1(string)> <price_per_day_1(double)> <days_1(int)>
<name_2(string)> <price_per_day_2(double)> <days_2(int)>
<name_3(string)> <price_per_day_3(double)> <days_3(int)>
```

Output Format :

- Print a table with the following **four** columns,
  each with a **width of 10 characters**:
  - Name — **left** aligned
  - Price — **right** aligned, two decimal places
  - Days — **right** aligned
  - Total — **right** aligned, two decimal places
    (Total = `<price_per_day(double)> * <days(int)>`)
- The first line should display the column headers.
  The next three lines should display the data for each guest.

Constraints -

- *length (name) < 7*

- *Each name consists of only **one** word*
- *0 < all the input values < 100000*
- *0 < all the calculation results < 100000*

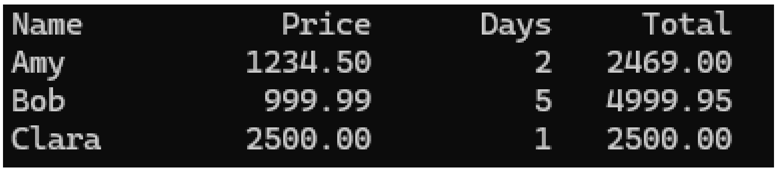
Hint:

- **`setprecision()` automatically performs rounding,**
- **so you don't need to round the value manually.**

Sample Input - 1

```
Amy   1234.5   2
Bob   999.99   5
Clara  2500    1
```

Sample Output - 1

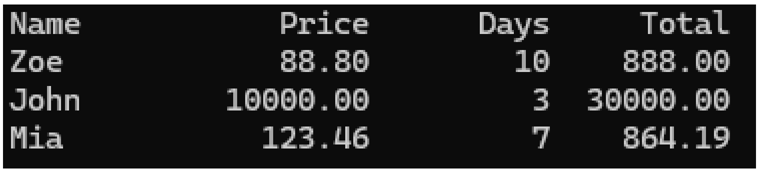


```
Name            Price       Days     Total
Amy           1234.50          2   2469.00
Bob            999.99          5   4999.95
Clara         2500.00          1   2500.00
```

Sample Input - 2

```
Zoe   88.8     10
John 10000     3
Mia  123.456  7
```

Sample Output - 2



```
Name            Price       Days     Total
Zoe             88.80         10    888.00
John         10000.00          3  30000.00
Mia            123.46          7    864.19
```

Attention:

The product of 123.456 and 7 is 864.192,

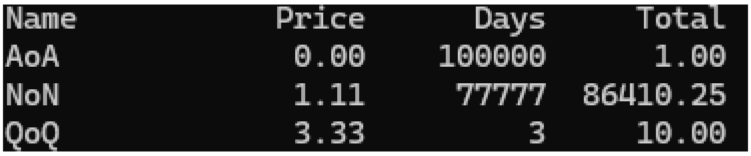
which rounds to 864.19 when rounded to two decimal places.

In the “price” column of your output, you must display 123.46,
which is the result of rounding 123.456 to two decimal places.

Sample Input - 3

```
AoA    0.00001      100000
NoN    1.111        77777
QoQ    3.333333     3
```

Sample Output - 3



```
Name                 Price        Days       Total
AoA                   0.00      100000        1.00
NoN                   1.11       77777    86410.25
QoQ                   3.33           3       10.00
```

Attention:
The product of 0.00001 and 100000 is 1,
which rounds to 1.00 when rounded to two decimal places.

But in the “price” column of your output, you must display 0.00,
which is the result of rounding 0.00001 to two decimal places.

PROBLEM 10 (10%)

The kingdom is holding a small talent show. Each performer has a score **g** and a **name** (**which may contain spaces**). The king has a special number **k**, and he wants to find the **performer with the highest score that is divisible by k**. Write a program to help the king find this performer.

Input :

- The input several lines in the following format:
  - Line 1: an integer **k**
  - Starting from line 2, the input comes in **pairs of lines**:
    - First line of each pair: the score **g** (`int`)
    - Second line of each pair: **name** (`string`)
  - **When g = 0, stop reading.**

Output :

- Print the **name** of the performer with the largest score divisible by **k**.
- If no score is divisible by **k**, print: “**None**”

Hint :

- You can use the following way to assign a string value to another one:

```
std::string a, b;
std::cin >> a;
b  =  a;          // This copies the entire string stored in a to b
```

Constraints -

- *0 < g ,k < 10000, the last g = 0,*
- *all g values are unique*
- *Number of input lines < 50*
- *length (each name) < 50*

Attention:

If the sample input lines contain trailing whitespace (行尾多餘空格) on your end,

please remove it manually to avoid errors between **cin** and **getline()**

Sample Input - 1

```
5
10
Alice Smith
7
Bob
15
Charlie Brown
0
```

Sample Output - 1

```
Charlie Brown
```

Scores divisible by **k** = 5 are : **{**10 (Alice Smith) , 15 (Charlie Brown)**}**.
The largest among them is 15, so the output is Charlie Brown.

Sample Input - 2

```
5
6666
Tan Gan Dee
7777
ALA Hsieh
0
```

Sample Output - 2

```
None
```

None of the scores {6666, 7777} are divisible by **k** = 5.
Therefore, the program prints “None”.

Sample Input - 3

```
1
99
Naval_officer 1
88
Naval_officer 2
77
Naval_officer 3
66
Naval_officer 4
55
Naval_officer 5
```

```
999
Naval_officer 6
4
Naval_officer 7
2
Naval_officer 8
789
Naval_officer 9
0
```

Sample Output - 3

```
Naval_officer 6
```

All the scores are divisible by **k** = 1.
Therefore, the program prints “Naval_officer 6” with the highest score.