Lab 02 Input – Processing–Output

This Lab sheet involves exercises in 3 parts:
Guided Exercises (Q1-3) ➔ Skill-Drilling Exercise (Q4) ➔ Take-Home Exercises (Q5-8)

Guided Exercises

Q1. Understand the Input Statement and the Input Stream [Given: **Lab02_Q1.cpp**]

Some facts about "`cin >>..`"
- Excess input from cin not "consumed" by the variables are kept in cin and are left for the next >> operation.
- Leading whitespaces are ignored. (whitespaces mean spaces, tabs, new line)
- Numeric values are delimited by white-spaces.

[Lecture notes Chapter 2 III. I/O Statements]

Your tasks: Read the following program; then finish Task 1, Task 2, Task 3, Task 4 below one by one.

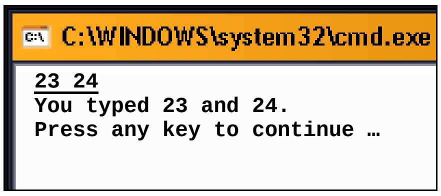
```
int main()
{
    int x1, x2;

    cin >> x1 >> x2;
    cout << "You typed " << x1 << "and " << x2 << endl;
    return 0;
}
```

`cin >> x1 >> x2;` is the same as
```
cin >> x1;
cin >> x2;
```

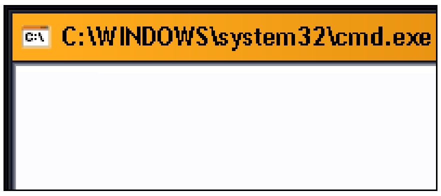
Task 1: Run the program according to Case 1 below and study what has happened. Repeat for Case 2.

Case 1: Input "`23<space>24<enter>`".



```
23 24
You typed 23 and 24.
Press any key to continue …
```

Case 2: Input "`23<enter>24<enter>`".



Task 2:
Fill in this box

Case 1: ←Task 3: Read the explanations below carefully.

- cin >> x1 waits for my input until I press <enter>.
- After I type 23<space>24<enter>, cin >> x1 takes 23. <space>24<enter> is left in the input stream.
- Then cin >> x2 starts.
- The input stream still contains <space>24<enter>, so cin >> x2 checks the input stream (not to wait for my input).
- cin >> x2 discards the leading whitespace (<space>) and takes 24.
- <enter> is left in the input stream. (The program will not take any further input. So this <enter> will be forgotten finally.)

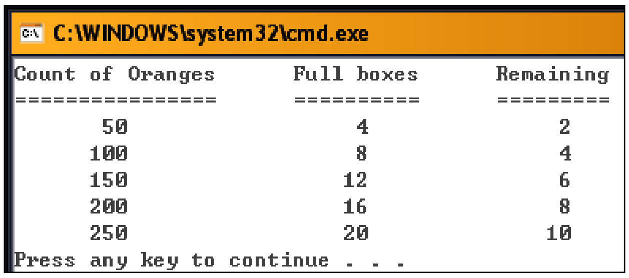
Case 2: ←Task 4: Fill in the blanks below.

- cin >> x1 waits for my input until I press ___________.
- After I type ___________, cin >> x1 takes _____. ___________ is left in the input stream.
- Then cin >> x2 starts.
- The input stream still contains ___________, so cin >> x2 checks the input stream.
- cin >> x2 discards the whitespace (___________), the input stream becomes empty, cin >> x2 cannot get any content into x2.
- So cin >> x2 waits again for my input until I press ___________.
- After I type ___________, cin >> x2 takes _____.
- <enter> is left in the input stream. (The program will not take any further input. So this <enter> will be forgotten finally.)

*Any question? **Ask!***

Q2. Packing Oranges:

Assume that we can pack **12** oranges per box.
We will create the table as shown here:


C:\WINDOWS\system32\cmd.exe

```
Count of Oranges        Full boxes        Remaining
================        ==========        =========
       50                    4                2
      100                    8                4
      150                   12                6
      200                   16                8
      250                   20               10
Press any key to continue . . .
```

Given framework **[Course web - Lab02_Q2_Given.cpp]** :

```
#include <iostream>
using namespace std;
int main()
{
    int nOranges;

    //ruler: 123456789012345678901234567890123456789012345678901234567890
    cout << "Count of Oranges       Full boxes       Remaining" << endl;
    cout << "================       ==========       =========" << endl;
    //sample:       50                   4                2

    nOranges=50;
    cout  << nOranges << " " <<____________ << " " << ____________ << endl;

    nOranges+=50; //same as nOranges = nOranges + 50;
    cout  << nOranges << " " <<____________ << " " << ____________ << endl;
    ...
}
```

(a) Complete the program (Version 1)

Step 1. For each data row, replace the **blanks**(_______) with **nOranges/12** and **nOranges%12**.

Run the program and study how the code works (though the columns are not aligned yet.)

Step 2. To align the columns:

```
|      50                 4                2|
|<------>|<----------------->|<-------------->|
    9             19                 16
```

(a.) apply `setw(..)` as shown below:

```
        setw(9)           setw(19)            setw(16)
           ↓                 ↓                   ↓
cout <<      << nOranges <<      << nOranges/12 <<      << nOranges%12 << endl;
```

The use of **`setw(..)`** is to set the width of the display field in next << operation.
(Note also that numbers are right-aligned.)

(b.) add **`#include <iomanip>`** at the beginning of the program.

↑ This library supports i/o formatting (setw, setprecision, fixed, etc..)

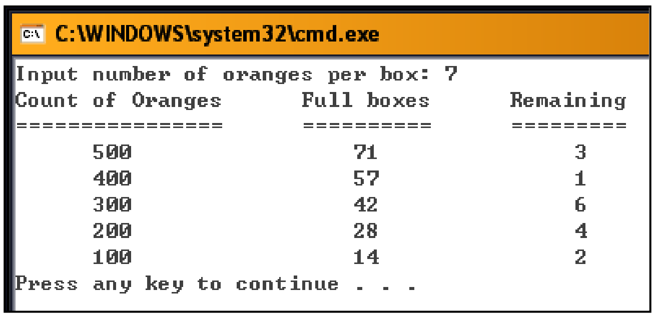
Compile and run it. Check the output.

(b) Modify the program for Version 2:

- Calculate based on user's input.
  (You'll need one more variable.)

- Count down from 500, everytime by
  `nOranges -= 100;`

Complete it and test it in PASS.


C:\WINDOWS\system32\cmd.exe

```
Input number of oranges per box: 7
Count of Oranges        Full boxes        Remaining
================        ==========        =========
      500                   71                3
      400                   57                1
      300                   42                6
      200                   28                4
      100                   14                2
Press any key to continue . . .
```

Q3. Hours - Minutes – Seconds

A little bit warm-up (Fill in the blanks below – by hand-calculation):

```
(i)     139 seconds = _________ minutes _________ seconds
(ii)    180 seconds = _________ minutes _________ seconds
(iii)    59 seconds = _________ minutes _________ seconds
```

(a) Write a program that reads in the number of seconds and converts it to minutes and seconds.

Sample input and output:

```
Please enter the number of seconds: 5000
5000 second(s) =  83 minute(s) 20 second(s)
```

** Underlined contents are input by user.

Hint:
- Modulus (remainder) `%` : Eg. 76%60 gives 16
  Integer division `/` : Eg. 76/60 gives 1 (integer part only)
- Try setting variables, eg.,

```
int input, min, sec;
..
min = ..;
sec = ..;
```

Check the correctness of your program with at least 6 test cases: 0, 5, 65, 120, 5000, ….
Put a tick (✓) here when finished: ☐

(b) This time you will modify the program to convert the input to hours, minutes and seconds.
(Variables: `int input, hr, min, sec;`)

Hand-calculation:

```
(i)     3600 seconds = _______ hours _______ minutes _______ seconds
(ii)    3640 seconds = _______ hours _______ minutes _______ seconds
(iii)   3852 seconds = _______ hours _______ minutes _______ seconds
```

Sample input and output of the program:

```
Please enter the number of seconds: 5000
5000 second(s) = 1 hour(s) 23 minute(s) 20 second(s)
```

Complete the program and test it in PASS.

Checkpoints (Make sure you fulfill the following items and put a ✓ where appropriate.)

Name: _________________________

- ☐ I have successfully finished Q1 - Q3, and obtained 100% correct in PASS.
- ☐ I understand how `cin` works.
- ☐ I know how to use **`setw`**, and that it needs **`#include <iomanip>`**
- ☐ **`setw`** doesn't have a lasting effect. So we need to apply **`setw`** again when needed.
- ☐ I've applied %, +=, and -= in program code.
- ☐ **% is useful**.

Are you familiar with the procedures covered in Q1 - Q3?
If you have any doubt about the above, don't hurry for the next exercise. Ask and clarify.
It may also be better to REDO them again later. After finishing the first trial, **`re-doing`** won't take you much time but is worthy.

** Submission: Put down this handout on the front desk and get the **Take-Home Exercise**

Take-Home Exercises

Before proceeding, check (✓) the points below:

☐ I'm okay with the lecture notes, examples, and lecture exercises covered during all previous lectures.
☐ I'm okay with all previous lab exercises.

To assure your progress, ask yourself this question:

After finishing the first trial, it should be much easier. **`Re-doing`** won't take you much time and is **worthy**.

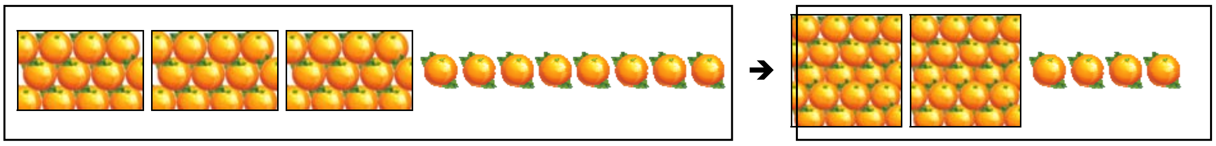
Do you have confidence to redo Q1-3 with minimum hints from the lab-sheet? Try!!

Q4. Re-packing Oranges

Suppose an amount of oranges have been packed with boxes of size b1 (ie. each box can store b1 oranges), resulting in totally n boxes with r oranges unpacked.

Now we need to repack them using other new boxes that have a new size b2.

For example:

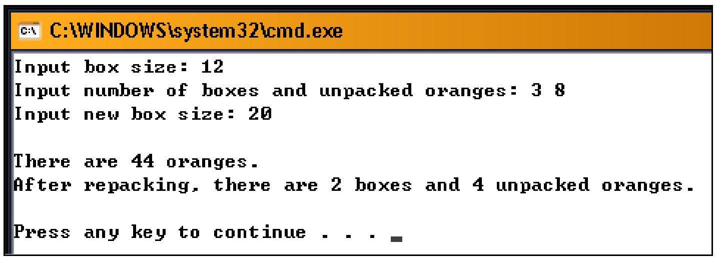


Currently $\overset{\uparrow}{12}$ oranges per box, count of packed boxes is 3, unpacked 8 oranges — New box size : 20 oranges per box

b1 n r b2

Your task: Complete the given program that asks the user to input b1, n, r, and b2, and outputs the repacking result.

The input/output of the program should follow this example:



```
C:\WINDOWS\system32\cmd.exe
Input box size: 12
Input number of boxes and unpacked oranges: 3 8
Input new box size: 20

There are 44 oranges.
After repacking, there are 2 boxes and 4 unpacked oranges.

Press any key to continue . . .
```

← User inputs

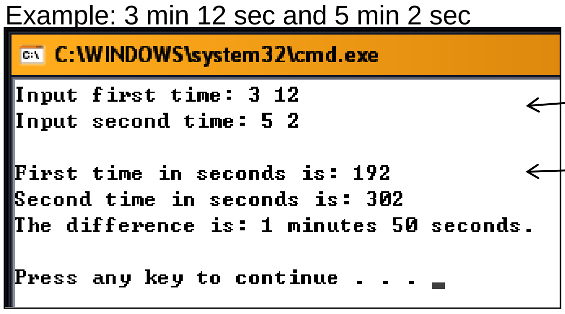
← Calculation results

Question 5. Difference between 2 times in min-sec

Complete the given program that asks for 2 time values (min-sec) and shows the difference between them. Assume that the first time value is earlier than the second one (i.e., no need to compare them).

The input/output of the program should follow these examples:

Example: 3 min 12 sec and 5 min 2 sec



```
C:\WINDOWS\system32\cmd.exe
Input first time: 3 12
Input second time: 5 2

First time in seconds is: 192
Second time in seconds is: 302
The difference is: 1 minutes 50 seconds.

Press any key to continue . . .
```

← User inputs

← Calculation results

Q6 Tutorial Grouping

Complete the given program that asks for the number of students and groups them evenly into 5 groups: L01, L02, L03, L04, L05. In case the count cannot be divided evenly, the first few groups can have 1 student more than the remaining groups.

Example 1:

```
Input total number of students: 103

[Grouping result]
L01: 21 students
L02: 21 students
L03: 21 students
L04: 20 students
L05: 20 students

Press any key to continue . . .
```

Example 2:

```
Input total number of students: 11

[Grouping result]
L01: 3 students
L02: 2 students
L03: 2 students
L04: 2 students
L05: 2 students

Press any key to continue . . .
```

Hint:

- Use more variables (say, use 1 variable for the count of each group).
- Do the calculations step by step.

Q7 Formatted Integer

Complete the given program that asks for an integer (assumed range: 1000000-9999999) and then shows it with commas as shown in the example below.

Note: you do not need to check whether the input falls within the range.

Example:

```
Input a number (1000000 - 9999999): 8105096

The number with commas: 8,105,096

Press any key to continue . . .
```

Hint:

- You will need to use setfill and setw. (Include iomanip). If needed, please review the lecture notes.
- / is useful: integer / integer is an integer (fractional part truncated).
- % is useful.

Q8 Ticketing

Complete the given program that calculates the total cost for a number of tickets (*Normal price*: $100).

(i) Standard discount (10%) for all customers. That is, pay $90 per ticket.

(ii) Special offer for members: buy 3 (at normal price) get one free. That is, pay $300 for every 4 tickets.

The program should run as shown in the examples below:

```
Membership [y/n]: n

Number of tickets: 10
Please pay: $900.0

Press any key to continue...
```

Example 1 (non member)

Apply (i) only
$90 x 10 = $900
🗎🗎🗎🗎🗎🗎🗎🗎🗎🗎

```
Membership [y/n]: y
Tips: Buy 3 get 1 free

Number of tickets: 10
Please pay: $780.0
(Average: $78.0 per ticket)

Press any key to continue...
```

Example 2 (member)

(ii) for 8 tickets, (i) for 2 tickets
$600+$180=$780
🗎🗎🗎🗎 🗎🗎🗎🗎 🗎🗎

Note: Show the payment **with 1 digit after the decimal point**.
If the customer is a member, show *"Tips: Buy 3 get 1 free"* and the average cost as in Example 2.