CSDC101 Fundamentals of Computing
Laboratory Exercise 7

PROBLEMS

1) Count Me Down

Write a program that will get and print a countdown. Be able to have two functions: (1) **int get_count(int count) function** - is used to get the count down; (2) **void print_count(int count)** - is used to print the count down. Include a loop that lets the user repeat this process for new countdown values.

SAMPLE INPUT AND OUTPUT

```
Enter number: 5
T minus 5 and counting
T minus 4 and counting
T minus 3 and counting
T minus 2 and counting
T minus 1 and counting
Another[y/n]? y
Enter number: 3
T minus 3 and counting
T minus 2 and counting
T minus 1 and counting
Another[y/n]? n
```

2) Let's fly

Write a program to assign passengers seats in an airplane. Assume a small airplane with small numbering as follows:

```
1 A B C D
2 A B C D
3 A B C D
4 A B C D
5 A B C D
6 A B C D
7 A B C D
```

The program should display the seat pattern with an **X** marking the seats already assigned. For example, after seats 1A, 2B, and 4C are taken, the display should look like this:

```
1 X B C D
2 A X C D
3 A B C D
4 A B X D
5 A B C D
6 A B C D
7 A B C D
```

INPUT FORMAT

Input begins with an integer N indicating the number of passenger seat requests that follow. Succeeding are N lines of input in the format nc, where n indicates the row number and c indicates the column letter of the seat being requested.

OUTPUT FORMAT

If the requested seat is available, display the seat view as shown above. Mark the reservation with an uppercase '**X**'. If the requested seat is not available, print "**Sorry, not available!**" without the quotation marks and prompt the user to choose a seat again. Separate output for each passenger request with an empty line. Assume all seats are available at the start of the program.

CONSTRAINTS

```
1 <= N <= 100
1 <= n <= 7
c could be A, B, C, or D
```

SAMPLE INPUT

```
3
3D
7C
3D
```

SAMPLE OUTPUT

```
 1  A  B  C  D
 2  A  B  C  D
 3  A  B  C  X
 4  A  B  C  D
 5  A  B  C  D
 6  A  B  C  D
 7  A  B  C  D

 1  A  B  C  D
 2  A  B  C  D
 3  A  B  C  X
 4  A  B  C  D
 5  A  B  C  D
 6  A  B  C  D
 7  A  B  X  D

Sorry, not available!
```

3) Health Bar

Yen is creating a health bar for a game. The problem is, she can't quite get it right. Help her!

She wants a player's health bar to be represented by 10 equal signs enclosed in square brackets, as in: [==========]. Each equal sign represents 10% of the player's health.

Yen wants you to create a program that would simulate the stream of changes of a health bar after a series of damages and regenerations have been applied. The health bar always begins with 100% health.

INPUT FORMAT

Input consists of an arbitrary pairs of input in the format cn, where c could either be a plus sign (+) to indicate regeneration or a minus sign (−) to indicate damage, and n is an integer indicating the percentage of regeneration or damage.

OUTPUT FORMAT

For each input, display the health bar depicting its current state after the regeneration or damage has been applied followed by the percentage value.

If regenerating results to a health bar more than 100%%, represent it as 100%
(e.g. [==========] 100%).

If being damaged results to the health being equal to 0 or going below 0, represent it as an empty health bar - 10 spaces enclosed in square brackets - with the string "DEAD" without the quotation marks (e.g. [ ] DEAD).

If applying the regeneration or damage results to a value that is not a multiple of 10, represent in the health bar its corresponding rounded down value but display the actual percentage (e.g. applying -5 to a 100% health bar results to 95%. The output should look like: [=========] 95%).

The health bar and the corresponding percentage value/string "DEAD" should be separated by a single space.

PROGRAM SPECIFICATION

Program must have the following:

1. A void function called **regenerate** that takes two pass-by-reference arguments s and p, where s is the string representation of the health bar and p is an integer indicating the equivalent percentage value. It takes a third pass-by-value integer argument n indicating the amount of damage to be dealt. The function applies the damage and updates the health bar's string and percentage representations.

2. A void function called **regenerate** that takes two pass-by-reference arguments s and p, where s is the string representation of the health bar and p is an integer indicating the equivalent percentage value. It takes a third pass-by-value integer argument n indicating the amount of healing to be dealt. The function applies the healing and updates the health bar's string and percentage representations.
3. A void function called **display** that takes two pass-by-value arguments s and p, where s is the string representation of the health bar and p is the equivalent percentage value. The function displays the required output of the program.

CONSTRAINTS

c could be $+$ or $-$

$1 <= n <= 100$

SAMPLE INPUT	SAMPLE OUTPUT
-5 +25 +10 -30 +18	[=========] 95% [==========] 100% [==========] 100% [=======] 70% [========] 88%

4) Visualizing Linear Function

Linear function can be expressed using the slope-intercept form: $y(x) = ax + b$, where a and b are some constants.

For instance, with $a=3, b=2$ function will yield values $y = 2, 5, 8, 11...$ for $x = 0, 1, 2, 3...$

Your task is to determine a and b by two points, belonging to the function. Recall your notes in math, a represents the *slope* and b represents the *y-intercept*.

Create a program that will determine the *slope* and the *y-intercept* given two pairs of values *(x1, y1), (x2, y2)* .

There should be two functions: (1) **int f_slope(int x1,int y1,int x2,int y2)** - is used to find the slope; (2) **int f_intercept(int x1, int y1, int a)** - is used to find the y-intercept.

**Note:*

To compute for the slope: ***a = (y2-y1) / (x2-x1)***

To compute for the y-intercept: ***b = y1 - m * x1***

Include a loop that lets the user repeat this process for new values for linear function.

SAMPLE INPUT AND OUTPUT

```
Enter first coordinates(x1,y1): 2 4
Enter second coordinates (x2,y2): 4 5
The slope of a line going through the point (2,4) and the point (4,5) is :0
The y-intercept of the point (2,4): 4
Another[y/n]? y
Enter first coordinates(x1,y1): 1 2
Enter second coordinates (x2,y2): 3 3
The slope of a line going through the point (1,2) and the point (3,3) is :0
The y-intercept of the point (1,2): 2
Another[y/n]? n
```