Su2019. LAB 1 (May1-7)
Introduction to C, Function Declaration vs. Definition, Basic I/O (scanf/printf, getchar/putchar, input/output redirection)

Due: May 7 (T), 11:59 am (noon).

0 problem A0 printf in Java
Specification
While this is a course on C programming, let's start our very first lab with a JAVA question. Download the partially implemented Java program `lab1A0.java` . This program reads a name and then an integer from the user, and then outputs the double and triple value of the entered integer.

Implementation
First, complete the `println` statement, so that it produces the output as shown below.
Next, complete the `printf` statement so that it produces the same output, as shown below. Try to do it without using string concatenations.

Objectives
The purpose of this exercise is for you to:
- learn that `printf` also exists in JAVA (actually JAVA borrowed the idea from C)
- learn the syntax of `printf` in JAVA and C
- observe that sometimes using `printf` can be easier than `println`.

Sample Inputs/Outputs:
```
red 300 % javac lab1A0.java
red 301 % java lab1A0
Please enter the name: Judy
Please enter the number: 22
Hi Judy, double and triple of input 22 is 44 and 66 respectively
Hi Judy, double and triple of input 22 is 44 and 66 respectively
red 302 % java lab1A0
Please enter the name: Joe
Please enter the number: 100
Hi Joe, double and triple of input 100 is 200 and 300 respectively
Hi Joe, double and triple of input 100 is 200 and 300 respectively
```

Submission:
Under the current directory where your file is located, issue the command
```
submit 2031 lab1 lab1A0.java
```

1 Problem A. scanf, printf in C
1.1 Specification
Write an ANSI-C program that reads input from the Standard Input, and then outputs the reformatted versions of the input.

1.2 Implementation
You can download the file `hello.c` and use it as a template.

- name your program `lab1A.c`
- use `scanf` to read input (from Standard Input), which are in the form of
  `Month Day Year` (i.e., three integers separated by white spaces).
- use `printf` to generate output in the format of `Year/month/day` and `Year-month-day`
- display the following prompt (leave a white space after the colon)
  `Enter month, day and year separated by spaces:`
  display output as follows
  `The input 'x xx xxxx' is reformatted as x/xx/xxx and x-xx-xxx`

- Note: you should do the reformatting only within `printf`. In particular, you should use at most three variables, and feed them into `printf` judiciously.

1.3 Sample Inputs/Outputs:
```
red 306 % gcc lab1A.c
red 307 % a.out
Enter month, day and year separated by spaces: 3 20 2019
The input '3 20 2019 ' is reformatted as 2019/3/20 and 2019-3-20
red 308 %
```

Submit your program using **submit 2031 lab1 lab1A.c**

2 Problem B. Functions in C

Download the program `lab1B.c`, compile it using **gcc lab1B.c**
Observe that the compilation process fails (why?), and `a.out` is not generated.

Modify the program to make it compile. Note that you should not modify or move the existing code. That is, **do not modify the code of main() and sum(), and also do not move the functions.**
Instead, add something to make the program compile.
Run the program to see the output.

Submit your program using
                submit 2031 lab1 lab1B.c

3 Problem C. Functions, scanf, printf, floats
3.1 Specification
Improve program `lab1B`, so that it can read two float numbers for the Standard Input, separated by two pound (#) signs, and then output the sum of the two float numbers.

3.2 Implementation
- name your program `lab1C.c`
- use `scanf` to read inputs (from Standard Input), which are in the form of
  `float1##float2` (i.e., two float numbers separated by two pound signs).
- use `printf` to generate output

3.3 Sample Inputs/Outputs: (note that by default `printf` displays six digits after decimal points of a floating point number.)

```
red 338 % gcc lab1C.c  -o lab1C
red 339 % lab1C
Enter two float numbers separated by ##: 2.35##5.64
2.350000 + 5.640000 = 7.990000
red 340 %
```

Submit your program using

submit 2031 lab1 lab1C.c

4 Problem D. Simple loops
4.1 Specification

Extend program `lab1C`, in such a way that it first prompts the user to enter an integer number, which indicates how many times the user wants to interact with the program. Then the program interacts with the user accordingly.

4.2 Implementation

- name your program `lab1D.c`
- use a loop (`for` or `while`) to interact (i.e., read input and generate output) n times, where n is entered by the user.

4.3 Sample Inputs/Outputs: (ONE blank line between each interaction/iteration):

```
red 338 % gcc lab1D.c  -o lab1D
red 339 % lab1D
Enter the number of interactions: 3

Enter two float numbers separated by ##: 2.35##5.64
2.350000 + 5.640000 = 7.990000

Enter two float numbers separated by ##: 1.1##2.2
1.100000 + 2.200000 = 3.300000

Enter two float numbers separated by ##: 2.5##6
2.500000 + 6.000000 = 8.500000
red 340 %
```

Submit your program using

submit 2031 lab1 lab1D.c

5. Problem E. getchar, putchar, input/output redirection

5.1 Specification

Download the provided program `copy.c`, which uses `getchar` and `putchar` to read user input from Standard input (keyboard) and write the input to Standard output (screen). (This program is also in the textbook and the lecture slides.)

Play with the program and make sure you understand the program. In particular, observe a few things about `getchar` and `putchar`:

- `getchar()` reads characters from Standard input (stdin), which by default is the keyboard. But the input can be redirected from an input file using `< filename.` In the latter case `getchar()` will read from the input file instead.
- `getchar()` returns **EOF** (which is a special negative number defined in C) when the "*end of file*" is reached.
  - If the program reads from a text file (redirected using <), then the end of the text file is "*end of file*";
  - If the program reads from Standard in (i.e., keyboard) , then in Unix, `ctrl D` indicates "*end of file*" (in Windows, it is `ctrl Z`)
- Although the program calls `putchar` or `printf` in every iteration of the loop (after every `getchar`), you will notice that the output is displayed only after a whole line is read in. This is correct and is related to the buffer used by the system. Specifically, instead of executing `putchar` or `printf` for every `getchar`, the system buffer stores the chars that are read in, and executes `putchar` only after a new line character `'\n'` is read in.
- Instead of a `char`, function `getchar` returns an `int`, and `putchar` takes as argument an `int`. This will be explained in class.

5.2 Sample Inputs/Outputs (from Standard input - keyboard):
```
red 308 % gcc copy.c
red 309 % a.out
hello
hello
how are you!
how are you!
I am good and thanks !
I am good and thanks !
^D (press Ctrl and D)
red 310 %
```

5.3 Sample Inputs/Outputs (use redirected input/output files):
You can always redirect the Standard in (keyboard) from an input file using `<`
You can always redirect the Standard out (screen) to an output file using `>`

Download file `greetings.txt`, whose content is
```
hello
how are you
I am good
```

```
red 311 % a.out < greetings.txt
hello
how are you
I am good
red 312 % a.out  < greetings.txt > output.txt
```

Nothing will be generated on the screen, because output is redirected using > . Now a new file `output.txt` should be generated (in the current directory). Use command `ls` or `ls -l` to confirm this. Then use command `cat` or `more` to view the content of `output.txt`

(If you don't know what is happening here, please review the CSE1020 Guided Tour posted on the course website. Also in the terminal, issue `cal` or `date` to view the output, and then issue `cal > tem.txt` or `date > tem.txt` to see how the output is redirected to a file) .

```
red 313 % ls -l
red 314 % cat output.txt
hello
how are you
I am good
```

You don't need to submit anything for this question but doing this gets you prepared for the next questions.

6. Problem F getchar, character comparison
6.1 Specification
In class we showed a simple program that uses `getchar` to read input character by character, counting the number of characters from the standard input (keyboard or directed).
Note that this program counts new line characters.
Modify the program so that it counts only visible characters, i.e., new line character is not counted.

6.2 Implementation
- Hint: you might need to compare every character `getchar` reads in against the new line character. In Java or C, how to compare two characters?
- Name your program **lab1F.c**

6.3 Sample Inputs/Outputs
```
red 308 % gcc lab1F.c -o lab1F
red 309 % lab1F
hello
how are you
I am good
^D (press Ctrl and D)
# of chars:  25
red 310 %
red 311 % lab1F < greetings.txt
# of chars:  25
red 312 % lab1F  < greetings.txt > outputF.txt
```

If your program runs correctly, a new file `outputF.txt` should be generated (in the current directory). Use command `ls` or `ls -l` to confirm this. Then use command `cat` or `more` to view the content of it.
```
red 313 % ls -l
red 314 % cat outputF.txt
# of chars:  25
```
Submit your program using
```
                submit 2031 lab1 lab1F.c
```

7. Problem G   getchar, character comparison

Modify the program **lab1F.c** so that it also counts the number of lines in the input.
Name your program **lab1G.c**

```
red 308 % gcc lab1G.c –o lab1G
red 309 % lab1G
hello
how are you
I am good
^D (press Ctrl and D)
# of chars:  25
# of lines:  3
red 310 % lab1G < greetings.txt
# of chars:  25
# of lines:  3
red 311 % lab1G < greetings.txt > outputG.txt
red 312 % cat outputG.txt
# of chars:  25
# of lines:  3
```

Submit your program and the output file using
                    submit 2031 lab1 lab1G.c

In summary, for this lab you should submit the following files:

lab1A0.java   lab1A.c   lab1B.c   lab1C.c   lab1D.c   lab1F.c   lab1G.c

You may want to issue **submit –l 2031 lab1** to get a list of files that you have submitted.

Common Notes

All submitted files should contain the following header:
```
/*************************************
* EECS2031 – Lab1 *
* Author: Last name, first name *
* Email: Your email address *
* eecs_username: Your eecs login user name *
* York Student #: Your student number
*************************************/
```

In addition, all programs should follow the following guidelines:
- Include the stdio.h library in the header of your .c files.
- **Assume that all inputs are valid (no error checking is required, unless asked to do so).**