Lab 4-1: Ifs and Loops...

Due date: Monday, January 31, 11:59pm.

Lab Assignment

Assignment Preparation

Lab type. This is an individual lab.

Collaboration. You can talk to other students in the lab, but no code sharing is allowed and no discussions outside the lab period.

Purpose. This is a short lab. It allows you to practice the use of loops.

Programming Style. All submitted C programs must adhere to the programming style described in detail at

http://users.csc.calpoly.edu/∼cstaley/General/CStyle.htm

When graded, the programs will be checked for style. Any stylistic violations are subject to a 10% penalty. Significant stylistic violations, epsecially those that make grading harder, may yield stricter penalties. Also note the the Lab 2 requirement for the content of the header comment in each file you submit applies **to each assignment** (lab, programming assignment, homework) in this course.

Testing and Submissions. Any submission that does not compile using the

```
gcc -ansi -Wall -Werror -lm
```

compiler settings will receive an automatic score of 0.

For each program you have to write, you will be provided with instructor's executable and with either a battery of tests if tests are needed. The programs you submit must either produce correct output (if they take no input) or pass all tests made available to you. You can check whether or not a program produces correct output by running the instructor's executable on the test case, then running yours, and comparing the outputs.

Program Outputs must co-incide. Any deviation in the output is subject to penalties. PLEASE, USE BINARY EXECUTABLES PROVIDED BY THE INSTRUCTOR!. The exception is made in case of floating point computations leading to differences in the last few decimal digits.

The Task

Temperature Conversion Table

Modify your `converter.c` program from the previous labs to print out a temperature conversion table. The modifications are outlined below.

TC1. Your program will take no inputs.

TC2. When started, your program will output a table of converted temperatures. Each line of output shall contain three parts:

1. Temperature in degrees Celsius;
2. Temperature in degrees Fahrenheit;
3. State of water at this temperature (`"ice"`, `"liquid"` or `"vapor"`).

TC3. The temperature conversion table starts at - 40 degrees Celsius.

TC4. Between -40 degrees Celsius and - 5 degrees Celsius, your program shall convert all integer temperature values divisible by five (i.e., -40, -35, -30 and so on).

TC5. Between -5 and +10 degrees Celsius, your program must convert every integer temperature value (-4, -3,. . . , +9, +10).

TC6. Between +10 and +35, your program must convert every integer temperature value divisible by 5.

TC6. Your program shall convert +36.

TC7. Between 36.4 degrees Celsius and 39.0 degrees Celsius, your program shall convert all temperature values different by 0.2 degrees (36.4, 36.6, 36.8, . . . , 38.8, 39.0).

TC8. Between 40 and 150 degrees Celsius, your program shall convert all temperature values divisible by 10 (50, 60,. . . , 150).

TC9. Your program shall end with printing two empty lines.

TC10. Name your program `ttable.c`.

Output. Sample lines of output are shown below:

```
-1.00 30.20 ice
0.00 32.00 ice
1.00 33.80 liquid
```

Horizontal Bars

Lab 4-2 will involve you writing programs that produce a variety of graphical images using PP file format. To prepare you for this task, the next three programs introduce you to the idea of printing two-dimensional images. All three programs will use ASCII graphics. The first program shall print a series of horizontal bars of different "colors" (characters) as shown below.

HB0. Name the program `horizontal.c`.

HB1. The program shall take no inputs.

HB2. The program shall output three horizontal "bars" rendered as ASCII characters. All three bars shall have the same dimensions: 40 characters long and five (5) characters tall.

HB3. The first bar shall be drawn using the `'*'` character.

HB4. The second bar shall be drawn using the `'-'` character.

HB5. The third bar shall be drawn using the `'#'` character.

The output of your program shall look as follows:

```
****************************************
****************************************
****************************************
****************************************
****************************************
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
########################################
########################################
########################################
########################################
########################################
```

The following constraints apply to your code:

HBC1. You must print every character in a single `printf()` statement. Simply writing

```
printf("****************************************\n");
```

is NOT ALLOWED.

HBC2. In addition to `#define`-ing all magic numbers in your program, you shall also `#define` all magic characters.

Vertical Bars

The second program in the series asks you to produce three vertical bars as described and shown in the output below.

VB0. Name the program `vertical.c`.

VB1. The program shall take no inputs.

VB2. The program shall output three vertical "bars" rendered as ASCII characters. All three bars shall have the same dimensions: 15 characters long and 15 characters tall.

VB3. The first bar shall be drawn using the `'*'` character.

VB4. The second bar shall be drawn using the `'-'` character.

VB5. The third bar shall be drawn using the `'#'` character.

The output of your program shall look as follows:

```
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
***************---------------###############
```

The following constraints apply to your code:

VBC1. You must print every character in a single `printf()` statement. Simply writing

```
printf("***************---------------###############\n");
```

is NOT ALLOWED.

VBC2. In addition to `#define`-ing all magic numbers in your program, you shall also `#define` all magic characters.

Frames

The third program will print a "frame"-like image shown below.

```
##############################
#----------------------------#
#----------------------------#
#----------------------------#
#----------------------------#
#----------------------------#
#----------------------------#
#----------------------------#
#----------------------------#
#----------------------------#
#----------------------------#
#----------------------------#
#----------------------------#
#----------------------------#
##############################
```

FR0. Name your program `frame.c`.

FR1. The program will have no inputs.

FR2. The program prints out the "frame" shape shown above. The shape consists of 15 rows and 30 characters in each row.

FR3. The outter part of the fram e is formed by the `'#'` character. The outter part is one character thick.

FR4. The inner part of the frame is formed by the `'-'` character.

The constraints are as before:

FRC1. You must print every character in a single `printf()` statement. Simply writing

```
printf("#----------------------------#\n");
```

is NOT ALLOWED.

FRC2. In addition to `#define`-ing all magic numbers in your program, you shall also `#define` all magic characters.

Submission.

Files to submit. You shall submit **four** files:

```
ttable.c,
horizontal.c,
vertical.c,
frame.c
```

Files can be submitted one-by-one, or all-at-once.

Submission procedure. Submission command:

Section 01

```
> handin dekhtyar-grader lab04-1-01  <your files go here>
```

Section 09

```
> handin dekhtyar-grader lab04-1-09  <your files go here>
```

Grading

Each program is worth 25% of the lab grade. Any submitted program that does not compile earns 0 points.

All programs will be checked for style conformance. Any style violation will be noted. Please note, that we will also check for conformance to the additional constraints specified for some of the programs. Violations will be penalized by a 10% penalty.

Testing

Instructor's executable files are provided. Note, that each program produces a single output, so no test files are necessary. The outputs of the instructor's executable and of your program must coincide.