ENEE 140 Lab 4

Lab instructions

This handout includes instructions for the recitation sessions on Wednesday and Friday. **Follow these instructions** to review the usage of the `printf` function, then **submit the homework** as indicated below.

1 `printf` review

The format string "..." of `printf()` statement has strict "matching" requirements: (1) the number of format indicator %'s should match the number of arguments (variable names); (2) the first % should match the first argument, the second % should match the second argument, and so on; (3) the data type of an argument should match the corresponding type indicated by the parameter after the format indicator %. Think about what would happen if there is a mismatch.

Read `printf2.c`, available in GRACE class public directory under `public/labs/week04/`, to predict the output. Then compile the code, run it and analyze the output.

When the field width of the `printf()` statement is specified, does the - sign counts as one space when the value is negative? Write a simple program to verify your answer.

Homework

Due: February 20 at 11:59 pm.

Write three programs and create one text file by following the instructions below. Submit them using the following commands:

```
submit 2016 spring enee 140 AAAA 4 count_char.c
submit 2016 spring enee 140 AAAA 4 data_entry.c
submit 2016 spring enee 140 AAAA 4 printf_questions.txt
```

Note: you must replace `AAA` with your own section number (0101, 0102, etc.)

1 Character input

The `getchar()` function allows you to read from the input one character at a time. Write a program that reads its input character-by-character, until it encounters the end-of-file (EOF) maker, and then prints the number of `'a'` characters it has read.

Hint: you can use the program on page 19 of the textbook as a starting point.

2 Data entry

Write a complete program called `data_entry.c` that prompts a student for his/her section, GPA, course load and credits, and after that prints this information on the console. To do this, implement five C functions that read and return each of these attributes and invoke these functions from the `main()` function.

1. Implement a C function with the following prototype:

   ```c
   int get_section();
   ```

   This function should start by printing the following message: "`Please enter your section number (1, 2, or 3):`", then it should wait for the user to type an integer.

   The function should return this integer.

2. Implement a C function with the following prototype:

   ```c
   float get_gpa();
   ```

   This function should start by printing the following message: "`What is your current GPA?`", then it should wait for the user to type a floating point number.

   The function should return this number.

3. Implement a C function with the following prototype:

   ```c
   int get_course_load();
   ```

   This function should start by printing the following message: "`How many courses are you taking this semester?`", then it should wait for the user to type an integer.

   The function should return this integer.

4. Implement a C function with the following prototype:

   ```
   int get_credits();
   ```

   This function should start by printing the following message: "`How many credits are you taking in total?`", then it should wait for the user to type an integer.

   The function should return this integer.

5. In the `main()` function, invoke these five functions to read all the information from the user, then print the information in the following tabular format (note that there is one tab after `Section:` and all the other attribute names, and the section number should be printed like this `0102` if the input is 2):

   ```
   Section:        0103
   Current GPA:    3.87
   Courses:        4
   Credits:        11
   ```

Hint. You can start from the template available in GRACE class public directory at `public/labs/week04/data_entry.c`.

3 `printf` questions

1. What is be the output for each of the following `printf()` statements? Write down your answers in `printf_questions.txt`.

   ```
   printf("%12d \n%12f \n %012d\n",23,3.1416,-23);
   printf("%012d \n%*d \n%0*d \n", -23, 12,23, 12,23);
   printf("%5.2f \n%.2f \n", 3.14159, 3.14159);
   printf("%*.*f \n", 5, 3, 3.14159);
   printf("%5.2d \n%10.5s\n", 1, "Hello, World!");
   printf("%-15ftest \n%+ftest \n% ftest\n", 3.1416, 3.1416, 3.1416);
   ```

   Hint. If you don't know the answers, write a complete C program and include these statements to find out.

2. Use one `printf()` statement to print out the format indicator %. Try to think of different ways to do it. For example, if you figure out 3 different ways to do it, your program should print out %%%.

3. When a real number has a higher precision (3.14159 for example) than the space specified in the printf statement (4 spaces for example), what will be printed out? The options we can think are rounded up (3.15), rounded down (3.14), rounded to the nearest (3.14), or truncated (3.14). Use 3.14159 as an example, if you see output 3.14, you know that rounded up is not the correct option. But you cannot tell whether it is rounded down, to the nearest, or truncated. Write multiple `printf()` statements with different values to show what is the correct option on our system.

   Hint. For each incorrect option, find a value that it will give wrong answer and print out a message. For example, when you see an output of 3.14 on value 3.14159, you know rounded up is wrong, so you can print out: not rounded up because $3.14159 = 3.14$

Reading assignment

K&R Chapters 2.5, 2.7, 2.8, 2.10, B2, B11

Weekly challenge

Write a function that reads integer numbers character-by-character and that returns the corresponding int value (read until end-of-line and ignore all characters that are not digits). Use this function to read two int values, then print the quotient and remainder that result from dividing the first value by the second value. Also print the result of dividing the two numbers using the rules of floating-point division.

You can use the following template (also available in the GLUE class public directory, under `public/challenges/week04/`):

```c
/*
 * read_divide_ints.c  --
 *
 *   Write a function that reads integer numbers character-by-character
 *   and that returns the corresponding int value (read until end-of-line
 *   and ignore all characters that are not digits). Use this function to
 *   read two int values, then print the quotient and remainder that result
 *   from dividing the first value by the second value. Also print the
 *   result of dividing the two numbers using the rules of floating-point
 *   division.
 */


#include <stdio.h>

// Function prototype
int read_int();

int
main()
{
    int dividend=0, divisor=0, quotient=0, remainder=0;
    float flt_div = 0.0;

    printf("Enter the dividend: ");
    // Read the dividend by invoking the read_int() function

    printf("Enter the divisor: ");
    // Read the divisor by invoking the read_int() function

    // Perform the arithmetic operations

    // Print the results
    printf("The quotient is %d\n", quotient);
    printf("The remainder is %d\n", remainder);
    printf("The result of floating-point division is %.2f", flt_div);
    return 0;
}

// Function implementation
int
read_int()
{
    int num = 0;
    int c;

    // Read integer number character-by-character
    return num;
}
```

The weekly challenge will not be graded. However, if you manage to solve it, you may submit it for extra credit. The deadline for submitting your solution to the weekly challenge is **Monday at 11:59 pm**. To be eligible for extra credit, you must implement correctly **all but two** of the weekly challenges. You can submit your program from a GRACE machine using the following command (replace AAAA with your section number):

```
submit 2016 spring enee 140 AAAA 1004 read_divide_ints.c
```