ENEE 140 Lab 5

Lab instructions

This handout includes instructions for the recitation sessions on Wednesday and Friday. **Follow these instructions** to review the usage of the `scanf` function, then **submit the homework** as indicated below. To prepare for the next lecture, complete the **reading assignment** and try to solve the **weekly challenge**.

1 `scanf` review

What happens if the number of variables passed to the `scanf` function, and/or the variable types, do not match those specified in the format string? Write a simple program to verify your answer.

Homework

Due: February 26 at 11:59 pm.

Write two C programs by following the instructions below. Submit them using the following commands:

```
submit 2016 spring enee 140 AAAA 5 char_class.c
submit 2016 spring enee 140 AAAA 5 sum_of_squares.c
```

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

1 Character classes

Write a complete C program called `char_class.c` that reads in one character from the keyboard and detects whether it is (i) a lower case letter, (ii) an upper case letter, (iii) a digit (0–9), or (iv) some other character. You will need multiple if statements for this. For example, here is the output on some inputs:

```
Please enter one character: D
D is an upper case letter.
Please enter one character: 8
8 is a digit.
Please enter one character: %
% is not a letter or digit.
Please enter one character: j
j is a lower case letter.
```

Hint. Remember that in ASCII code, 0–9 have consecutive codes (or numerical values); a–z also have consecutive codes, and so do A–Z.

2 Sum of squares

Write a C program called `sum_of_squares.c` that asks user to enter a positive integer n, and prints out the result of $1^2 + 2^2 + ... + n^2$. For example, for the input $n = 3$, the output should be `1 + 4 + 9 = 14`.

Reading assignment

K&R Chapters 2.2, 2.9, 3.3, 6.1, B5, B6

Weekly challenge

Write a program that prompts the user for a positive number, reads this number into an `unsigned` variable, then prints the binary representation of this number. On the GRACE computers, unsigned integers are represented using 32 bits (b_0, b_1, ... b_{31}). Each bit can be either 0 or 1, and the value of the unsigned integer is computed using the following formula:

$$U = \sum_{i=0}^{31} = b_i \cdot 2^i$$

For example, the number 5 has $b_0 = 1$, $b_1 = 0$, $b_2 = 1$, and the rest of the bits are 0:

$$1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 = 1 + 4 = 5$$

You program should prompt the user for a positive number, then it should print the values of each bit in the binary representation, starting with b_0. For example, the output of the program could look like this:

```
Enter a positive integer: 5
Binary representation of 5:
Bit  0: 1
Bit  1: 0
Bit  2: 1
Bit  3: 0
Bit  4: 0
Bit  5: 0
Bit  6: 0
Bit  7: 0
Bit  8: 0
Bit  9: 0
Bit 10: 0
Bit 11: 0
Bit 12: 0
Bit 13: 0
Bit 14: 0
Bit 15: 0
Bit 16: 0
Bit 17: 0
Bit 18: 0
Bit 19: 0
Bit 20: 0
Bit 21: 0
Bit 22: 0
```

```
Bit 23: 0
Bit 24: 0
Bit 25: 0
Bit 26: 0
Bit 27: 0
Bit 28: 0
Bit 29: 0
Bit 30: 0
Bit 31: 0
```

You can use the following template (also available in the GRACE class public directory, under `public/challenges/week05`):

```c
/*
 * dec2bin.c
 *
 *  Prompt the user for a positive number, then print the binary
 *  representation of that number starting from the least-significant bit.
 */

#include <stdio.h>


int
main()
{
        return 0;
}
```

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 1005 dec2bin.c
```