ECE15: Homework #3

Notes:

- In the sample runs below, computer output is shown in black, and user inputs in red.
- To be accepted by the automated homework checker, your output should match the output shown exactly, including spelling, capitalization, punctuation, and spacing.
- To clarify the number of spaces, we show them as ␣. In your program, use regular spaces.
- Comment your programs clearly. We may deduct points for under-commented programs.
- You may assume that the user will always enter the input correctly.

Problem 1

Write a program `weekday.c` that prompts the user for a date (entered as `month/day`) this year, and outputs the day of the week the date falls on. Use the fact that this year (2011), January 1st was a Saturday and that January, March, May, July, August, October, and December have 31 days, April, June, September, and November have 30 days, and February this year had 28 days.

Note: Do not specify in the program the first day of each month (e.g., that February 1st will fall on a Tuesday, etc.). Instead, use one switch statement to calculate the number of days till the date entered, and another switch statement to print the day of the week.

Here is a sample run:

```
(~)$ a.out
Enter␣today's␣date:␣03/21
Happy␣Monday!
(~)$ a.out
Enter␣today's␣date:␣5/14
Happy␣Saturday!
(~)$ a.out
Enter␣today's␣date:␣11/24
Happy␣Thursday!
(~)$
```

Problem 2

Write a program `sequence.c` that repeatedly prompts the user for an integer until the entered integer is the same as the one entered the first time. The program then outputs the number of entered integers that are:

- even;
- divisible by three or five;
- strictly greater than the integer preceding them.

For example, in the following sample run, there are three even integers (988, 11000, 458), two integers divisible by three or five (11000, 15), and four integers that are greater than the integer preceding them ($988 > 7$, $11000 > 988$, $458 > 15$, $7 > -209$):

```
(~)$ a.out
Enter an integer: 7
Enter an integer: 988
Enter an integer: 11000
Enter an integer: 15
Enter an integer: 458
Enter an integer: -209
Enter an integer: 7
Even: 3
Divisible by three or five: 2
Greater than the preceding integer: 4
(~)$
```

In the next sample run there are four even integers (0, 60, 78, 0), six integers divisible by three or five (0, 60, 129, 78, 39, 0), and three integers that are greater than the integer preceding them ($60 > 0$, $129 > 43$, $487 > 17$):

```
(~)$ a.out
Enter an integer: 0
Enter an integer: 60
Enter an integer: 43
Enter an integer: 129
Enter an integer: 78
Enter an integer: 39
Enter an integer: 17
Enter an integer: 487
Enter an integer: 0
Even: 4
Divisible by three or five: 6
Greater than the preceding integer: 3
(~)$
```

Final sample run:

```
(~)$ a.out
Enter an integer: 30
Enter an integer: 30
Even: 2
Divisible by three or five: 2
Greater than the preceding integer: 0
(~)$
```

Problem 3

Write a program `wedge2.c` that prompts the user for a positive integer and prints a wedge of $*$'s whose longest line is of length equal to the entered number. Here is a sample run:

```
(~)$ a.out
Enter an integer: 1
*
(~)$ a.out
Enter an integer: 2
*
**
*
(~)$ a.out
Enter an integer: 3
*
**
***
**
*
(~)$
```

Problem 4

(Note: the material we have already learned is sufficient for writing this program. However, on Tuesday we'll cover an example similar to this problem, so if you find this problem difficult, you can wait till then.)

Write a program `reverse.c` that prompts the user for a nonnegative integer and outputs a sequence consisting of the entered integer, as many $*$'s as the number of digits of the entered integer, and the entered integer with its digits reversed. Here is a sample run:

```
(~)$ a.out
Enter an integer: 0
0*0
(~)$ a.out
Enter an integer: 123
123***321
(~)$ a.out
Enter an integer: 12000
12000*****00021
(~)$ a.out
Enter an integer: 021
21**12
(~)$ a.out
Enter an integer: 3024
3024****4203
(~)$
```