CSI 333 – Fall 2011

Programming Assignment II

Administrative Information

- Individual assignment.

- **Deadline:** 11 PM, Monday, Oct. 3, 2011.
  Cutoff Point: 11 PM, Wednesday, Oct. 5, 2011.

- The C source files for the two parts must be named `p2a.c` and `p2b.c` respectively.

- The two source files must be submitted together using the `turnin-csi333` command.

- README file (on `itsunix.albany.edu`) by 10 PM on Saturday, September 24, 2011.

        ~csi333/public/prog2/prog2.README

Important Remarks:

- When you use `turnin-csi333` to submit your files, you must submit both p2a.c and p2b.c at the same time. Thus, the Unix command to be used for turning in your files is the following:

```
/usr/local/bin/turnin-csi333  -c csi333  p2a.c  p2b.c
```

- Programs that don't compile or don't generate the executable won't receive any credit.

- Your program must compile and work correctly on `itsunix.albany.edu`.

Administrative Information (continued)

Lateness Policy:

- No penalty if the program is submitted **by 11 PM on Monday, Oct. 3, 2011**.

- Lateness penalty: 10 points per day.

- Program **won't** be accepted **after 11 PM on Wednesday, Oct. 5, 2011**.

- If you submit both a regular version and a late version, only the late version will be graded.

Other Notes:

- **Weightage:** 7%

- **Total Points:** 100
    - **Part (a):** 40 points (35 points for correctness and 5 points for structure/documentation).
    - **Part (b):** 60 points (55 points for correctness and 5 points for structure/documentation).

Description of Part (a)

Goal: Given two decimal integers d and r, convert d into its representation in radix r and print the resulting representation.

Notes:

- The first integer (d) will be a non-negative decimal integer.

- The second integer (r) will be one of 2, 3, 4, ..., 15, 16.

- Use letters A, B, C, D, E and F to represent 10, 11, 12, 13, 14 and 15 respectively (as in hex).

Examples of Program Execution

Note: The following examples assume that the executable version of the program for Part (a) is in the file `parta.out`.

Example 1:

```
unix2> parta.out
Enter two integers: 138    16
Answer = 8A
unix2>
```

Example 2:

```
unix2> parta.out
Enter two integers: 284    13
Answer = 18B
unix2>
```

Example 3:

```
unix2> parta.out
Enter two integers: 68    2
Answer = 1000100
unix2>
```

Program Outline for Part (a):

1. Prompt the user to type two decimal integers.
2. Read the two integers.
3. Convert the first integer into its representation in the radix specified by the second integer.
4. Print the representation and **stop**.

Additional Notes Regarding Part (a)

- Use the division method (discussed in Lecture 1) to generate the digits of the required representation.

- Use a `char` array to store each digit generated by the division method as an appropriate character. (This array should be printed out at the end.)

Description of Part (b)

Goal: <u>Strict-left-to-right</u> evaluation of an arithmetic expression.

Assumptions Regarding the Input Expression:

- Contains only single digit integer constants, operators ('+', '−', '∗' and '/', where '/' denotes integer division) and spaces.

- Begins with an integer constant (<u>without</u> any preceding sign).

- Integer constants and operators <u>alternate</u>.

- Each input expression is terminated by the newline ('\n') character.

<u>**Note:**</u> An expression containing just a single digit is valid.

Examples of expressions and their values:

9	Value $= 9$
7/3	Value $= 2$
$7 + 4 \ * \ 5$	Value $= 55$
$7 + 4 * 5/8 - 9$	Value $= -3$

Examples of program execution: Assume that the executable version of the program for Part (b) is in the file partb.out.

Example 1:

```
unix2> partb.out
Enter expression: 9*2 - 5/3 -9
Value = -5
unix2>
```

Examples of program execution (continued):

Example 2:

```
unix2> partb.out
Enter expression: 7 +4 * 5 / 8 -9
Value = -3
unix2>
```

Program Outline for Part (b):

1. Prompt the user for an expression.

2. Read the expression character by character and carry out a strict-left-to-right evaluation of the expression.

3. Print the value of the result obtained in Step 2 and **stop**.

Additional Notes Concerning Both Parts

- For both parts, your program should read from `stdin` and write to `stdout`.

- No error checks are needed either in Part (a) or in Part (b).

- For both parts, after <u>each</u> call to the function `printf`, include the following C statement:

      fflush(stdout);

Example 1:

```
printf("Enter two integers: "); fflush(stdout);
```

Example 2:

```
printf("Value = %d\n", result); fflush(stdout);
```

Program Grading

- Programs will be graded using a script written by the TAs.

- The script will compile your source program, generate the executable version and run the executable on new test data.

- The TAs will grade the version that you submit; once the submission is closed, you **won't** be allowed to make any changes to your program.