CSI 333 – Programming at the Hardware-Software Interface – Fall 2011

Programming Assignment II

Date given: Sep. 20, 2011 **Due date:** Oct. 3, 2011
Weightage: 7%

The deadline for this assignment is **11 PM, Monday, October 3, 2011**. With lateness penalty, the program will be accepted until **11 PM, Wednesday, October 5, 2011**. The assignment will *not* be accepted after that deadline.

Important Notes: There are two parts in this assignment. The C source program for Part (a) of this assignment should be in a file called `p2a.c`. Similarly, the C source program for Part (b) of this assignment should be in a file called `p2b.c`. The two source files must be submitted *together* using the `turnin-csi333` command by the deadline specified above. Additional information about the `turnin-csi333` command will be included in the `README` file for this assignment.

You must follow the programming and documentation guidelines indicated in the handout on "Course Policies".

The total grade for the assignment is 100 points, with 40 points for Part (a) and 60 points for Part (b).

Description of Part (a):

You are required to write a C program that accepts two decimal integers, say d and r. You may assume that the first decimal integer d is nonnegative and that the second decimal integer r which represents the radix, will be one of 2, 3, 4, ..., 15, 16. The program should convert the first decimal integer d into its representation in radix r and print the result.

Examples:

(i) Suppose the first decimal integer is 138 and the second decimal integer is 16. In this case, the output produced by your program should be `8A`, which is the hexadecimal (radix 16) representation of the decimal integer 138.

(ii) Suppose the first decimal integer is 284 and the second decimal integer is 13. In this case, the output produced by your program should be `18B`, which is the radix 13 representation of the decimal integer 284. (In base 13, the digits used are 0, 1, 2, ..., 9, A, B and C, where A, B and C represent 10, 11 and 12 respectively.)

Your program should be written so that it handles just one pair of integers. Thus, the outline for your program is as follows. (Note that no error checks are needed.)

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.

Note that your program should read the two integers from `stdin` and write the answer to `stdout`. You may assume that when prompted, the user will type two integers separated by one or more spaces.

Note: For any radix $r \geq 2$, the digits to be used are 0, 1, ..., $r - 1$. Use the letters A, B, C, D, E and F to represent 10, 11, 12, 13, 14 and 15 respectively, as done in the hexadecimal system. Thus, representations in radix 11 can use the digits 0, 1, ..., 9, A; representations in radix 12 can use 0, 1, ..., 9, A, B, and so on.

Description of Part (b):

You are required to write a C program to carry out a **strict-left-to-right** evaluation of an arithmetic expression consisting of integer constants and the operators $+$, $-$, $*$, and /. Here, the operator / denotes integer division; that is, the remainder is discarded. In a strict-left-to-right evaluation, there is no notion of precedence. For example, the value of the expression $6+4*3$ when evaluated in a strict-left-to-right fashion is 30. (Under usual precedence rules, where multiplication has higher precedence than addition, the value of the above expression would be 18.) Similarly, the value of the expression $6 + 4 * 3/7 - 9$ when evaluated in a strict-left-to-right fashion is -5.

Here is some additional information about the input.

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

2. There may be zero or more spaces between successive non-blank characters of an expression.

3. You may assume that the given expression is valid; that is, it satisfies all of the following conditions.

   (a) The expression consists only of integer constants, the operators $+$, $-$, $*$, / and spaces. In particular, the expression *won't* contain parentheses.

   (b) Each integer constant in the expression consists of *only one* decimal digit.

   (c) The expression begins with an integer constant, *without* any preceding sign.

   (d) In the expression, integer constants and operators alternate.

   Note that an expression consisting of a single digit integer constant, without any operators, is a valid expression.

The outline of your C program is as follows.

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**.

Thus, each time your program for Part (b) is executed, it should handle just one expression. As in part (a), bear in mind that your program reads its input from `stdin` and writes its output to `stdout`. As in Part (a), no error checks are needed.

Information about `README` file: The `README` file for this assignment will be available by 10 PM on Saturday, September 24, 2011. The name of the file will be `prog2.README` and it will be in the directory `~csi333/public/prog2` on `itsunix.albany.edu`. This file will contain information about the `turnin-csi333` command. It may also include information regarding some sample inputs and outputs that you can use to test your programs.