CSI 333 – Programming at the Hardware-Software Interface – Spring 2017  
Project I

The total grade for the assignment is 100 points.

You must follow the programming and documentation guidelines available in the Blackboard module Projects.

# Description:

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:

1. 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.
2. 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.

Your program must 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.

Notes:

* Recall that for any radix *r ≥* 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.
* 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.
* After each call to the function printf, include the following C statement: fflush(stdout);.  
  Example: printf("Enter two integers: "); fflush(stdout);.

# Electronic Submission:

Important Notes: ignoring any of the following rules will result in penalty or even ZERO grade for the project.

* 1. For Project 1 you must turn in the file **named** “**p1.c”**.
  2. At the top of each of your C source file the following information must appear in the form of comments:
  3. your name,
  4. your Unix login id,
  5. the name of your lab instructor and
  6. the day and time of your lab class.
  7. Make sure that your programs compile and produce correct results on the UNIX machines (itsunix.albany.edu) supported by Information Technology Services (ITS) unit of UAlbany. Programs that cause compiler or linker errors on the ITS UNIX machines will NOT receive any credit.
  8. Using the turnin-csi333 program as discussed below is the ONLY acceptable way of submitting programming assignments in this course. You should NOT email the files to the instructor or to the TAs.
  9. Remember that you must submit only your C source files. DON'T turn in unnecessary files (e.g. object files with extension “.o” created by compiling C source files, executable files such as “a.out”, etc.).

To submit your files electronically, you must have the source files on one of the ITS UNIX machines. For this project the file p1.c must be in your working directory and you must be logged on to one of those machines to actually carry out the electronic submission.

To perform submission you should type the following command to the UNIX operating system:

/usr/local/bin/turnin-csi333 -c csi333 -p hw1 p1.c

After you issue the above command, the system responds with:

The sections of csi333 are:

FR\_1025

FR\_1235

FR\_0140

WE\_1235

Enter your section:

Depending upon the day and time of your weekly discussion section, you would type the appropriate section. For example, if your discussion class meets on Wednesdays at 12:35 PM, you would type WE\_1235 followed by the return key. The system will then respond with

Your files have been submitted to csi333, hw1 for grading.

In the above message, "hw1" refers to the name of the project that is currently active. If you submit your program during two days after due date the name of the project will be "hw1-late". Attempts to submit the program after the two day grace period will fail.

Additional information about the turnin program:

* 1. If you use the turnin command above again at a later time (before the deadline), then the files submitted previously would be completely replaced by the newly submitted files.
  2. At any time, you can obtain the names of the files that you have submitted to the current project using the following command:

/usr/local/bin/turnin-csi333 -c csi333 -v

# Some sample data to test your program:

Important Note: Some sample inputs that can be used to test your programs are given below. However, you should remember that when we compile and run your source files, we will use other data. Just because your programs work for the sample inputs given below, you shouldn't assume that they will work for all inputs. Therefore, you should test your programs thoroughly with other input values.

Recall that the input to the Project 1 consists of two integers and the output is the representation of the first integer using the second integer as the radix (base). The following table gives several examples of inputs and the corresponding outputs.

|  |  |  |
| --- | --- | --- |
| **Input Number** | **Radix** | **Output** |
| 138 | 16 | 8A |
| 2279 | 12 | 139B |
| 37373 | 10 | 37373 |
| 741 | 2 | 1011100101 |
| 0 | 11 | 0 |
| 284 | 13 | 18B |

# 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.

Points: 90 points for correctness and 10 points for structure/documentation.

# Example of program execution:

The following examples assume that the executable version of the program is in the file p1.out.

unix2> p1.out

Enter two integers: 138 16

Answer = 8A

unix2>