CS270 Programming Assignment 1
“Radix Conversions”

Due Wednesday, Feb 8 (via checkin by 5:00pm)
Extra credit for early submission and for one additional extension
Homework and programming assignments are to be done individually.

Goals

In this assignment, you will learn C programming and reinforce your understanding of number representation by implementing C functions to convert numbers from any specified radix into decimal and to convert decimal numbers into any specified radix. A “specified radix” in this program can range from 2 to 36.

The Assignment

Make a subdirectory called PA1 for the programming assignment; all files must reside in this subdirectory. We have provided you with a number of files to complete the assignment. Copy the following files into your PA1 directory:

http://www.cs.colostate.edu/~cs270/.Spring12/Assignments/PA1/main.c
http://www.cs.colostate.edu/~cs270/.Spring12/Assignments/PA1/myfunctions.h
http://www.cs.colostate.edu/~cs270/.Spring12/Assignments/PA1/myfunctions.c
http://www.cs.colostate.edu/~cs270/.Spring12/Assignments/PA1/myfunctions_binary.o
http://www.cs.colostate.edu/~cs270/.Spring12/Assignments/PA1/Makefile

You will need to implement the following functions inside “myfunctions.c”. The skeleton structure has already been provided for you.

```
1) int readRadixA (int radixA);
2) char decimalToSymbol(int decimalNumber);
3) void writeRadixB(int decimalNumber, int radixB);
```

Function 1: readRadixA:

Use Horner's Algorithm (as reviewed in lecture) to convert a null-terminated sequence of char into a number (int in C). Read characters from standard input, convert this number into an integer, print this, and return it.

To complete this function, you will need to utilize a provided helper function, symbolToDecimal, which converts a digit from radixN (i.e., a char that is either a digit or an upper case letter) into its decimal equivalent, e.g., the character 0 (ascii 48) is mapped to the integer 0, written as ‘0’ → 0, ‘1’ → 1, …, ‘9’ → 9, ‘A’ → 10, ‘B’ → 11, …, ‘Z’ → 35. This provides 36 unique symbols which we will use to represent numbers with bases ranging from 2 to 36.

Function 2: decimalToSymbol:

Read the symbolToDecimal function that is provided. Understand how it converts, symbol to decimal number using ascii conversion. You can then, write this function to return the decimal equivalent of given symbol.

To complete this function, you will need to convert an integer into its radixN equivalent char, written as, ‘0’ → 0, ‘1’ → 1, …, ‘9’ → 9, ‘A’ → 10, ‘B’ → 11, …, ‘Z’ → 35.

Function 3: writeRadixB:

Use repeated division and modulo operations (as reviewed in lecture) to convert an integer into characters (using a function decimalToSymbol that you have to write). The integer, decimalNumber, represents a non-negative number in radix 10, and must be converted into the radix specified by argument radixB and print the characters in radixB.

To complete this function, you will need to write the helper function, decimalToSymbol, which converts a decimal value into its radixN equivalent char, e.g., 0 → 0, 1 → 1, …, 9 → 9, 10 → A, 11 → B, …, 35 → Z. For those who are unable to write decimalToSymbol, we are providing you with a binary myfunctions_binary.o, so that, you can still verify your function writeRadixB even when you are not able to write the previous function. To use this binary you can use command “make binary”.

Compile and Run:

We provided you with a Makefile to compile the program.
Compile as follows:
```
%> make
```
or (using binary)
```
%> make binary
```

The program is executed as follows:
```
%> pa1
```
The program will then prompt you to enter the *fromRadix* (enter radixA) and *toRadix* (enter radixB).

Try the program with the following test cases:
```
Test 1:
Enter the fromRadix:16
Enter the toRadix:2
Enter radixA Number: ABC

Outputs must be:
2748
101010111100

Test 2:
Enter the fromRadix:5
Enter the toRadix:17
Enter radixA Number: 33342

Output must be:
2347
821
```

Calculate these results by hand and verify if you are getting the correct output. We will run your program using different inputs, so DO NOT HARDCODE values!

For this assignment you must also submit a README file with your name and answers to the following questions. Copy the question into the file and then type in the answer after the question.

Question 1: Are you doing your assignments on the school machines or at home? If at school what is the name of the machine you are using to answer these questions?
Question 2: Type gcc --version on the command line and write down the output. This assumes Linux, if you are running on another operating system, then write down the compiler version.

Submission Instructions
When you are done, your directory should have main.c, myfunctions.c, myfunctions.h, Makefile and a README file. To package the files into a single compressed file, type the following command from inside PA1 directory:

```
%> cd PA1
%> make pack
```

This will create a file called PA1.tar.gz one directory above PA1. All assignments will be submitted directly via checkin, which will be explained and demonstrated in recitation. A sanity check of your PA1.tar.gz will ensure that your submission has all the required files:

```
%> mkdir ~/Temp
%> cp PA1.tar.gz ~/Temp
%> cd ~/Temp
%> tar -zxvf PA1.tar.gz
%> ls PA1
```

Grading Criteria
Points will be awarded as follows: functionality - 75 points (30 for function #1 and 15 for function #2 and 30 for function #3), coding style and comments - 10 points, following assignment directions - 5 points, and supplying answers to the README questions - 10 points. The grading factors we consider for coding style include having clear and concise comments, consistent indentation, and the minimal amount of code to solve the problem. You will also need to ensure that every function (declared in myfunctions.h) has a properly formatted description header. The extra credit problem will be 10 points.

Extra Credit
Use of lower case letters and upper case letters for function symbolToDecimal. The user should be able to input char a or A, b or B, c or C....z or Z and the function symbolToDecimal will produce 10 for both a and A, 11 for both b or B...35 for z or Z.

Late Policy
Our late policy is intended to m=penalize late submissions. Since we were late in posting this assignment, we deserve a penalty (i.e., you deserve “free” extra credit). Our original submission deadline was going to be Monday Feb 6. Now the deadline is Wed Feb 8, but if you submit by the original due date, you can get 20% extra credit. Late assignments will be accepted up to 48 hours past the due date with a deduction of 10% per 24 hours. Late assignments will not be accepted after 48 hours. Please contact the instructor or teaching assistant if you have problems with checkin. So here’s how we will calculate your final score. We grade the assignment out of 100 points (max score 110 with extra credit). This will be multiplied by either 1.2, 1.1, 1, 0.9 or 0.8 depending on when you submit.