CS270 Programming Assignment 1
Number Formats

Goals: To design, code, compile, run, and debug a C program with several functions. The program uses the C language to explore number representation.

The Assignment

Make a subdirectory called PA1 for the programming assignment, all files should reside in this subdirectory. Implement the following functions in a file named pa1.c.

```
// Print out the hex value of an integer, in the C language format.
// Example: input of 19088743 prints "0x01234567"
void printIntegerInHexadecimal(int i);

// Print out the binary value of an integer, divided into groups of four digits.
// Example: input of 19088743 prints "0000 0001 0010 0011 0100 0101 0110 0111"
void printIntegerInBinary(int i);

// Print out the hexadecimal value of a floating point number.
// Example: input of 2.50f prints "0x40200000"
void printFloatInHexadecimal(float f);

// Print out the binary value of a float, including sign, exponent, and fraction.
// Example: input of 2.50f prints "0 10000000 01000000000000000000000"
void printFloatInBinary(float f);
```

Hint: Use printf and format specifiers, some of which are shown below.

Specifier	*Format*
%c	character
%d	signed integer
%u	unsigned integer
%x	hexadecimal
%f	floating point
%e	scientific notation
%s	string

Note that you cannot use the %d, %u, or %x on floating point values, they only work for integer types. Printing a floating point value in hexadecimal or binary can be done via type casting, as discussed in class and the recitation.

Getting Started

We have provided a simple C program to compute factorials to get you started.

Copy the code shown below into a file called pa1.c in your PA1 subdirectory:

```
// File:        pa1.c
// Description: ... fill this in
// Author:      ... fill this in
// Date:        ... fill this in

#include <stdio.h>

// Function:     factorial
```

```
// Description:  computes the factorial of an integer
// Parameters:   value: positive integer
// Return:       factorial: nonzero integer
// Error Checks: none
int factorial(int value)
{
      int result = 1;
      for (int index = value; index > 1; --index) {
                  result *= index;
      }
      return result;
}
```

Copy the code shown below into a file called main.c in your PA1 subdirectory:

```
// File:        main.c
// Description: ... fill this in
// Author:      ... fill this in
// Date:        ... fill this in

#include <stdio.h>

// Function declaration
int factorial(int value);

// Function:     main
// Description:  entry point for programming assignment
// Parameters:   argc: number of arguments, argv: argument strings
// Return:       factorial (integer)
// Error Checks: none
int main(int argc, char *argv[])
{
      printf ("Factorial Program\n");
      for (int value = 0; value < 10; ++value) {
            printf("Value: %d, Factorial %d\n", value, factorial(value));
      }
}
```

Compile and run the factorial program, as shown below. Then replace the code in pa1.c with the print functions specified on the previous page, and modify main.c to code test the print functions. The main.c file must also declare the functions in pa1.c before calling them, exactly as shown on the previous page. Test each function from the main entry point with four different values.

Include these test cases as follows:

```
printIntegerInHexadecimal(01234567);
printIntegerInBinary(01234567);
printFloatInHexadecimal(0.125f);
printFloatInBinary(24.5f);
```

To compile both files into a program called pa1, type the following commands:

```
gcc -g -std=c99 -Wall -c pa1.c -o pa1.o
gcc -g -std=c99 -Wall -c main.c -o main.o
gcc -g pa1.o main.o -o pa1
```

To run the compiled program, type the following command:

```
$ ./pa1
```

Verify that the function returns the same result that you have calculated by hand. Do the test cases above match your results from Homework Assignment 1? Since the functions in this assignment all printed to standard out, you will need to visually inspect that the results are as expected. 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.

Question 3: Does the machine you are on use the little endian or big endian representation? How did you determine that?

Question 4: Did your program get the same result for problems 8 and 9 from Homework Assignment 1? If not, explain why there was a difference.

Submission Instructions

When you are done, your directory should have pa1.c, main.c, and a README file. Submit them using RAMCT.

Grading Criteria

We will use our own unit tests to verify the functions in pa1.c, and we will read and run all of your code.

The grading criteria is functionality (50 points), coding style and comments (25 points), following assignment directions (15 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.

Late Policy

Late assignments will be accepted up to 24 hours past the due date and time for a deduction of 25% and will not be accepted past this period. RamCT will reports assignments as late the second the deadline passes, so make sure and leave yourselves time to submit to RamCT. Please contact the instructor or teaching assistant if you have RamCT problems.