CMSC 11, Additional lab exercises on functions

Write functions for the following tasks. Also write main() drivers that would test your functions.

```
float cube( float x )
// computes and returns x*x*x
```

```
int sign( float x )
// returns 1 if the parameter x is positive, 0 if x is zero, -1 if x is negative
```

```
float squareroot( float x )
// computes and returns the square root of x using Newton's algorithm
// recall an estimate can be iteratively improved by replacing it with (estimate + x/estimate) / 2.0
```

```
float fourthroot( float x )
// computes and returns the fourth root of x by applying the square root function twice
```

```
float cuberoot( float x )
// computes and returns the cube root of x using a variant of Newton's algorithm
// hint: how would you iteratively improve an initial estimate for the cube root of x?
// using the last 3 functions, write a main() driver that prints a table of square roots, cube roots and fourth roots
```

```
double factorial( int n )
// compute and returns 1*2*3*...*n = n!
// we use a double (instead of int or float) for the result since this function grows very fast
// note: this function gives the number of ways n different letters can be arranged (permutations)
// e.g., if n=3, there are 1*2*3=6 different arrangements: abc, acb, bac, bca, cab, cba
```

```
double combinations( int n, int s )
// computes and returns n!/s!/(n-s)!
// note: this function gives the number of ways a committee of s members can be selected from n people
// e.g., if n=4 and s=2, there are 4!/2!/2! = 24/2/2 = 6 different 2-element subsets from {a,b,c,d}:
// {a,b}, {a,c}, {a,d}, {b,c}, {b,d}, {c,d}
```

```
char leapyear( int year )
// returns 1 if the parameter year is a leap year, 0 otherwise
// recall all leap years are divisible by 4 with some important exceptions on century years,
// use the notes in the calendar handout
```

```
void dayOfWeek( int month, int day, int year )
// prints “Sunday”, “Monday”, etc., use the notes in the calendar handout
```

```
void printcenter( char *s )
// prints the string s in the center of the screen (assume a screen that is  80 columns wide)
// you may use the predefined strlen() function where strlen(s) returns the number of chars in the string s
```

```
void printbits( int n )
// prints the 32-bit binary representation of the decimal integer n (output them in groups of 4 bits/nibble)
// note: the parameter n can be negative using the standard twos-complement notation
// hint: the shift right >> and logical & operators might be very useful in extracting the individual bits
```

jmsamaniego@uplb.edu.ph