# ECE 175: Computer Programming for Engineering Applications

Homework Assignment 4

Due Date: September 23, 2010 11:59 PM

Write comments to your programs. Programs with no comments will receive PARTIAL credit. At each program you turn in at least the following info should be included

- Author: YOU MUST PUT YOUR NAME ON THE PROGRAM OTHERWISE IT WILL NOT BE GRADED!
- Date created:
- Brief (two-line) description of the program
- Submit .c files

**Problem 1:** After studying the population growth of Gotham City in the last decade of the 20th Century, we have modeled Gotham's population function as:

$$P(t) = 52.966 + 2.184t,$$

where $t$ is the number of years after 1990, and $P$ is the population in thousands. Thus $P(0)$ represents the population in 1990, which was 52.966 thousand people. Write a program that defines a function named *population* that predicts Gotham's population in the year provided as an input argument. Write a program that calls the *population* function and interacts with the user as follows:

Enter a year after 1990: 2015
Predicted Gotham City population for 2015 (in thousands): 107.566.

**Problem 2:** Write a program that calculates the speed of sound ($\alpha$) in air of temperature $T$ (in Fahrenheit). The formula to compute the speed in ft/sec is:

$$\alpha = 1086\sqrt{\frac{5T + 297}{247}}. \tag{1}$$

Be sure your program does not lose the fractional part of the quotient in the formula shown. Your program should ask the user to enter the current air temperature and repeat until the user wants to quit. As part of your solution, write a call function that displays instructions to the program user.

**Problem 3:** Write a program that finds all prime numbers contained in file "numbers.dat" and places them in file "primes.dat". A prime number (or a prime) is a natural number that has exactly two distinct divisors: 1 and itself.

Your program should be modular. Write a function named *primality* that takes as an argument an integer number. The function returns 0 if the number is not prime and 1 if the number is a prime. For primality testing use the following algorithm.

*Algorithm:* Let $n$ be the number tested for primality. If $n$ is not divided by any number $m$ smaller than $\sqrt{n}$ then $n$ is a prime. Else $n$ is not a prime.

Example: 13 is a prime because no number less than 4 divides it.

The rationale behind this simple algorithm is the fact if a number $n$ can be factored, at least one of the factors has to be less or equal to its square root $\sqrt{n}$. This is easily shown via contradiction. If $n$ could be analyzed to two factors $a > \sqrt{n}$ and $b > \sqrt{n}$, then $ab > n$. Hence, at least one factor has to be less or equal to $\sqrt{(n)}$.

Prime numbers used for cryptographic applications are typically very large (in the order of thousand bits long).

**Problem 4:** Write a program for an automatic teller machine (ATM) that dispenses money. The user should enter the amount desired (a multiple of 10 dollars) and the machine dispenses this amount in the *least number of bills*. The bills dispensed by the ATM are 50s, 20s, and 10s. Write a function named *bills* that takes as an argument the withdrawal amount and prints how many of each kind of bill to dispense. Your program should be checking if the user has entered a multiple of 10. Also it should be interactive prompting the user on whether he wants to complete another transaction.

**Submit your .c files named problem1.c problem2.c problem3.c problem4.c via D2L dropbox**