ECE 175: Computer Programming for Engineering Applications

Homework Assignment 5

Conventions: Name your C programs as *hwx_py.c* where x corresponds to the homework number and y corresponds to the problem number. As an example the C program for hw1 problem 1 should be named as *hw1_p1.c*.

Submission Instructions: Use the dropbox on D2L to submit only the .c files.

Problem 1 (15 points): 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:

Sample Code Execution: **Red text indicates information entered by the user**

```
Enter a year after 1990: 2015
Predicted Gotham City population for 2015 (in thousands): 107,566.
```

Problem 2 (15 points): Write a C program that computes the sum of a series.

$$S = \sum_{i=1}^{n} i^j.$$

Your program should use the following two functions.

long int power(int base, int exp);

long int series(int n, int j);

The first function returns base raised to the exp power whereas the second function returns the series sum

Sample Code Execution: **Red text indicates information entered by the user**

```
Enter the values of n and j : 5 2

1^2 + 2^2 + 3^2 + 4^2 +5^2 = 55
```

Problem 3 (20 points): Write a program that finds all prime numbers from 1 to 1,000,000. Your program should call a user-defined function called "primes" that returns PASS (i.e., 1) if a number is prime and FAIL (i.e., 0), otherwise. The function prototype should be as follows:

int prime(int p);

A prime number (or a prime) is a natural number that has exactly two distinct divisors: 1 and itself. For primality testing use the following algorithm.

Algorithm: Let n be the number tested for primality. If n is not divisible by any number m smaller than $\lceil\sqrt{n}\rceil$ 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 (analyzed to factors other than 1 and itself), 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).

Details on the longest known prime can be found here: `https://www.mersenne.org/primes/?press=M74207281`.

Problem 4 (20 points): Write a program for an automatic teller machine (ATM) that dispenses money. The user should enter the desired withdrawal amount (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 argument the withdrawal amount and prints the types of bills dispensed. Your program should be checking if the user has entered an amount that is multiple of 10.

Sample Code Execution: **Red text indicates information entered by the user**

```
Enter the withdrawal amount (must be multiple of  $10):325

The amount you entered is not a multiple of $10

Enter the withdrawal amount in $:380
Your dispensed bills are:
   $50: 7
   $20: 1
   $10: 1


Enter the withdrawal amount (must be multiple of  $10):160
Your dispensed bills are:
   $50: 3
   $10: 1
```

Lab Assignment (30 points)

Submit your .c files named hw5_p1.c, hw5_p2.c, hw5_p3.c, and hw5_p4.c on ZyBooks and via D2L dropbox