CSE 2451 Lab 1

Objectives

- iteration and selection statements
- C functions
- recursion

Introduction

In this lab, the students will write customized function(s) in one source file, **lab1.c**, and compile the source file to executable program.

1. Perfect number (6 pts)

An even perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. A divisor of an integer x is an integer that can divide x evenly with 0 remainder.

Given an integer **num**, write a checkPerfectNumber() function matching the following function prototype. The function should return 1 if **num** is an even perfect number, and return 0 otherwise. The value of input **num** will range between $[-10000, 10000]$.

function prototype:

```c
int checkPerfectNumber(int num);
```

Example 1:

```
Input: num = 28
Output: 1
Explanation: 28 = 1 + 2 + 4 + 7 + 14
1, 2, 4, 7, 14, 28 are all divisors of 28
but 28 is excluded because it is the input number itself
```

Example 2:

```
Input: num = 16
Output: 0
Explanation: 16 != 1 + 2 + 4 + 8
1, 2, 4, 8, 16 are all divisors of 16
but 16 is excluded because it is the input number itself
```

Note: you may use a brute force solution iterating through 1 to num, time complexity of your solution is not a concern for this lab.

2. A More General Power Function (bonus 4 pts)

A more general solution for the power function example shown in the lecture slides:

```c
int power(int base, int exponent) {
    int result;
    if (exponent == 0) // base case
        result = 1;
    else // recursive case
        result = base * power(base, exponent - 1);
    return result;
}
```

In the above example, the power() function is limited to non-negative input for its exponent parameter (runs into segmentation fault). Can you write a more general power function that accepts both negative and non-negative input values for the exponent parameter? You may assume input for base is gauranteeed to be non-zero. Your function definition should match the given function prototype below:

function prototype:

```c
double power_int(double base, int exponent);
```

3. Source File Template (4 pts)

Your source file should follow the template below, using the exact test cases given in the template. **Note**: the conversion character (placeholder) used in the following template is "%lf" (abbreviation of 'long float', not 1f 'one f').

```c
#include <stdio.h>
// put your function prototypes here
int checkPerfectNumber(int num);
double power_int(double base, int exponent); // if you finished the bonus
int main(void) {
    // test case for even perfect number
    int p = 496;
    int n = 36;

    printf("checkPerfectNumber(%d) = %d\n", p, checkPerfectNumber(p));
    printf("checkPerfectNumber(%d) = %d\n", n, checkPerfectNumber(n));

    // if you finished the bonus question
    double b1 = -7.1, b2 = -3.0; int e1 = -4, e2 = 4;
    printf("power_int(%lf, %d) = %lf\n", b1, e1, power_int(b1, e1));
    printf("power_int(%lf, %d) = %lf\n", b2, e2, power_int(b2, e2));
    return 0;
}
// put your function definition(s) here


// End-Of-File
```

4. Compile and Run Your program

Nagivate the terminal to the directory of your lab1.c source file, and compile your program with the following command:

```
gcc -Wall lab1.c -o prog -std=c99
```

You need to make sure your source file compiles with no error or warning messages from gcc. If you only finished part of the assignment, please comment out the incomplete or incorrect parts so that I can still compile your submitted source file. **Note: if your source file does not compile properly, you will get 0 points for your submission.**

5. Submission

You only need to submit the source file **lab1.c** to carmen. Please submit plain text file (the exact text file you used to compile the executable program), not a Word or PDF version of it.