CS 137: Assignment #2

Due on Friday, Sep 30, 2022, at 11:59 PM

Submit all programs using the Marmoset Submission and Testing Server located at https://marmoset.student.cs.uwaterloo.ca/

Victoria Sakhnini

Fall 2022

Notes:

- Use the examples to guide your formatting for your output. Remember to terminate your output with a newline character.
- Integers should be read using scanf.
- You are allowed to use only syntax/language features that we have covered so far up to the end of M4.
- You must NOT use MATH Library

Problem 1

<u>Definition</u>: Let $\sigma(n)$ represent the sum of all [positive] proper divisors of an integer n, that is, the sum of the [positive] divisors not equal to the number. For example, $\sigma(16) = 15$ since the [positive] proper divisors of 16 are 1; 2; 4; 8 and their sum is 15. Then, n is...

```
... abundant if and only if \sigma (n) > n. 
 ... perfect if and only if \sigma (n) == n. 
 ... deficient if and only if \sigma (n) < n.
```

<u>Task</u>: Create a C program isAbundant.c that reads an integer and prints exactly one of Abundant, Perfect, or Deficient depending on the above scheme, followed by a newline character.

Assumptions:

• You may assume all input satisfies n > 1 and that the values entered are valid integers with a magnitude at most of 10^9 .

Sample Input #1

6

Sample Output #1

Perfect

Sample Input #2

3

Sample Output #2

Deficient

Sample Input #3

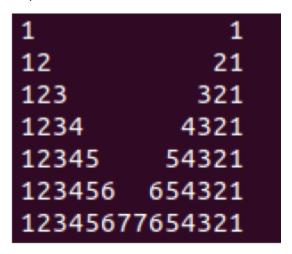
Sample Output #3

24

Problem 2

Create a program vpattern.c, which reads a positive integer n (assume n<=9) and prints a pattern of n rows. See examples below.

For input 7:



For input 1:

11

Note: each line ends with \n

You must use loops for this question.

Note: If your solution passes the public test on marmoset, your solution compiles.

Only the secret test after the deadline will check if you printed the correct shape.

Problem 3

- a) Create the file functions.h which contains the following declarations:
- int isSophieGermainPrime(int p);
- int base2nat(int bs, int num);
- int nat2base(int bs, int num);
- b) Implement all the functions above (explained below) in the file functions.c

<u>Note</u>: You are to submit this file (along with functions.h file) containing only your implemented functions (that is, you must delete the test cases portion and the main function). However, <u>you must keep the required included libraries.</u>

c) Submit functions.zip which contains the files functions.c, and functions.h Here are the objectives of the three functions:

1)

<u>Definition</u>: A Sophie Germain¹ prime is a [positive] prime number p such that 2p + 1 is also a prime number. For example, 2 is a Sophie Germain prime since both 2 and 2(2) + 1 = 5 are prime numbers.

Task: Create the function

```
int isSophieGermainPrime(int n)
```

which determines if an integer ${\bf n}$ is a Sophie Germain prime. The function should return 1 if ${\bf n}$ is a Sophie Germain prime and 0 otherwise.

Assumptions: You may assume that the values entered are valid integers such that the magnitude of 2n + 1 is at most 10⁹.

Fast Facts: Sophie Germain

Known For: French mathematician, physicist, and philosopher specializing in elasticity theory and number theory.

 ${\bf Also\ Known\ As:}\ {\bf Marie\text{-}Sophie\ Germain}$

Born: April 1, 1776, in Rue Saint-Denis, Paris, France

Died: June 27, 1831, in Paris, France **Education**: École Polytechnique

Awards and Honors: Number theory named after her, such as Sophie Germain prime, Germain curvature, and Sophie Germain's identity. The Sophie Germain Prize is awarded annually by the Foundation Sophie Germain.

When you see a number such as 734 it is generally assumed you are using the base 10 number system (also known as the decimal system). That is:

```
734 = 7*10^2 + 3*10^1 + 4*10^0
```

It is, however, possible to use any number as a base. For example, assuming we are in a base 5 number system, the notation 2301 would generate the decimal number 326:

```
2*5^3 + 3*5^2 + 0*5^1 + 1*5^0 = 326 (this equation is in decimal)
```

Note that when using base 10 we have precisely 10 unique digits for each position, that is 0,1,2,3,4,5,6,7,8,9. Similarly, base 5 only allows for 5 digits 0,1,2,3,4 (To represent the "normal" (i.e. decimal) value 5 in base 5 we would write 10; 6 would be represented as 11, 7 would be 12 etc. To see this, consider 12 (in base 5) means we compute $1*5^1 + 2*5^0$ to get 7).

Task: Create the function

```
// pre: 1<bs<10 and num>0
int base2nat(int bs, int num)
```

which returns a positive integer representing the decimal value of num (num is in base bs).

<u>Assumptions</u>: You may assume that the values entered are valid integers such that the magnitude of num is at most 10^9 .

III)

```
// pre: 1<bs<10 and num>0
int nat2base(int bs, int num);
```

It takes a base (bs) and a non-negative integer (num) in decimal and returns the value in base bs.

The following code will help you for testing

```
1. #include <stdio.h>
2. #include <assert.h>
3. #include "functions.h "
5. int main(void){
            assert(isSophieGermainPrime(11));
6.
7.
            assert(isSophieGermainPrime(41));
8.
            assert(base2nat(5,23114)==1659);
            assert(base2nat(7,1)==1);
10.
            assert(base2nat(3,1211012)==1328);
11.
            assert(base2nat(8,715)==461);
12.
            assert(nat2base(5,1659)==23114);
13.
            assert(nat2base(9,1331)==1738);
14.
```