cs201 – sum 2020

Assignment 1 – (40 points)

Download the assign1.c file from the assignment 1 folder on d2l. Do the following functions inside this file.

Part A: Hex to Binary(15 points)

In the given file assign1.c, write a function that takes a 4 digit hexadecimal number as a string of 0-9 or A-F values and prints out the binary digits. Assume the user will type in all capitals for letters.

HINT: the ASCII code for 0-9 starts at 48 and ends at 57. The ASCII code for A - F starts at 65 and ends at 70. Anything else is not a valid hex digit; if you encounter a invalid hex digit print an error message and return.

```
----------------------
EXAMPLE RUN #1:
----------------------
enter hex digits: 34AD
0011010010101101
----------------------
EXAMPLE RUN #2:
----------------------
enter hex digits: 87JA
10000111
invalid digit J. Stopping now!
----------------------
```

Part B: Using Bit-Wise Operators (15 points)

Write a function that takes an unsigned integer num (note this is always positive or zero) and an int position pos (0..31) and returns the value of the (**pos**)th bit as an integer (this will be either 1, 0 or -1 for error). Check for pos being between 0 and 31 and if it's not return a -1.

```
----------------------
EXAMPLE RUN #1:
----------------------
enter number: 8
enter position (0 to 31): 3
value is 1
----------------------
EXAMPLE RUN #2:
----------------------
enter number: 8
enter position (0 to 31): 2
value is 0
----------------------
EXAMPLE RUN #3:
----------------------
enter number: 1000
```

```
enter position (0 to 31): 100
value is -1
```

Part C: Detecting Overflow (10 points)

write a function that takes two int numbers. It then returns a TRUE if there is an overflow (both positive and negative) of the sum of the two numbers.

HINT: For testing purposes, the largest positive int value before overflow is 2147483647. The largest negative value is -2147483648. When testing, make sure the two numbers aren't overflows themselves or you're function wont work. Get two numbers whose magnitude is less than 2147483647 for positive (negative: 2147483648) but the sum's magnitude is greater than 2147483647 for positive (negative: 2147483648). A positive and a negative number's sum won't overflow if neither number is an overflow itself.

```
--------------------------------
EXAMPLE RUN #1:
--------------------------------
enter two numbers to add: -2147483648 -1
there has been overflow
----------------------
EXAMPLE RUN #2:
----------------------
enter two numbers to add: 1000000000 2000000000
there has been overflow
----------------------
EXAMPLE RUN #3:
----------------------
enter two numbers to add: 100000 400000
the sum is okay
----------------------
EXAMPLE RUN #4:
----------------------
enter two numbers to add: 2147483647 -1
the sum is okay
----------------------
```