CS252: Homework 2

Number Conversions

Purpose:

The purpose of this assignment is to give you practice programming in C and working with number conversions.

Submission Requirements:

Submit your code (.c files) along with screenshots of your code compiling and running. All files should be submitted as a single zip to Canvas.

Grading Criteria:

Your code must compile to receive a score. Programs without comments will not receive full credit.

Problem 1:

The focus of this problem includes:

- Understanding number conversion

Using your solution to Lab 6, make a copy of your methods and upgrade them to do any base conversion to decimal and back. The signatures (prototypes) should look like:

```
int baseToDec(int base, char* value);
char* decToBase(int base, int dec);
```

Example calls would be:

```
int dec = baseToDec(2, "11001");
int dec = baseToDec(8, "157");
int dec = baseToDec(16, "f8");
```

Try to make as few changes to your code as possible. Look at what you can generalize when modifying the base 2 only code. Your code should work for any base from 2 up to 16. But please do not code every possible case individually!

Problem 2:

The focus of this problem includes:

- C programming practice
- Using the conversion methods you created

Note: Carefully read the program description to ensure you are completing the tasks as required.

Write a program that allows the user to perform simple arithmetic with different bases.

Upon starting, the program should tell the user that it is a variety base math program along with brief instructions on how to use the program.

The program should then enter a loop where it gives a prompt, e.g. “base: “ followed by “input: “. Upon receiving input from the user the program should process it, report the output result (or error), and loop

back to the prompt. This should continue until the user gives a keyphrase to exit the program (e.g. “quit”, “end”, or “exit”).

Input from the user should begin with a base, e.g. 5, 2, or 16 (bases between 2 and 16 should be supported). Input from the user should then be of the form NSN, where N represents a number in the given base and S represents a mathematical symbol. There should be no spaces between the three parts (the two N’s and the S).

Binary numbers should be no more than seven characters long (e.g. 1101100). All other bases should be no more than four characters long. The mathematical symbol should be from the set + - / * %. These symbols correspond to their natural use in the C language: addition, subtraction, division, multiplication, and modulus.

If the user inputs a binary number with more than seven digits (or any other base with more than four digits), the program should report a meaningful error message. If the user inputs a mathematical symbol outside the expected set, the program should report an error. If the input provided by the user is not of the form NSN (e.g. if it is only SN or only N), then the program should report an error. All error messages should be meaningful in their context, telling the user what was wrong with the input and allowing them to enter correctly formatted input.

If a grammatically correct input is received, the program should convert the two numbers to base 10 and perform the given operation. The program should then convert the result back to the original base and report it to the user.

For example:

```
Base: 2
Input: 0000101+0001100
0010001
```

If the mathematical operation given by the user results in an answer requiring more than 7 bits (in binary) or 4 digits (other bases), the program should report an error message of bit/number overflow. If the mathematical operation given by the user is invalid (e.g. division by zero), the program should report a relevant error message.