Homework 5: Numeric ⬄ String Conversion with Multiple Bases

**Assigned**: February 27th, 2018 **Due**: March 8th, 2018

Make a subdirectory "hw5" of your cs240 folder for this assignment. Copy the starter files from /courses/cs240/s18/kamaral/GROUP/hw5 to your hw5 subdirectory.

All parts for this homework will be in a single file, convert.c.

**Part 1: Converting One Digit from String to Numeric**

In convert.c, there is a function called digit\_btoi which takes two arguments, a character and a number.

The character is the string representation of a digit in the following base.

The number is that representation’s base.

digit\_btoi has a return value of integer and should return the value of the digit as a number.

Remember, after representations of base 11 and higher, we need to use more than just the usual ASCII numbers for digits. For example, in Base 16, we need to use all the way to F to represent 15.

This function should work on the following characters up to the last digit used in the base:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ

For example, in base 16, only the following digits should be acceptable

0123456789ABCDEF

For any digit invalid digit or invalid base, the function should return -1 or INVALID\_DIGIT.

But for base 36, all numbers and letters should be used. Also, make sure that the function works with lowercase or uppercase digits.

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ

0123456789abcdefghijklmnopqrstuvwxyz

Example values:

digit\_btoi('0', 16) = 0

digit\_btoi('F', 16) = 15

digit\_btoi('1', 16) = 1

digit\_btoi('Z', 36) = 35

digit\_btoi('z', 36) = 35

digit\_btoi('q', 10) = -1

digit\_btoi('@', 10) = -1

digit\_btoi('9', 0) = -1

You’re only expected to implement from base 2 to up base 36 in this assignment. All other bases should be considered invalid. (Technically, you can do base 1, but it would be the number system consisting only of 0's).

**Part 2: Converting One Digit from Numeric to String**

In convert.c, there is a function called digit\_itob which takes two numeric arguments.

The first number is the numeric digit value, up to the base.

The second number is the base with which to give the representation.

digit\_itob has a return value is going to be a character, which is going to be the string representation of that digit value in that base. Let's say that we only output uppercase from this function.

If the inputs are invalid, return '\0'.

Think of this function as the opposite of the one from Part 1.

Part 1 takes String digits to Numeric values.

Part 2 takes Numeric digits to String values.

digit\_itob( 15, 16) = 'F'

digit\_itob( 0, 10) = '0'

digit\_itob( 1, 16) = '1'

digit\_itob( 35, 36) = 'Z'

digit\_itob( 35, 10) = '\0'

digit\_itob( 10, 0) = '\0'

digit\_itob(-1, 10) = '\0'

**REMEMBER FOR PART 1 AND 2**

Character values are inherently numeric, so you can do things like:

Boolean condition for checking if c is a capital letter: 'A' <= c && c <= 'Z'

Finding c's offset from Capital A: c - 'A'

**Part 3: Converting a Full String to Number**

In convert.c, there is a function called btoi which takes two arguments, a character array (for a string) and a numeric argument.

The first argument is the string representation of a number in a base.

The second argument is the numeric base.

The return value for this function is going to by the numeric value for the string representation of the number. It's very like atoi in the class notes.

If the number is invalid, like with atoi, we're going to return 0. Ignore leading spaces.

The function should return once you reach the first invalid character in the string, just like atoi.

This function should be able to make use of the digit\_btoi function on each character in the string to save you some headaches. That way, you don't need to fumble with ASCII code manipulation in this part of the code.

btoi("ZOOT", 36) = 1664957

btoi("HELLO", 36) = 29234652

btoi("2018", 10) = 2018

btoi("7F", 16) = 127

btoi("9", 8) = 0

**Part 4: Converting from a Number to a Full String Representation**

In convert.c, there is a function called itob which takes three arguments, a character array (for a buffer) and two numeric arguments.

The first argument is a buffer which has been passed to the function so that it can return a string value, like with fscanf and fgets.

The second argument is the numeric base for which you should be producing the representation.

The third argument, is the actual number that you're producing the representation of.

This function is void, since it is returning the string value through the first argument.

If the number is negative, the function should return the string "0\0".

Follow the process we described in the discussion on Modulo to extract digits from a number.

This function should be able to make use of the digit\_itob function on each digit, again to avoid the headache of doing ASCII manipulation in this function.

itob(buffer, 36, 1664957) buffer = "ZOOT"

itob(buffer, 36, 29234652) buffer = "HELLO"

itob(buffer, 10, 2018) buffer = "2018"

itob(buffer, 16, 127) buffer = "7F"

itob(buffer, 10, -1) buffer = "0"

**Final Remarks**

The main method is already written for you. Please do not modify it.

You can test whether or not your program works by running it and giving it the example inputs I mentioned above.