UMass Boston CS 240
Homework 3
Due 10/21/2019 18:45

1 Note

1. Write your full name in your code.
2. The algorithm offered in instructions are just suggestions not requirements. You are welcomed and encouraged to do this homework with other algorithms.
3. You don't have to write a general program, just satisfy the test of what required will get the full score. Especially for getting those integers, if it just works on the format of the text offered is OK and will not lose points.
4. You are not allowed touse any integer to string manipulation library functions, such as itoa(), etc.
5. The submission will be three files under the hw3 folder: itox.c, itox.h, itoxDriver.c.

2 Introduction

The decimal number system probably arises from our tendency of counting with our fingers. It is not the only way. A tribe of people have so many four-legged animals that they use the quaternary number system – base 4. The digits are 0, 1, 2, and 3. For example, one, two, three, four, five, six, seven, and eight are represented as 1_4, 2_4, 3_4, 10_4, 11_4, 12_4, 13_4, and 20_4. Another tribe love their toes as much as fingers so they use the vigesimal number system – base 20. The digits are 0, 1, . . ., 9, A, B, . . ., and J. For example, the decimal numbers 30, 31, and 32 are represented as $1A_{20}$, $1B_{20}$, and $1C_{20}$.

This assignment is to write C code that converts a *positive* integer among its quaternary, decimal, and vigesimal representations.

3 Integer to Quaternary String

Write the C function `itoq()` to covert an integer to a quaternary string. Its prototype is described in the header file `itox.h`. Write your code in the file `itox.c`. You can build from the stub `itox.c` that provided.

The idea behind `itoq()` is to convert an int variable (an `int` has 32 bits on our machine) directly to an ASCII string of quaternary digits, '0', . . ., '3'. In this assignment, we will consider only positive integers. The algorithm is as follows.

1. Divide the int by 4.
2. The remainder is the first quaternary digit, to be placed in `quaternaryStr[15]`.
   Use '0' to represent remainder = 0, '1' for remainder = 1, and so on.
3. Update the int to be the quotient.
4. Repeat steps 1, 2, and 3 for `sizeof(int) * 4` times.
   When Step 2 is executed for the second time, the quaternary digit will go into `quaternaryStr[14]`; when Step 2 is executed for the third time, the hex digit will go into `quaternaryStr[13]`; and so on.
5. Terminate `quaternaryStr` with '`\0`'.

Note that the array `quaternaryStr` declared by the caller can be declared with a size `sizeof(int) * 4 + 1`. `sizeof()` is evaluated at compile time to be the number of *bytes* in a variable of a given type. For example, it is 4 bytes to an int on our machine, but it might be 8 bytes on a different machine. Thus `sizeof()` allows your code to be portable.

4 Quaternary String to Integer

Write the C function `qtoi()` to convert a quaternary string to an integer. Its prototype is also described in `itox.h`. Write your code in file `itox.c`. Your code should only accept the valid quaternary digits: 0, . . ., 3. An algorithm is as follows.

1. Start at the last character in the string, which is the least significant digit.
   Remember to skip over the '`\0`' character.
2. Convert the character to its decimal equivalent.
   For example, character '2' is 2.
3. Multiply the decimal number by powers of 4.
   The last character will be multiplied by 4^0, the next by 4^1, and so on.
4. Repeat for all of the characters in the quaternary string.
5. Sum the products.

5 Integer to Vigesimal String

Write similar functions for conversions to and from vigesimal strings. The differences are the following:

- The vigesimal number system is base 20.
- The digits are '0', . . ., '9', 'A', . . ., 'J'.
- From a 32-bit int, there can be at most 8 vigesimal digits. So `vigesimalStr[]` must have `sizeof(int) * 2 + 1` bytes.

6 The Driver

The driver is the main program that will call `itoq()`, `qtoi()`, `itov()`, and `vtoi()`. This is not the same thing as a device driver. The driver should be named `itoxDriver.c`, and you can use the stub `itoxDriver.c` from my hw3 directory.

Now you have three files of source code: the header file `itox.h`, the utility file `itox.c`, and the driver `itoxDriver.c`. The following is how you compile multiple files into one executable, and test your code.

```
gcc -Wall itox.c itoxDriver.c -o itox
./itox < test.txt
```

Make a subdirectory hw3 in your home directory for this assignment. You are not allowed to use any integer to string manipulation library functions, such as `itoa()`, etc.

7 A Static Library

(This is for playing, you don't have to do this or submit this. We will get to it later.)

After you are sure your `itoq()`, `qtoi()`, `itov()`, and `vtoi()` are working correctly, you can compile and build a *static library* as follows.

```
gcc -Wall -c itox.c     //compile only, produce xbits.o

ar rc myLib.a itox.o    //build a static library myLib.a

//compile the driver and link/load the utilities from the library
gcc -Wall -L/home/user??/hw3 itoxDriver.c -o itox myLib.a

./itox < test.txt
```