Homework 3 C Programming: Binary Representation

Due Tuesday 9/10/2019 11pm

This assignment you will write your first C program for the class. We have provided a tar packet that includes some starting C code that read a hexadecimal value from command line. Your job is to fill in the code so that you will print the binary representations (in 1's and 0's) of this hexadecimal value stored in an unsigned long variable. You may assume that your program will be run in an environment where unsigned long is stored in 8 bytes (64 bits).

Download hw3handout.tar

Use instructions similar to homework 2 to download hw3handout.tar file, copy it to w204 machine's 311 folder, use `tar xvf` command to unpack it to hw3handout folder.

Configure your vim editor and learn to use it

You will need to use an editor to write your C code while you remotely logged onto W204 machine. We highly recommend you give the vim editor a try. Learning to use a terminal based editor will give you an extra tool when you are working with systems that need to be remotely managed. You will find a help document prepared by our teaching staff about some basics of using vim.

There is also an online tutorial of using vim editor you could try out.
https://www.openvim.com

We have specific requirements for proper indentation of C code. In particular, we ask that you indent each level of code with exactly **2 spaces (no tab indentation).** The easiest way to do this if you decide to use vim is to do the following command on a w204 machine where you download and run a shell script from our website to configure your vim editor. The commands you type are in bold red.

```
cse-p204inst33.cse.psu.edu 123% ls
311  311backup  bin  old311  pastclasses  perl5  recommendation  test  www
cse-p204inst33.cse.psu.edu 124% cd 311
cse-p204inst33.cse.psu.edu 125% curl http://www.cse.psu.edu/~yuw17/cmpsc311/vagrant/install.sh >
install.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   479  100   479    0     0  32855      0 --:--:-- --:--:-- --:--:-- 34214
cse-p204inst33.cse.psu.edu 126% chmod +x install.sh
cse-p204inst33.cse.psu.edu 127% ./install.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    82  100    82    0     0   3661      0 --:--:-- --:--:-- --:--:--  3727
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
```

```
                                 Dload  Upload   Total   Spent    Left  Speed
100    64  100    64    0     0   3146      0 --:--:-- --:--:-- --:--:--  3200
```

Edit/Compile/Test your C code

1. Compile your code

Assuming you successfully copied and unpacked hw3handout.tar file in ~/311 folder on your W204 account, the following command will help you compile the given C program. The original tar file contains a complete C program that compiles and works.

```
cse-p204inst11.cse.psu.edu 160% cd ~/311
cse-p204inst11.cse.psu.edu 161% ls
hw3handout  hw3handout.tar  install.sh
cse-p204inst11.cse.psu.edu 162% cd hw3handout
cse-p204inst11.cse.psu.edu 163% ls
binary.c  Makefile
cse-p204inst11.cse.psu.edu 164% make
gcc -Wall -Og -g -o binary binary.c
cse-p204inst11.cse.psu.edu 165% ls
binary  binary.c  Makefile
cse-p204inst11.cse.psu.edu 166% ./binary 0xa
10
cse-p204inst11.cse.psu.edu 167% ./binary 0x1000
4096
cse-p204inst11.cse.psu.edu 168% vim binary.c
cse-p204inst11.cse.psu.edu 169% vim binary.c
cse-p204inst11.cse.psu.edu 170% make
gcc -Wall -Og -g -o binary binary.c
cse-p204inst11.cse.psu.edu 171% ./binary 0xb
11
cse-p204inst11.cse.psu.edu 172% make
make: `binary' is up to date.
```

You use make command in the hw3handout folder to compile the binary.c file to create an executable called binary. If you didn't modify the content of binary.c after a make, trying make again it will say that "'binary' is up to date." which implies there is no need to recompile the c file because nothing has changed since the last time.

Make sure that your make does not generate any error/warning messages while compiling your C code. Any warning/error messages while compiling your code will result in a zero for this assignment. For example, the following make command shows some warnings. You must address these warnings and remove all compiler warnings/error messages before you submit.

```
[vagrant@localhost hw3]$ make
gcc -Wall -Og -g -o binary binary.c
binary.c: In function 'main':
binary.c:8:9: warning: 'j' is used uninitialized in this function [-Wuninitialized]
   printf("%d\n", j);
          ^
```

2. Test your code

Currently the code you are supplied simply prints out the decimal (base 10) value of the hexadecimal number passed in the command line. Your job is to modify and write C code so that it prints out the following binary representations of the number.

```
[vagrant@localhost hw3]$ ./binary 0xa
1010
[vagrant@localhost hw3]$ ./binary 0xffff
1111111111111111
[vagrant@localhost hw3]$ ./binary 0xffffeeeeddddcccc
1111111111111111111011101110111011101110111011101110011001100
[vagrant@localhost hw3]$ ./binary 0x0
0
[vagrant@localhost hw3]$ ./binary 0x3a
111010
```

Your solution should work in the following ways:

- It prints out binary representations for all numbers representable by a 8-byte 64 bits unsigned long int.
- It doesn't print out the leading zeros in the binary representation (except for 0x0 where you print out a single 0).

Test your code thoroughly with different input values as long as it is between `0x0` and `0xffffffffffffffff`.

3. Edit your code

Edit your code with vim and save it and repeat step 1/2 to compile and test your code.

Always start with small changes before your compile and test your code.

Keep your code properly indented.

Add comment in your function to explain your algorithm for this assignment. Use block comments that are on its own lines. Do not use tail comments at the end of each line of the code.

Submit your C code

You are to submit just the **binary.c** file to gradescope.