Project 2 Problem 2 Puzzlebox Guide

10-19-2021 - Alice Zhang

Helpful links

Helpful links:
Text to hex converter: https://www.browserling.com/tools/text-to-hex
Text to binary converter: https://www.convertbinary.com/text-to-binary/
All in one hext/binary/decimal converter:
https://www.rapidtables.com/convert/number/hex-to-decimal.html
Bitwise AND,OR,XOR,shift calculator:
https://miniwebtool.com/bitwise-calculator/?data_type=2&number1=01101000&number2=11111110&operator=AND
Collatz: https://www.dcode.fr/collatz-conjecture

General tips

Run gdb

gdb -tui file (will run gdb)
c (continue)
set args input.txt (set input to be from input.txt file)
b phase01 (set a breakpoint wherever you want)
i b (prints out all the break points you have, numbered in the order that you made them)

> delete 1 (only do this after you run the **i b** command this deletes the specific breakpoint number that you specify)

n (can also do next, but this will take you to the next line to execute) (also if you type in n once, you can just hit enter to go to the next line after)
step (steps into functions, also goes to the next line)

Phase 1

Inputs and types

Int a int b int c

Answer

a < c < b a
a can’t equal 0

Phase 2

Inputs and types

Int a int b int c

Answer

v = (a + hash%26) + b + (all the other calculations done on v)
a and b are arbitrary non zero integer inputs
c=v

How to solve

The value of c depends on the values of a and b
v = (a + hash%26) + b *ONLY INITIALLY*
And then there are a bunch of operations done on v
Use gdb debugger to print out the FINAL value of v
Then just set c equal to that value of v

Phase 3

Inputs and types

Int a int b int c int d

Answer

abcd are the integer indices where 1’s need to be inserted into shot to make it match targ

How to solve

This is exactly like the lab
So you need to print out the binary value of targ using print/t targ
Print out the initial value of shot (which is just a bunch of zeros) to help with your understanding
So basically the values of abcd are indices of shot where the 1's should go with the bit shifting
Count where the ones are in targ then get the binary value of targ to match the binary value of shot

Phase 4

Inputs and types

char s int a int b int c

Answer

char s is a char with a pattern of non repeating letters (in a row)
a is the index of the start of the first repetition
b is the index of the start of the second repetition
c is the index of the end of the second repetition (the length of s + 1)

How to solve

targ is going to be an integer based on the hash of your x500
So use gdb to print targ
targ is the length of the pattern you need to give for char s
abc are indices of the pattern.
a is the index of the beginning of the pattern
b is the index of the beginning of the recond repetition of the pattern
c is the index of the end of the second repetition of the pattern (length of s + 1)
It doesn't matter what your pattern is, but there are a few restrictions:
Your pattern cannot start at index 0 since a can't be 0
You can't have the same two letters in a row

Example

Say targ = 3
Then my input can be
acatcat 1 4 7

Take note that b=4 NOT 3 since the second pattern starts at index 4, not 3

Phase 5:

Inputs and types:

int n int m

Answer

You have solved the puzzle when the loop executes t number of times
In order for this to happen:
The input value of m has to = t
N needs to be big enough for the loop to execute t number of times so that s=t

How to solve

Put in some arbitrary number that is greater than 1 for n and m to first walk through the loop with gdb
Basically, n needs to be big enough for the loop to execute t number of times since for the loop to exit, s must = t

The value of m has to equal t

But in order for n to be big enough, know that:
if(n & 1) is checking if the ones value of the binary value of n is 1
n = (n << 2) - n + 1
So (n<<2) is equivalent to multiplying the decimal value of n by 4 (since (n<<1) is equivalent to multiplying the decimal value of 2

Fun clue

So phase 8 is the same thing as this essentially, except phase 8 is structured sort of like an assembly loop

I would practice comparing the two in preparation for the exam--it's also a great example of how C translates

Phase 6:

Inputs and types

int a int b int c int d

Answer

a=location1
c=location2
b=decimal value of reversed hex value of the characters that are overed with Xs and Ys

How to solve

a should = location1
c should = location2

To find b, print out actual and print out expect
Then reverse the order of letters for the word that's been replaced with X's or Y's
Then convert to hex
Then convert back to decimal

Example

Example:
Let's say the value to decode is " who", location1=12, and location2=20
Note that there is a SPACE before the "w"
Helpful tip: spaces = 20 in hexadecimal and 32 in decimal
Quotes are not part of the value in this example--just for better readability
Reverse " who" to get "ohw "
Hex value of "ohw " is 6F 68 77 20
Then convert to decimal to get 1869117216

So assuming the value you need to decode is the same for b and d, you just put the same decimal value for d

Final input is:
12 1869117216 20 1869117216

A note that if the value you need to decode may be different so just repeat the process in the example if that's the case

Helpful links:
Text to hex converter: https://www.browserling.com/tools/text-to-hex
Text to binary converter: https://www.convertbinary.com/text-to-binary/
All in one hext/binary/decimal converter:
https://www.rapidtables.com/convert/number/hex-to-decimal.html
Bitwise AND,OR,XOR,shift calculator:
https://miniwebtool.com/bitwise-calculator/?data_type=2&number1=01101000&number2=11111110&operator=AND

Why do we need to convert to hex at all? Can't we just go from ascii character -> decimal?

Because if you need to take the COMBINED hex value of the value you need to decode and then get the decimal

You do the conversion of the hex values TOGETHER
So if you have the two hex values of 20 and 6F
The converted decimal value of 20 put next to the converted decimal value of 6F are different
20 hex is 32 in decimal
6f hex is 111 in decimal
HOWEVER 206f is 8303 in decimal

Phase 7

Inputs and types

float a

Answer

The puzzle is solved fi a - flar.f is < 1e-8.

How to solve

If you expand 1e-8, it’s this super small number, so even though there are multiple possible answers for a, an easy way to approach this is to
Print out the valeu for flar.f and then make that your input for a

This works because 1e-8 is a super small number, but it’s still > 0, so if a - flar.f =0, then that works

Note, NO, you cannot just put in 0 for a--that’s wrong even though you may pass the tests because that’s a special case that the code didn’t account for

Why?
If the code checked for the value of 0 then returned an error, it would get weird with all the floating-point operations.

Phase 8(like phase05)

Inputs and types

Int n int m

Answer

You have solved the puzzle once s=t and m=t
This means that the input value of m has to = t
And also n must be large enough that the loop executes at least t number of times

How to solve

Here you have a weirdly structured loop
Put in some arbitrary number that is greater than 1 for n and m to first walk through the loop with gdb
Basically, n needs to be big enough for the loop to execute t number of times since for the loop to exit, s must = t

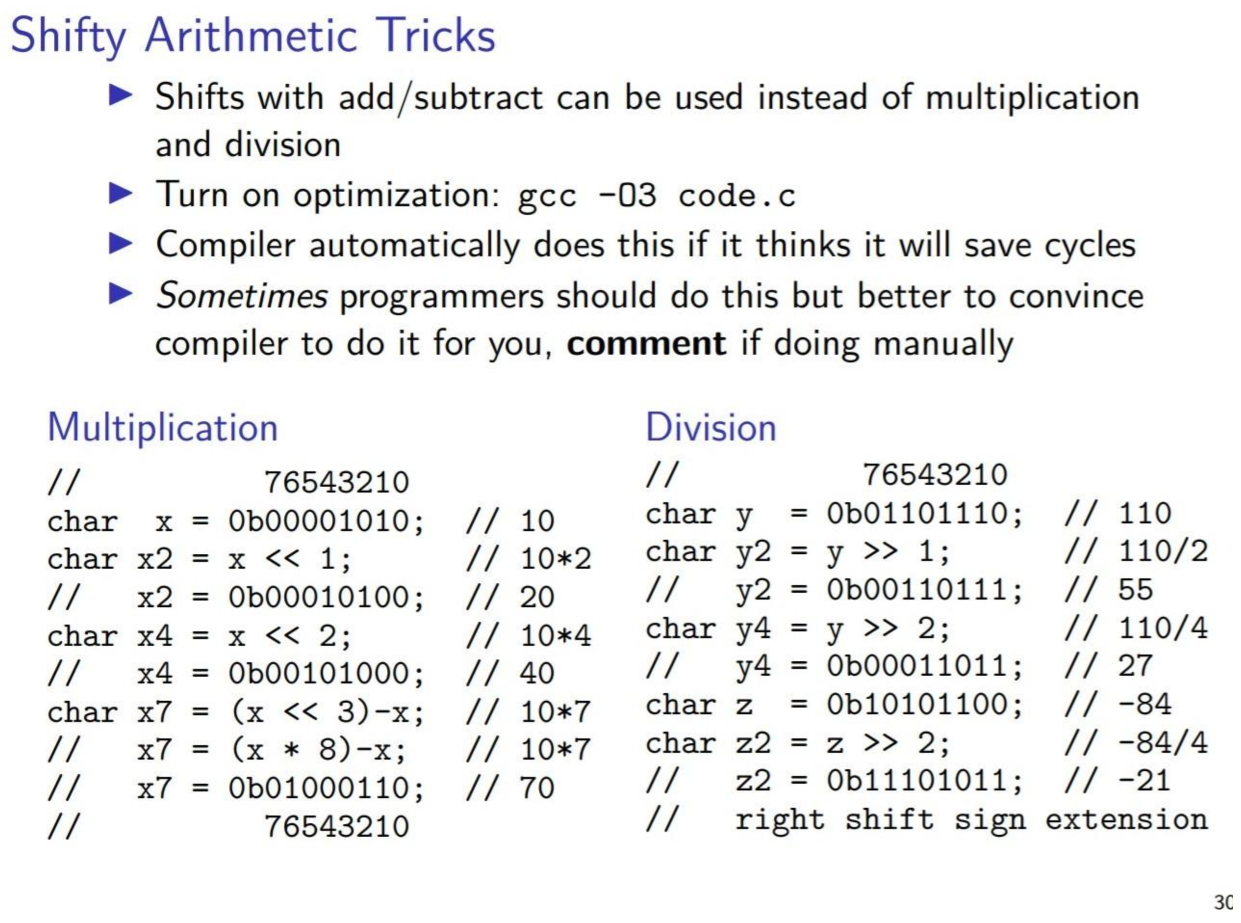
The value of m has to equal t

But in order for n to be big enough, know that:
if(n & 1) is checking if the ones value of the binary value of n is 1
n = (n << 2) - n + 1
So (n<<2) is equivalent to multiplying the decimal value of n by 4 (since (n<<1) is equivalent to multiplying the decimal value of 2

Example

Example input: Say my t = 37
1000 37

Notes for reference:



Shifty Arithmetic Tricks

- Shifts with add/subtract can be used instead of multiplication and division
- Turn on optimization: `gcc -O3 code.c`
- Compiler automatically does this if it thinks it will save cycles
- *Sometimes* programmers should do this but better to convince compiler to do it for you, **comment** if doing manually

Multiplication

```
//            76543210
char  x = 0b00001010;   // 10
char x2 = x << 1;       // 10*2
//    x2 = 0b00010100;  // 20
char x4 = x << 2;       // 10*4
//    x4 = 0b00101000;  // 40
char x7 = (x << 3)-x;   // 10*7
//    x7 = (x * 8)-x;   // 10*7
//    x7 = 0b01000110;  // 70
//            76543210
```

Division

```
//            76543210
char y  = 0b01101110;   // 110
char y2 = y >> 1;       // 110/2
//    y2 = 0b00110111;  // 55
char y4 = y >> 2;       // 110/4
//    y4 = 0b00011011;  // 27
char z  = 0b10101100;   // -84
char z2 = z >> 2;       // -84/4
//    z2 = 0b11101011;  // -21
//    right shift sign extension
```

30

Tip

Set a break point at the line of the last statement in the loop (LAFT) so you don’t need to click through

Phase 9

Inputs and types

Int a int b int c int d int e

Answer

You have solved the puzzle if abcde are the correct integer values for the indices of letters that correspond with what chong.s is supposed to be

How to solve

Print out chong.s
Let’s say it’s “waldo”
So print out letters
Count the indices of where “w” “a” “l” and so on are then input those integer values as the values of abcde

Example

Say letters = "enwgWLEKHoDvlPaucRjJiXBIGdNOrksyUTbQmAphSxFZVtfMYCqz"
And chong.s = "Waldo\000\000"
Then input is
4 14 12 25 9

Phase 10 (like phase06)

Input and types

Int a

Answer

The input is your x500 reversed, put into 7 bit binary values, then converted back to decimal

How to solve

Easiest way is to walk through an example
So my x5000 is zhan6698

Revered it's 8966nahz
Convert that to binary to get
00111000 00111001 00110110 00110110 01101110 01100001 01101000 01111010

Notice how for example, the letter z is this value in binary
01111010

This is 8 bits, not 7
So you need to get rid of that 0 highlighted in red
Then take that whole thing without the extra 0s
01110000111001011011001101101101110110000111010001111010

Convert to decimal to get
31777756196402298
For your answer

Why do we need to convert to binary? Can't I just get the decimal values of the characters from an ascii table? In other words, can't we just go from ascii characters -> decimal?

No, because of the issue with 7 bits, you need to convert to binary first, then take that entire binary number and convert back to decimal

So why do we need 7 bits for each character?

Because if you look at the for loop:

```
for(int i=0; i<8; i++){
    ans[i] = (char) (a >> (7*i) & 0x7F);
  }
```

(a >> 7) will cut out 7 bits (think 7 indices of the BINARY value of a) each time

Why? Because you only input one large integer so how else is the program supposed to get the value of each character? It’s like string slicing or array indexing

What’s with the & 0x7F?

Also the & 0x7F is doing the binary and operator with 1111 1110 (the binary version of the hexadecimal 0x7F).

Tip

This is super similar to phase06, if you haven’t noticed already