***Table of Contents***

[***Q1: Binary Arithmetic 2***](#_Toc69127452)

[***Q1.1. Add 11011 to 1011 2***](#_Toc69127453)

[***Q1.2. Rewrite tbase-10 numbers as 8-bit two's complement integers: -31, & -59 3***](#_Toc69127454)

[***Q1.3. What does the bit pattern 11101001 represent if you interpret it as an 8-bit two's complement integer? 4***](#_Toc69127455)

[***Q1.4. Draw up the truth table for the circuit below. From observing the result, what function do you think this circuit performs? 5***](#_Toc69127456)

[***Q1.5. Draw the circuit diagram for the Boolean logic equation: (AB + C) D 6***](#_Toc69127457)

[***Q2: Linux Assignment 7***](#_Toc69127458)

[***Q2.1. Enter the commands below at the Linux terminal on the AWS VM, and try to interpret the output. 7***](#_Toc69127459)

[***Q2.2: For each of the commands marked with an \*, group them into a shell script so that you can automate execution of the commands. Write the shell script using the Vim text editor. 9***](#_Toc69127460)

[***Q2.3: 11***](#_Toc69127461)

[***Q2.4: Maths Tables Shell Script using Vim 13***](#_Toc69127462)

## Q1: Binary Arithmetic

### Q1.1. Add 11011 to 1011

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Carry** | **1** | **1** | **0** | **1** | **1** | **0** |
| Binary #1 | 0 | 1 | 1 | 0 | 1 | 1 |
| Binary #2 | 0 | 0 | 1 | 0 | 1 | 1 |
| **Total** | **1** | **0** | **0** | **1** | **1** | **0** |

1. 1st Column: 1 + 1 = 0; this results in a Carry of 1;
2. 2nd Column: 1 + 1 = 0; this results in a Carry of 1, then 0 + Carry of 1 (from 1st Column) = 1;
3. 3rd Column: 0 + 0 + Carry of 1 = 1; this results in no Carry;
4. 4th Column: 1 + 1 = 0; this results in a Carry of 1;
5. 5th Column: 1 + 0 = 1 + Carry of 1 = 0; this results in a Carry of 1;
6. 6th Column: 0 + 0 + Carry of 1 = 1; this results in no Carry;

**In conclusion, 11011bin + 1011bin = 100110bin**

The binary calculation converted to decimal numbers is outlined below, and multiply each bit by its corresponding 2x and add all results together to calculate the integer:

11011bin = (1 \* 24) + (1 \* 23) + (0 \* 22) + (1 \* 21) + (1 \* 20) => 27dec

1011bin = (1 \* 23) + (0 \* 22) + (1 \* 21) + (1 \* 20) => 11dec

100110bin = (1 \* 25) + (0 \* 24) + (0 \* 23) + (1 \* 22) + (1 \* 21) + (0 \* 20) => 38dec

**In conclusion, the conversion from binary to decimal confirms 27dec + 11dec = 38dec**

***References:***

<https://www.swarthmore.edu/NatSci/echeeve1/Ref/BinaryMath/BinaryMath.html>

<http://web.math.princeton.edu/math_alive/1/Lab1/BinAddEx3.html>

### Q1.2. Rewrite base-10 numbers as 8-bit two's complement integers: -31, & -59

1. Take the integer -31 as a positive and divide by 2 to calculate remainders:

3110 = 31/2 = 15 **r1**;

15/2 = 7 **r1**;

7/2 = 3 **r1**;

3/2 = 1 **r1**;

1/1 = **1**

**=> 11111**

1. The remainders are organised from the bottom of the above list, with 3 extra zeros to the left, to ensure an 8 bit One’s Complement binary number is established:

3110 = 0001 11112 = (1\*24) + (1\*23) + (1\*22) + (1\*21) + (1\*20) = 31

1. To get the negative integer, invert all bits (0 to 1, 1 to 0), then add 1 to the inverted integer to calculate the 8 bit Two’s Complement:

-3110 = 1110 00002 + 1 = **1110 00012**

1. Take the integer -59 as a positive and divide by 2 to calculate remainders:

5910 = 59/2 = 29 **r1**;

29/2 = 14 **r1**;

14/2 = 7 **r0**;

7/2 = 3 **r1**;

3/2 = 1**r1**;

1/1 = **1**

**=> 110111**

1. The remainders are organised from the bottom of the above list, with 2 extra zeros added to the left, to ensure an 8 bit One’s Complement binary number is established:

5910 = 0011 10112 = (1\*25) + (1\*24) + (1\*23) + (0\*22) + (1\*21) + (1\*20) = 59

1. To get the negative integer, invert all bits (0 to 1, 1 to 0), then add 1 to the inverted integer to calculate the 8 bit Two’s Complement:

-5910 = 1100 01002 + 1 = **1100 01012**

***References:***

<https://binary-system.base-conversion.ro/convert-signed-integers-from-decimal-system-to-binary-two-complement.php>

<https://www.allmath.com/twos-complement.php>

### Q1.3. What does the bit pattern 11101001 represent if you interpret it as an 8-bit two's complement integer?

1. The bit pattern 11101001 indicates a negative integer, as the leftmost bit is equal to 1 (negative).
2. Subtract 1 from the bit pattern:

1110 10012 – 1 = 1110 10002

1. Invert all bits (0 to 1, 1 to 0), and multiply each bit by its corresponding 2x and add all results together to calculate the integer:

0001 01112 = (1\*24) + (0\*23) + (1\*22) + (1\*21) + (1\*20) = **2310**

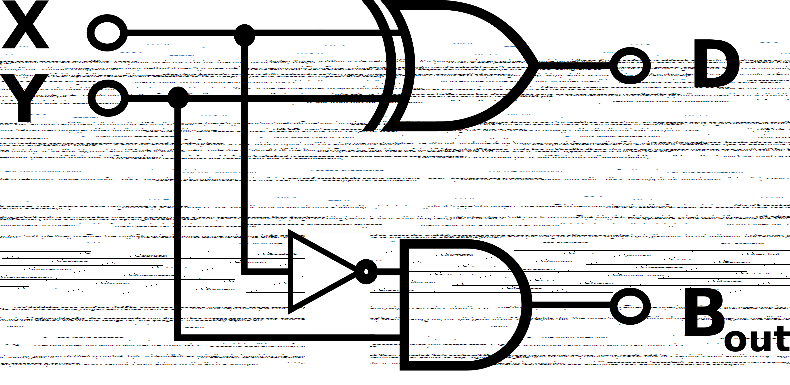
1110 10012 = **-2310**

References:

<https://binary-system.base-conversion.ro/binary-two-complement-converted-to-signed-integer-in-decimal-system.php?binary_two_complement=11101001>

<https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html#fromtwo>

### Q1.4. Draw up the truth table for the circuit below. From observing the result, what function do you think this circuit performs?



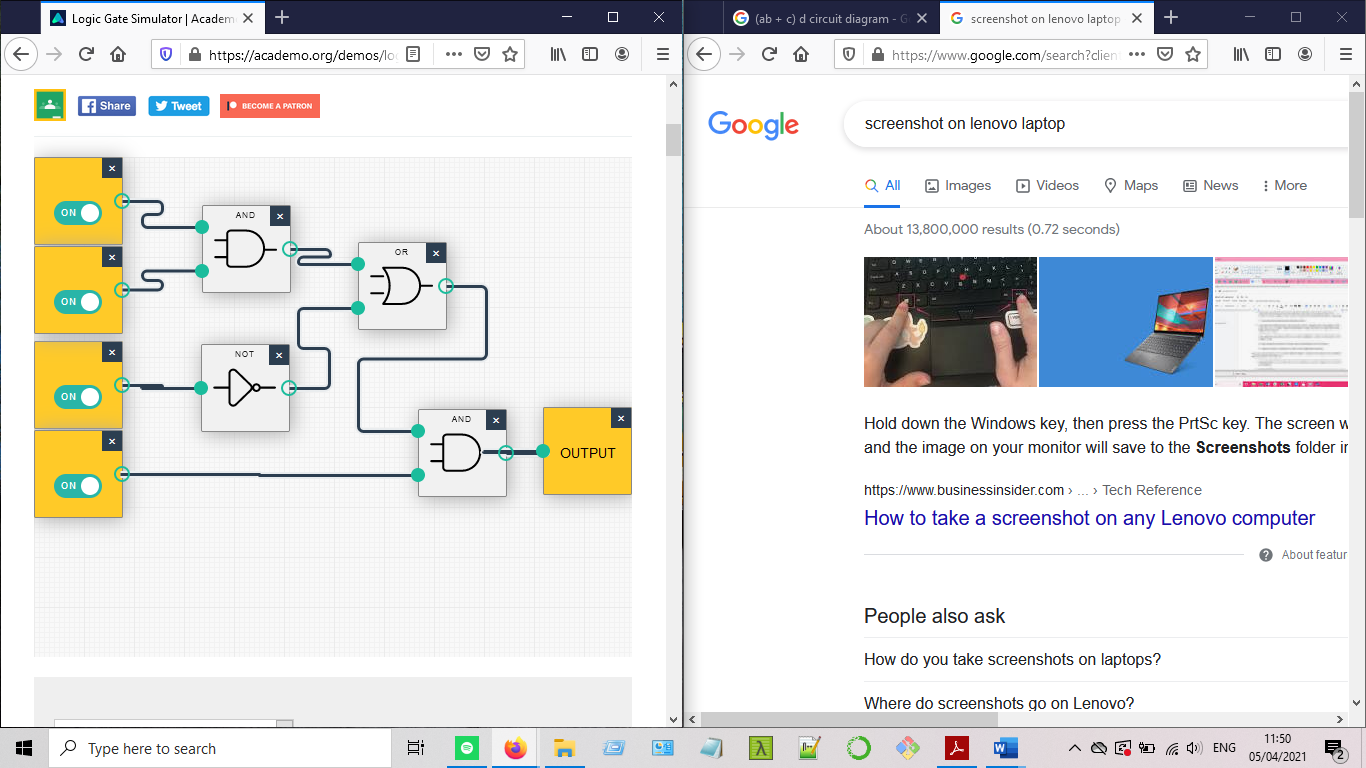
|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **D** | **B** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

1. For Row 1, no input signals are present, therefore no output signals achieved **(0)**.
2. For Row 2, X has no input signal **(0)**, but the NOT Gate returns this as an input signal (1).
   1. Y has an input signal **(1)** which passes through the AND Gate along with X (1).
   2. As both X & Y are 1, this results in B having an output signal **(1)**.
   3. Also, X & Y (0 & 1) pass through the XOR Gate, as X & Y are not the same value, this results in D having an output signal **(1)**.
   4. **This circuit setup is the only scenario that allows both inputs to produce both outputs.**
3. For Row 3, X has an input signal **(1)**, but the NOT Gate returns no input signal (0).
   1. Y has no input signal **(0)** which passes through the AND Gate along with X (0).
   2. As both X & Y are 0, this results in B having no output signal **(0)**.
   3. Also, X & Y (1 & 0) pass through the XOR Gate, as X & Y are not the same value, this results in D having an output signal **(1)**.
4. For Row 4, X has an input signal **(1)**, but the NOT Gate returns no input signal (0).
   1. Y has an input signal **(1)** which passes through the AND Gate along with X (0).
   2. As X & Y are 0 & 1 respectively, and both are not 1, this results in B having no output signal **(0)**.
   3. Also, both X & Y are 1 and pass through the XOR Gate with the same value (1), therefore resulting in D having no output signal **(0)**.

**Reference:**

<https://whatis.techtarget.com/definition/logic-gate-AND-OR-XOR-NOT-NAND-NOR-and-XNOR>

### Q1.5. Draw the circuit diagram for the Boolean logic equation: (AB + C) D



The circuit diagram above consists of the following inputs (A-D) and corresponding Boolean Expressions and Gates, leading to output (E):

1. AND Gate: A \* B = AB
2. NOT Gate: C’
3. OR Gate: AB + C
4. AND Gate: (AB + C) \* D = (AB + C) D

Result: E = (AB + C) D

Reference:

<https://academo.org/demos/logic-gate-simulator/>

<https://slideplayer.com/slide/10782446/>

<https://www.allaboutcircuits.com/textbook/digital/chpt-7/circuit-simplification-examples/>

## Q2: Linux Assignment

### Q2.1. Enter the commands below at the Linux terminal on the AWS VM, and try to interpret the output.

**echo hello world:** outputs the string hello world.

**passwd:** prompts the user to change their current password to a new longer password.

**date\*:** outputs DAY MONTH DATE TIME HH:MM:SS UTC YEAR

**hostname\*:** outputs the IP address of the VM’s host.

**arch\*:** outputs the computer’s architecture (x86\_64).

**uname -a\*:** outputs all information on the VM including user’s credentials and computer specs. Linux ip-172-31-93-200 5.4.0-1038-aws #40-Ubuntu SMP Fri Feb 5 23:50:40 UTC 2021 x86\_64 x86\_64 x86\_64 GNU/Linux

**dmesg | more:** outputs the list of all loaded drivers in kernel (you may need to press q to quit)

**uptime\*:** outputs how long the VM has been up, number of users and load averages.

**whoami\*:** outputs the user’s username.

**who\*:** outputs all currently active users of the VM with their usage timestamps and IP addresses.

**last:** outputs all users of the VM with IP addresses, current/previous login and activity on the VM since the VM’s launch.

**finger\*:** outputs all currently active users of the VM, their idle time (if idle), login time and IP addresses.

**w\*:** outputs all currently active users of the VM, their IP address, login time, idle time (if idle), JCPU (time used by all processes attached to the tty (teletype)), PCPU (time used by the current process), and users’ previous command entered into the VM under the ‘what’ field.

**top\*:** outputs the previous command of the user on top of a list of real-time commands of other users on the VM (you may need to press q to quit).

**echo $SHELL:** outputs the type of shell in the VM (/bin/bash)

**echo {con,pre}{sent,fer}{s,ed}:** outputs and combines strings entered.

**man ls:** outputs the user manual of the ls command (lists contents of current directory). (you may need to press q to quit).

**man who:** outputs the user manual of the who command (show who is logged into the VM). (you may need to press q to quit)

**clear:** clears all commands the user previously entered.

**cal 2000:** outputs the year 2000 calendar.

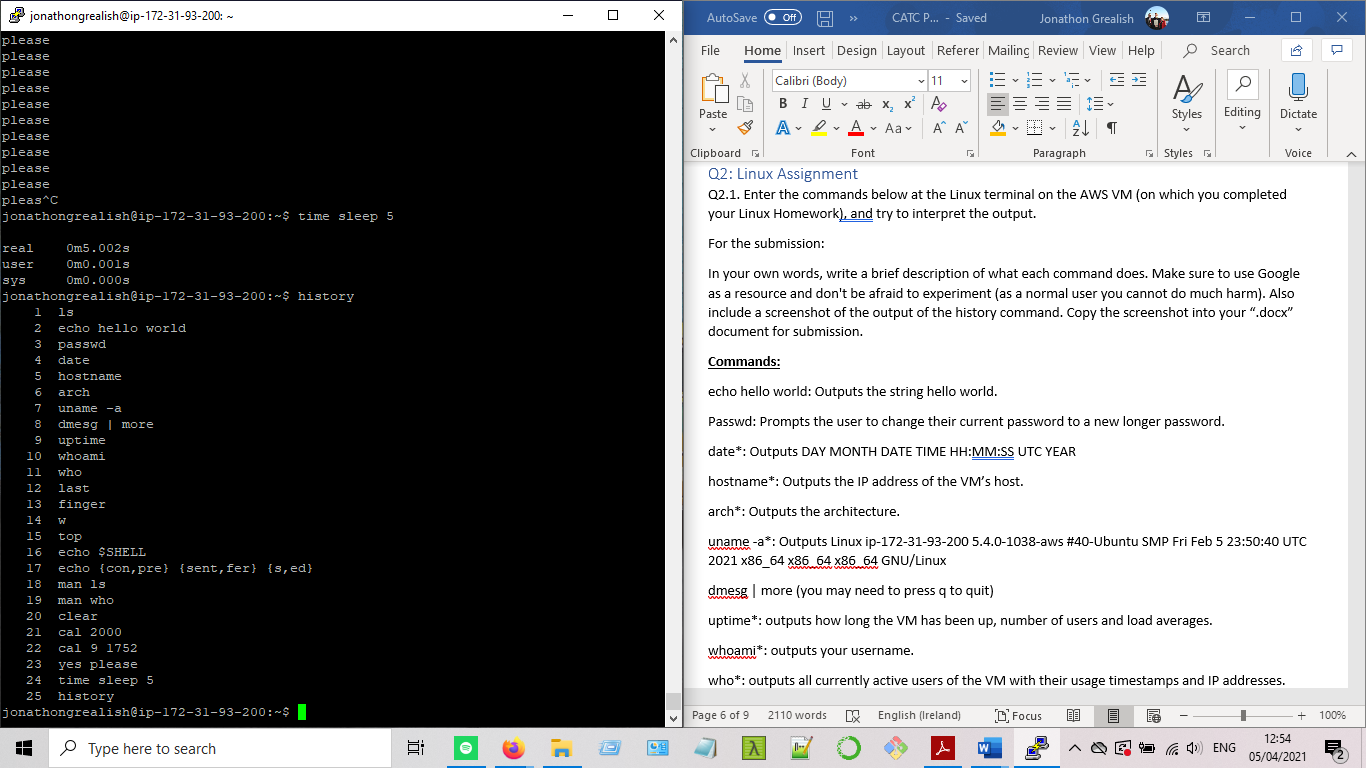
**cal 9 1752** (do you notice anything unusual. Why is this the case?): outputs the calendar of September 1752, where some dates are missing (3rd – 13th September). Google led me to <https://unix.stackexchange.com/questions/17903/is-cal-broken-what-happened-in-september-1752>

where the inventor of the cal command had an error in his code that overwrote 12 extra bytes with zeroes, outputting the strange calendar with missing dates.

**yes please:** outputs please infinitely. (you may need to press Ctrl-c to quit)

**time sleep 5:** outputs real, user and sys times.

**history\*:** outputs a history of the user’s commands in current session:



### Q2.2: For each of the commands marked with an \*, group them into a shell script so that you can automate execution of the commands. Write the shell script using the Vim text editor.

**Date, hostname, arch, uname -a, uptime, whoami, who, finger, w, top, history**

**date**

Mon Apr 5 17:56:27 UTC 2021

**hostname**

ip-172-31-93-200

**architecture**

x86\_64

**uname -a**

Linux ip-172-31-93-200 5.4.0-1038-aws #40-Ubuntu SMP

Fri Feb 5 23:50:40 UTC 2021 x86\_64 x86\_64 x86\_64 GNU/Linux

**uptime**

17:56:27 up 21 days, 20:35, 6 users, load average: 0.00, 0.00, 0.00

**Who Am I?**

jonathongrealish

**Which Users Are Active?**

endalynch pts/0 2021-04-05 00:51 (86.40.219.7)

magdalenamalik pts/1 2021-04-05 10:48 (37.228.254.45)

caioforteribeiro pts/2 2021-04-05 17:36 (37.228.212.144)

lonankeane pts/3 2021-04-05 17:53 (95.44.18.130)

jonathongrealish pts/4 2021-04-05 16:11 (80.233.59.209)

conormccaffrey pts/7 2021-04-05 14:01 (109.79.11.153)

**User Information**

Login Name Tty Idle Login Time Office Office Phone

caioforteribeiro pts/2 2 Apr 5 17:36 (37.228.212.144)

conormccaffrey pts/7 15 Apr 5 14:01 (109.79.11.153)

endalynch pts/0 Apr 5 00:51 (86.40.219.7)

jonathongrealish pts/4 Apr 5 16:11 (80.233.59.209)

lonankeane pts/3 2 Apr 5 17:53 (95.44.18.130)

magdalenamalik pts/1 1:42 Apr 5 10:48 (37.228.254.45)

**User Summary**

17:56:27 up 21 days, 20:35, 6 users, load average: 0.00, 0.00, 0.00

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

endalync pts/0 86.40.219.7 00:51 27.00s 0.15s 0.04s vim student.sh

magdalen pts/1 37.228.254.45 10:48 1:42 0.09s 0.09s -bash

caiofort pts/2 37.228.212.144 17:36 2:29 0.04s 0.04s -bash

lonankea pts/3 95.44.18.130 17:53 2:08 0.03s 0.01s vim

jonathon pts/4 80.233.59.209 16:11 3.00s 0.13s 0.00s w

conormcc pts/7 109.79.11.153 14:01 15:14 0.32s 0.00s lynx

**Real-Time System Information**

top - 17:56:27 up 21 days, 20:35, 6 users, load average: 0.00, 0.00, 0.00

Tasks: 128 total, 1 running, 126 sleeping, 0 stopped, 1 zombie

%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 93.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 s

MiB Mem: 978.6 total, 98.1 free, 283.5 used, 597.0 buff/cache

MiB Swap: 0.0 total, 0.0 free, 0.0 used. 522.4 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

10 root 20 0 0 0 0 S 0.0 0.0 0:35:72 ksoftirqd+

11 root 20 0 0 0 0 I 0.0 0.0 0:44:00 rcu\_sched

12 root rt 0 0 0 0 S 0.0 0.0 0:10:10 migration+

13 root 20 0 0 0 0 S 0.0 0.0 0:00:00 cpuhp/0

14 root 20 0 0 0 0 S 0.0 0.0 0:00:00 kdevtmpfs

15 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 netns

16 root 20 0 0 0 0 S 0.0 0.0 0:00:00 rcu\_tasks+

17 root 20 0 0 0 0 S 0.0 0.0 0:00:00 kauditd

18 root 20 0 0 0 0 S 0.0 0.0 0:00:00 xenbus

19 root 20 0 0 0 0 S 0.0 0.0 0:00:01 xenwatch

20 root 20 0 0 0 0 S 0.0 0.0 0:00:58 khungtaskd

21 root 20 0 0 0 0 S 0.0 0.0 0:00:00 oom\_reaper

22 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 writeback

23 root 20 0 0 0 0 S 0.0 0.0 0:00:02 kcompactd0

24 root 25 5 0 0 0 S 0.0 0.0 0:00:00 ksmd

25 root 39 19 0 0 0 S 0.0 0.0 0:04:20 khugepaged

71 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 kintegrit+

72 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 kblockd

73 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 blkcg\_pun+

74 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 tpm\_dev\_wq

75 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 ata\_sff

76 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 md

77 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 edac-poll+

78 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 devfreq\_wq

79 root rt 0 0 0 0 S 0.0 0.0 0:00:00 watchdogd

82 root 20 0 0 0 0 S 0.0 0.0 0:04:92 kswapd0

83 root 20 0 0 0 0 S 0.0 0.0 0:00:00 ecryptfs-+

85 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 kthrotld

86 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 nvme-wq

87 root 0 -20 0 0 0 I 0.0 0.0 0:00:00 nvme-rese

**History**

1 vim jonathongrealish.sh

2 vimtutor

3 vim jonathongrealish.sh

4 vim jonathongrealish.txt

5 bash jonathongrealish.sh

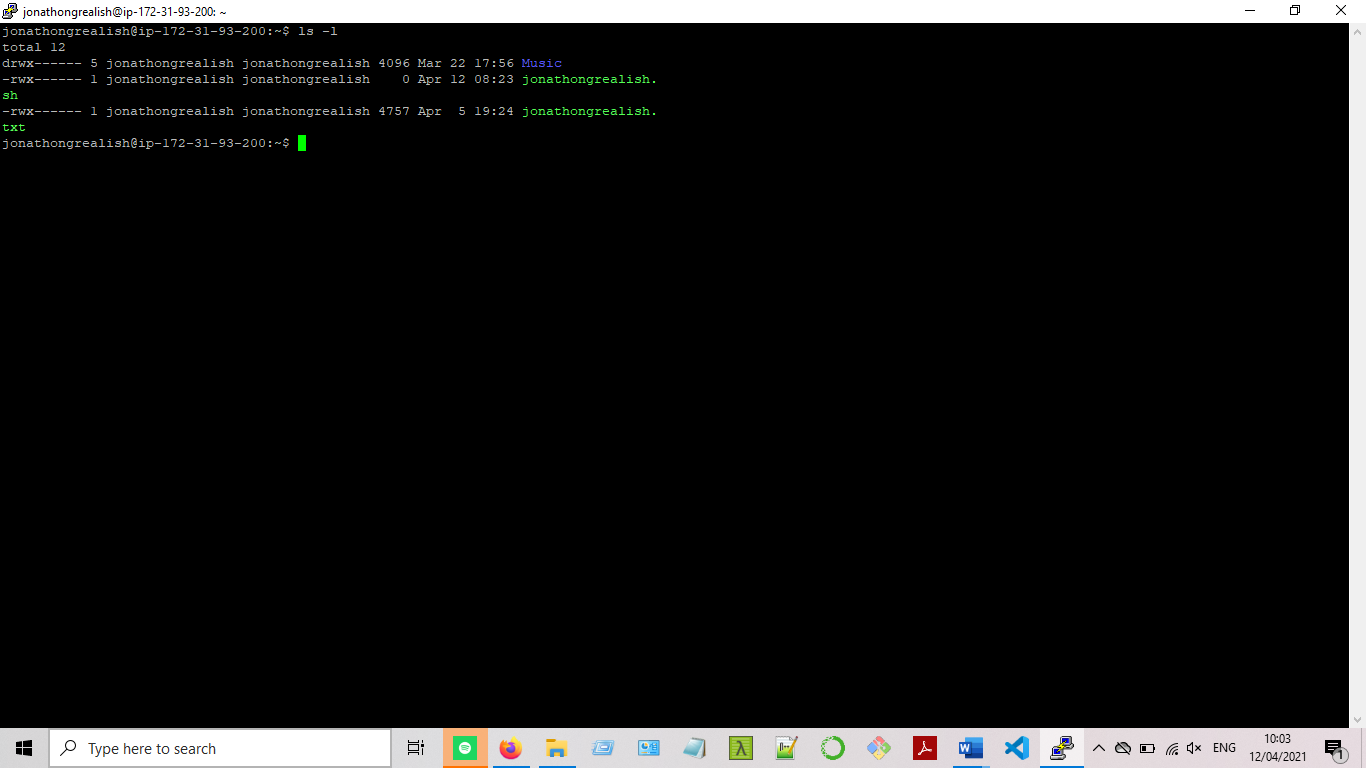
6 vim jonathongrealish.txt

"jonathongrealish.txt" 92L, 4757C 1,1 Top

### Q2.3:

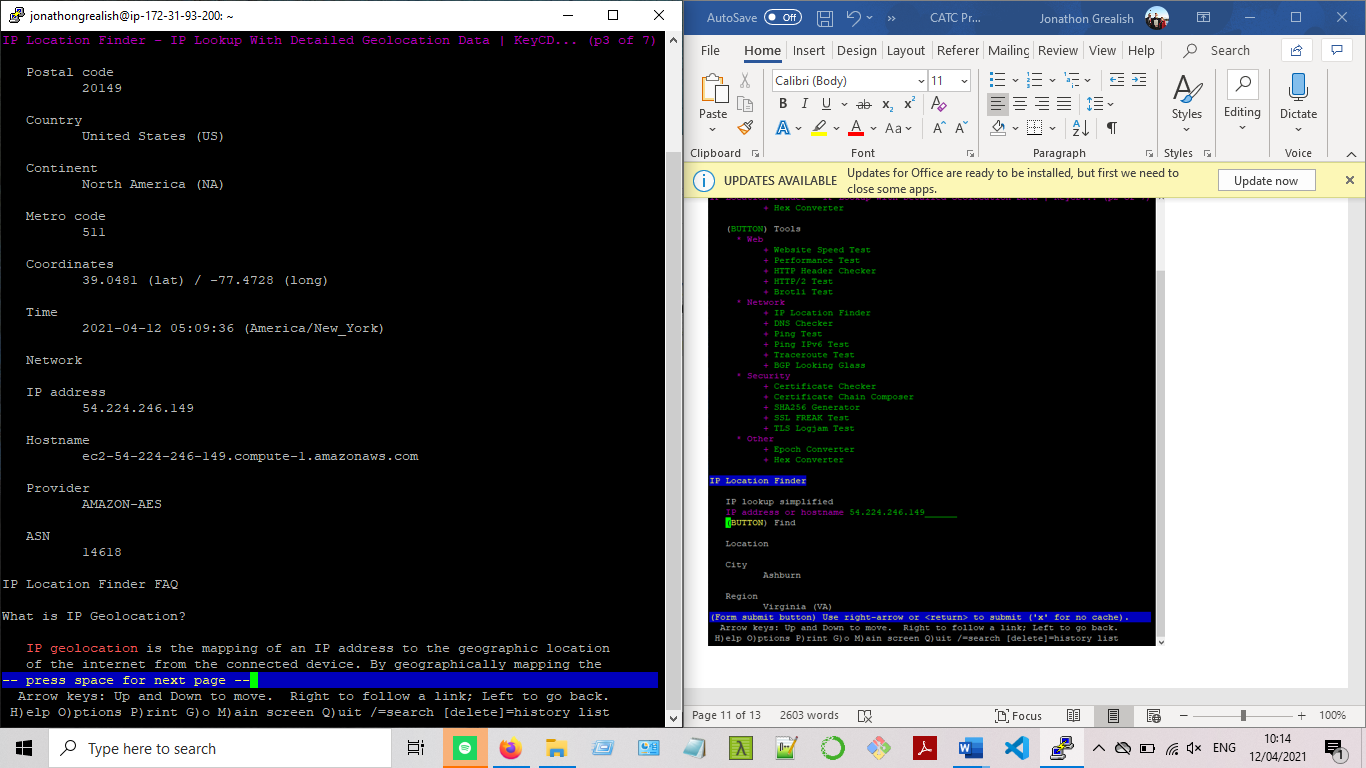
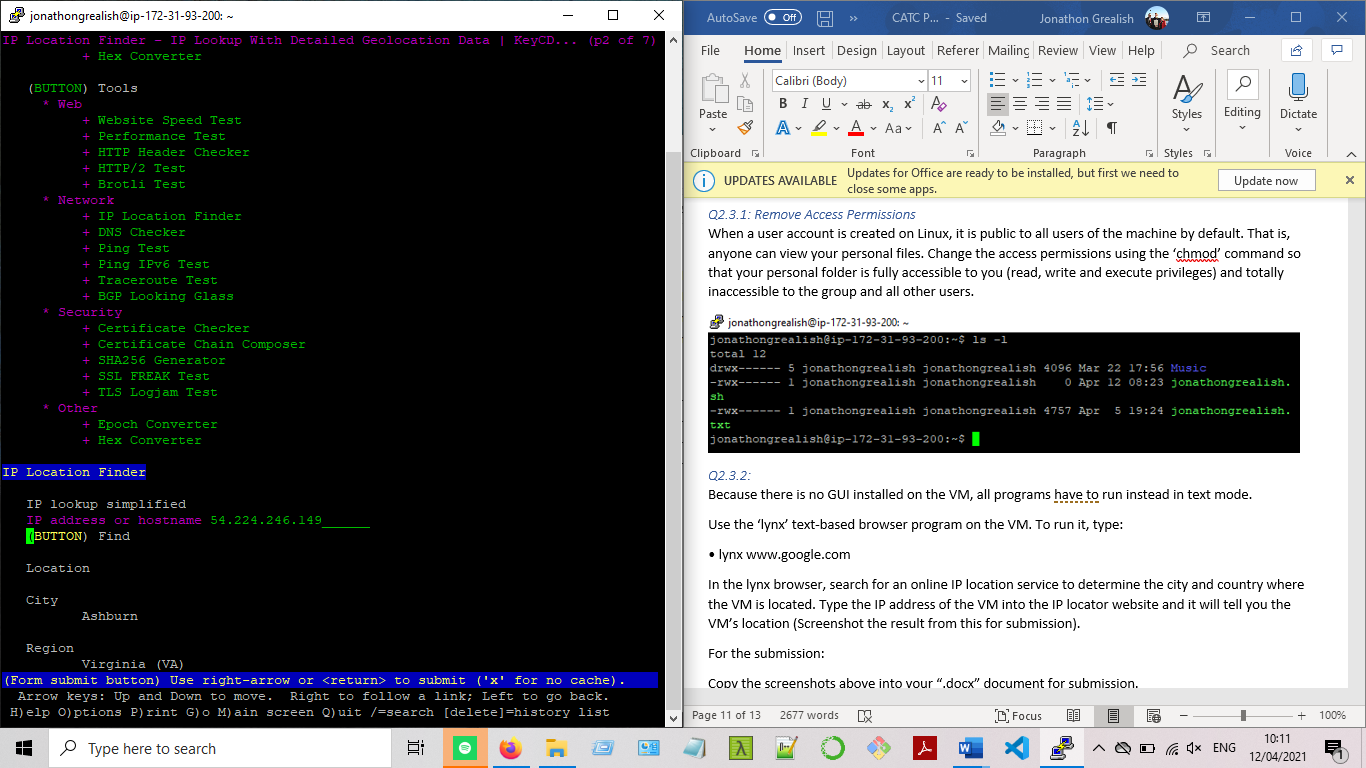
#### Q2.3.1: Remove Access Permissions for All but You

When a user account is created on Linux, it is public to all users of the machine by default. That is, anyone can view your personal files. Change the access permissions using the ‘chmod’ command so that your personal folder is fully accessible to you (read, write and execute privileges) and totally inaccessible to the group and all other users.

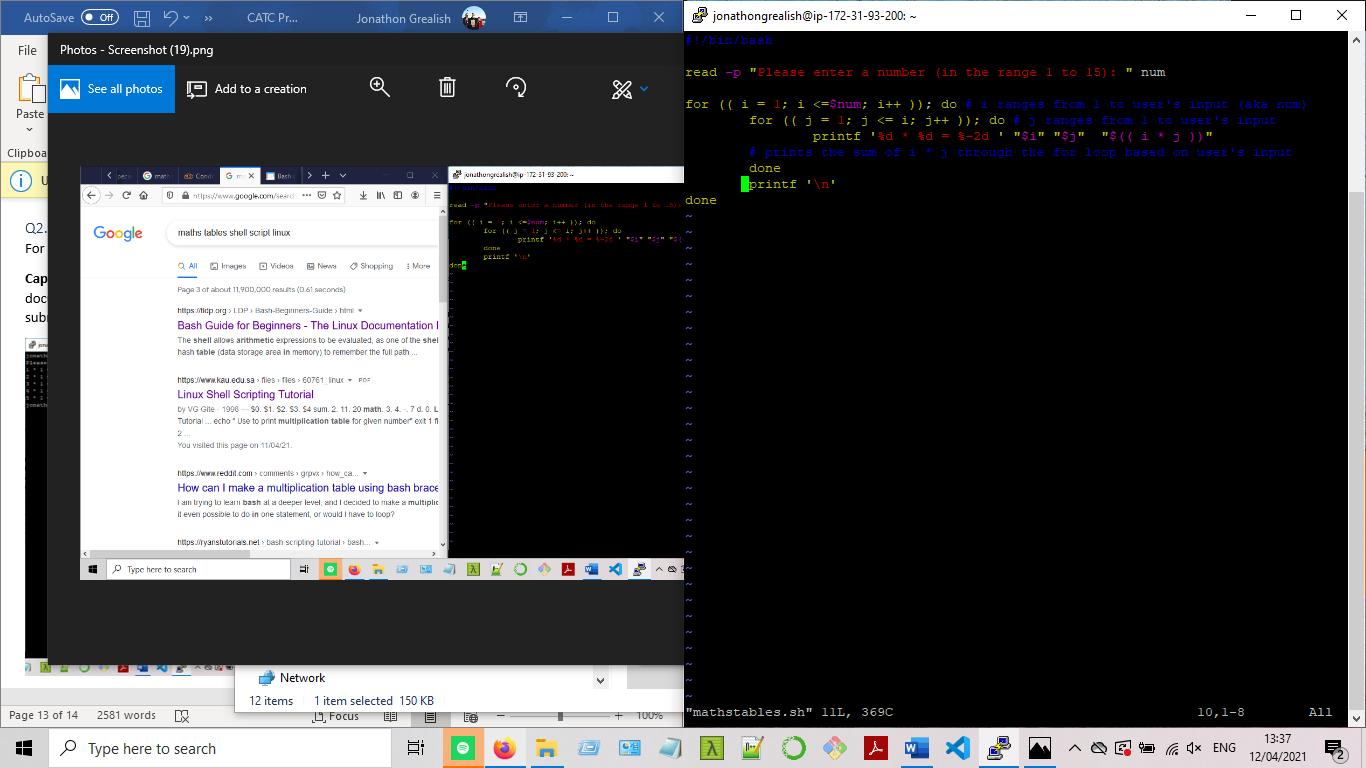


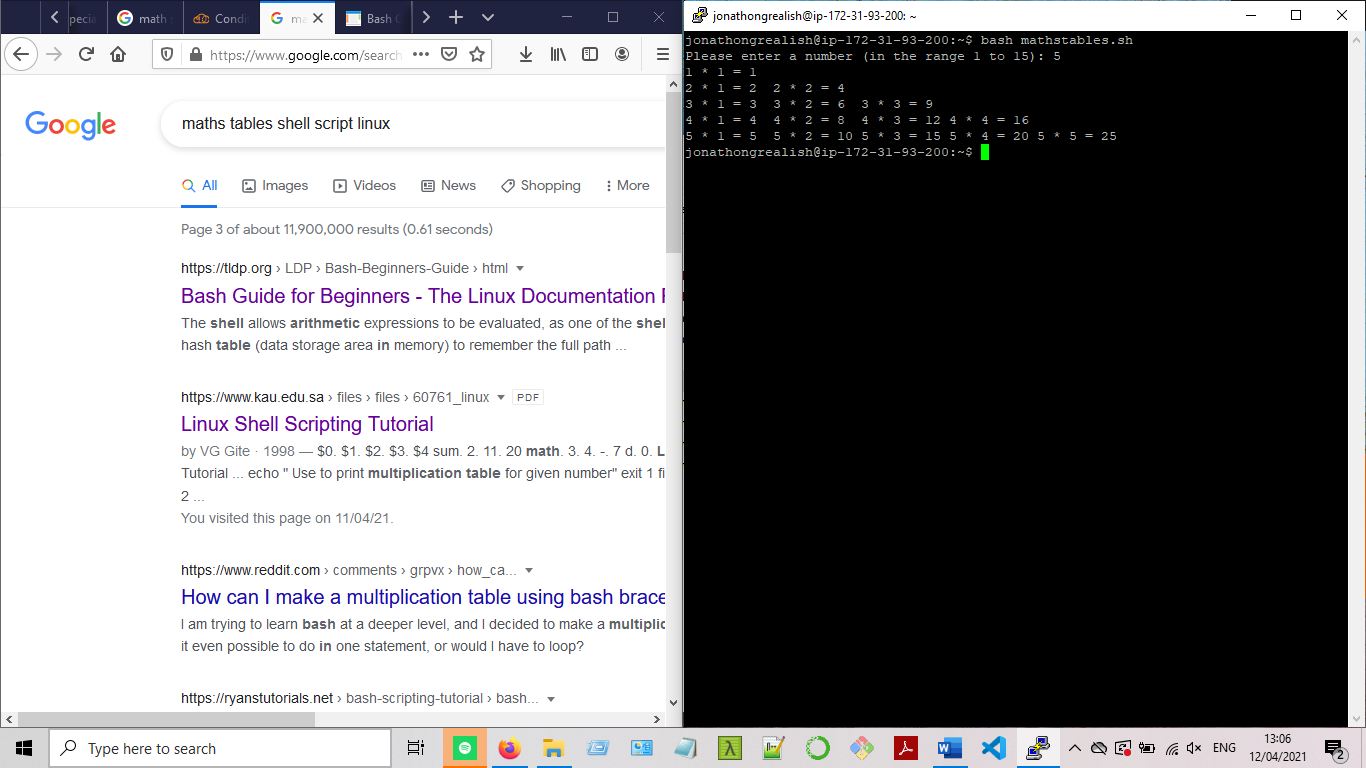
#### Q2.3.2: Use Lynx to find the city and country where the VM is located

**Ashburn, Virginia (VA), USA**



### Q2.4: Maths Tables Shell Script using Vim





References:

<https://www.programmersought.com/article/8933881466/>

<https://unix.stackexchange.com/questions/334124/shell-using-expr-multiplication-table>