

CSc 3320: Systems Programming

Spring 2021

Midterm 1: Total points = 100

Assigned: 26th Feb 2021: 12.01 PM

Submission Deadline: 2nd Mar 2021: 12.01 PM

(No extensions. If your submission is not received by this time then it will NOT be accepted.)

Submission instructions:

1. Create a Google doc for your submission.
2. Start your responses from page 2 of the document and copy these instructions on page 1.
3. Fill in your name, campus ID and panther # in the fields provided. If this information is missing TWO POINTS WILL BE DEDUCTED.
4. Keep this page 1 intact. If this *submissions instructions* page is missing in your submission TWO POINTS WILL BE DEDUCTED.
5. Start your responses to each QUESTION on a new page.
6. If you are being asked to write code copy the code into a separate txt file and submit that as well. The code should be executable. E.g. if asked for a C script then provide myfile.c so that we can execute that script. In your answer to the specific question, provide the steps on how to execute your file (like a ReadMe).
7. If you are being asked to test code or run specific commands or scripts, provide the evidence of your outputs through a screenshot and/or screen video-recordings and copy the same into the document.
8. Upon completion, download a .PDF version of the google doc document and submit the same along with all the supplementary files (videos, pictures, scripts etc).

Full Name: Tracy Michaels

Campus ID: tmichaels1

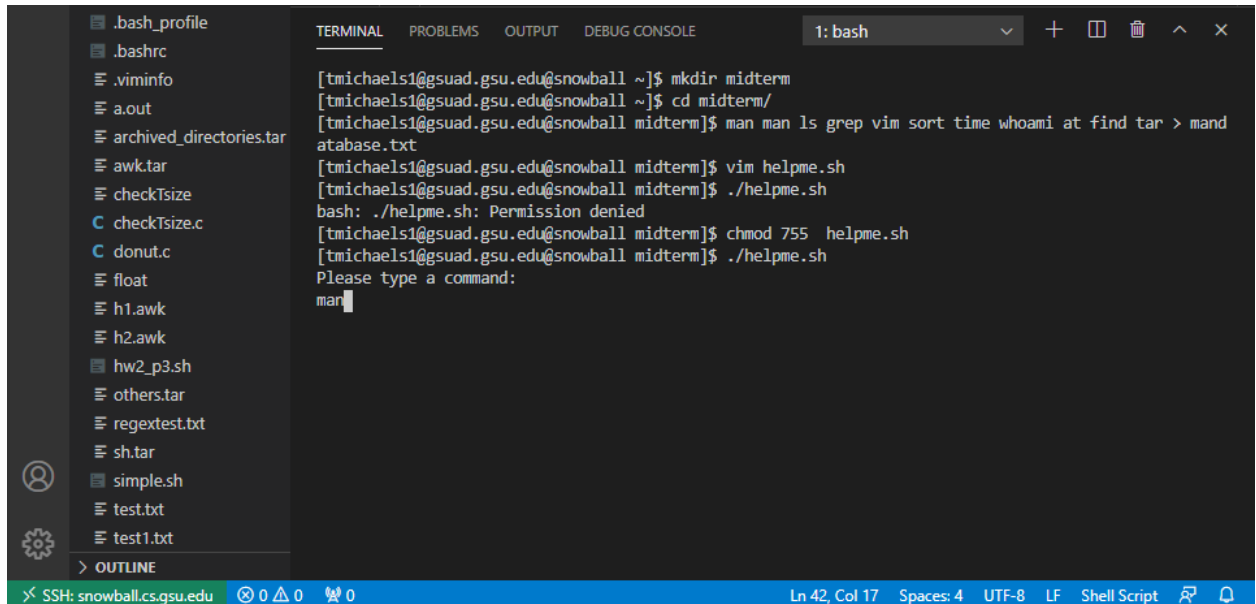
Panther #: 002430918

Questions 1-5 are 20pts each

1. Pick any of your 10 favourite unix commands. For each command run the *man* command and copy the text that is printed into a *mandatabase.txt*. Write a shell script *helpme.sh* that will ask the user to type in a command and then print the manual's text associated with that corresponding command. If the command the user types is not in the database then the script must print *sorry, I cannot help you*
2. On your computer open your favourite Wikipedia page. Copy the text from that page into a text file **myexamfile.txt** and then copy that file to a directory named **midterm** (use *mkdir* to create the directory if it doesn't exist) in your snowball server home directory (use any FTP tool such as Putty or Filezilla to copy the file from your computer to the remote snowball server machine: see Lab 6). Write a shell script that will find the number of occurrences of a particular keyword typed by the user. Present evidence of your testing with at least 5 trials (different keywords each time)
3. Write a shell script to find files in a directory hierarchy (e.g. your home directory) that have not been accessed for N days and compress them. Here N is a parameter and the user will be asked for that input as the first step of the script execution.
4. Build a phone-book utility that allows you to access and modify an alphabetical list of names, addresses and telephone numbers. Use utilities such as *awk* and *sed*, to maintain and edit the file of phone-book information. The user (in this case, you) must be able to read, edit, and delete the phone book contents. The permissions for the phone book database must be such that it is inaccessible to anybody other than the user.
5.
 - A. Write a C script that will compute the factorial of a given number (positive integer).

B. Write a C script to find the new integer value of an original integer when it is bit-shifted left by 3 bits and added to its complement (one's complement of the original integer).
(Note: You can manually type in the binary representation of the original integer)
(10 bonus points for writing the C script to convert the integer to binary and vice-versa)
(10 bonus points for writing a shell script that will execute both the C scripts from above for a given integer number)

- 1) Used command `man man ls grep vim sort time whoami at find tar > mandatabase.txt` to get the information from each manuals and store them in a text
Created `helpme.sh` that prompts for a command and if that command is in the `mandatabase.txt` it will print the man pages for that command



```
.bash_profile
.bashrc
.viminfo
a.out
archived_directories.tar
awk.tar
checkTsize
checkTsize.c
donut.c
float
h1.awk
h2.awk
hw2_p3.sh
others.tar
regextest.txt
sh.tar
simple.sh
test.txt
test1.txt
> OUTLINE

TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE  1: bash
[tmichaels1@gsuad.gsu.edu@snowball ~]$ mkdir midterm
[tmichaels1@gsuad.gsu.edu@snowball ~]$ cd midterm/
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ man man ls grep vim sort time whoami at find tar > mand
atabase.txt
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ vim helpme.sh
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./helpme.sh
bash: ./helpme.sh: Permission denied
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ chmod 755 helpme.sh
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./helpme.sh
Please type a command:
man
```

The image shows a Visual Studio Code editor window with the title bar "helpme.sh - tmichaels1 [SSH: snowball.cs.gsu.edu] - Visual Studio Code". The interface includes a menu bar (File, Edit, Selection, View, Go, Run, Terminal, Help), a sidebar with icons for Explorer, Search, Source Control, Run and Debug, and Extensions, and a main editor area.

The Explorer sidebar on the left shows a file tree for the "midterm" directory. The files listed are: foo.class, foo.java, foo.sh, hello, hello.c, hello.sh, myName, myName.c, Result, Result.txt, helpme.sh (selected), mandatabase.txt, others, public, sandbox, sh, test, txt, .bash_history, .bash_logout, .bash_profile, .bashrc, .viminfo, a.out, archived_directories.tar, awk.tar, checkTsize, checkTsize.c, donut.c, float, h1.awk, h2.awk, hw2_p3.sh, others.tar, regexTest.txt, sh.tar, simple.sh, test.txt, and test1.txt.

The main editor area displays the "helpme.sh" script. The script is a shell script that prompts the user to enter a command and then searches for that command in a file named "mandatabase.txt". The script uses a case statement to handle different commands: man, ls, grep, vim, sort, time, whoami, at, find, tar, and *. If the command is not found, it echoes "Sorry, I cannot help you".

```
1  #!/bin/bash
2  #prompts user to enter a command
3  #if command is present in mandatabase.txt return relevant information
4  #must have mandatabase.txt in same directory
5
6  echo "Please type a command:"
7
8  read userIn
9
10 case $userIn in
11     man)
12         sed -rn '/MAN\(\1\)$/,/MAN\(\1\)$/' ./mandatabase.txt
13         ;;
14     ls)
15         sed -rn '/LS\(\1\)$/,/LS\(\1\)$/' ./mandatabase.txt
16         ;;
17     grep)
18         sed -rn '/GREP\(\1\)$/,/GREP\(\1\)$/' ./mandatabase.txt
19         ;;
20     vim)
21         sed -rn '/VIM\(\1\)$/,/VIM\(\1\)$/' ./mandatabase.txt
22         ;;
23     sort)
24         sed -rn '/SORT\(\1\)$/,/SORT\(\1\)$/' ./mandatabase.txt
25         ;;
26     time)
27         sed -rn '/TIME\(\1\)$/,/TIME\(\1\)$/' ./mandatabase.txt
28         ;;
29     whoami)
30         sed -rn '/WHOAMI\(\1\)$/,/WHOAMI\(\1\)$/' ./mandatabase.txt
31         ;;
32     at)
33         sed -rn '/AT\(\1\)$/,/AT\(\1\)$/' ./mandatabase.txt
34         ;;
35     find)
36         sed -rn '/FIND\(\1\)$/,/FIND\(\1\)$/' ./mandatabase.txt
37         ;;
38     tar)
39         sed -rn '/TAR\(\1\)$/,/TAR\(\1\)$/' ./mandatabase.txt
40         ;;
41     *)
42         echo 'Sorry, I cannot help you'
43         ;;
44 esac
```

The status bar at the bottom shows "SSH: snowball.cs.gsu.edu", "0 errors, 0 warnings, 0 info", "Ln 44, Col 5", "Spaces: 4", "UTF-8", "LF", "Shell Script", and a search icon.

File Edit Selection View Go Run Terminal Help helpme.sh - tmichaels1 [SSH: snowball.cs.gsu.edu] - Visual Studio Code

EXPLORER
OPEN EDITORS
TMICHAELS1 [SSH: SNOW...

- foo.class
- foo.java
- foo.sh
- hello
- hello.c
- hello.sh
- myName
- myName.c
- Result
- Result.txt
- midterm
 - helpme.sh
 - mandatabase.txt
- others
- public
- sandbox
- sh
- test
- txt
- .bash_history
- .bash_logout
- .bash_profile
- .bashrc
- .viminfo
- a.out
- archived_directories.tar
- awk.tar
- checkTsize
- checkTsize.c
- donut.c
- float
- h1.awk
- h2.awk
- hw2_p3.sh
- others.tar
- regextest.txt
- sh.tar
- simple.sh
- test.txt
- test1.txt

- OUTLINE

helpme.sh X mandatabase.txt
midterm > helpme.sh

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: bash
[tmichaels1@gsuad.gsu.edu@snowball midterm]\$./helpme.sh
Please type a command:
man
MAN(1) Manual pager utils MAN(1)

NAME
man - an interface to the on-line reference manuals

SYNOPSIS
man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M path] [-S list] [-e extension] [-i|-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justification] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page ...] ...
man -k [apropos options] regexp ...
man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
man -f [whatis options] page ...
man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager] [-r prompt] [-7] [-E encoding] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
man -w|-W [-C file] [-d] [-D] page ...
man -c [-C file] [-d] [-D] page ...
man [-?V]

DESCRIPTION
man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections, following a pre-defined order and to show only the first page found, even if page exists in several sections.

The table below shows the section numbers of the manual followed by the types of pages they contain.

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions eg /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]

A manual page consists of several sections.

Conventional section names include NAME, SYNOPSIS, CONFIGURATION, DESCRIPTION, OPTIONS,

SSH: snowball.cs.gsu.edu 0 0 0 Ln 15, Col 22 (2 selected) Spaces: 4 UTF-8 LF Shell Script

2)asks user for a word and searches myexamplyfile for that word and returns the number of times it occurs in that file

The screenshot shows a Visual Studio Code editor window with the title bar "countoccurrence.sh - tmichaels1 [SSH: snowball.cs.gsu.edu] - Visual Studio...". The interface includes a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The Explorer view shows a file tree for "TMICHAELS1 [SSH: SNOW...]" with folders like ".conrig", ".vscode-server", "awk", "csc3320", "hw3", "Lab3", "Lab4", "lab6", and "midterm". The "midterm" folder is expanded, showing files like "countoccurrence.sh", "helpme.sh", "mandatabase.txt", "myexamfile.txt", and others. The main editor area has three tabs: "helpme.sh", "myexamfile.txt", and "countoccurrence.sh". The "countoccurrence.sh" tab is active, displaying a shell script with 18 lines of code. The script starts with a shebang, a comment about counting word occurrences, and uses a loop to read user input and count occurrences in "myexamfile.txt" using grep and wc. The terminal at the bottom shows the script being executed with various inputs: "buffalo" (96 occurrences), "bison" (17 occurrences), "wikipedia" (1 occurrence), "asdf" (0 occurrences), and "homonyms" (1 occurrence). The terminal output is as follows:

```
midterm > countoccurrence.sh
1  #!/bin/bash
2  #count the number of times a word entered by the user occurs in a wikiped
3  #must have myexamfile.txt in same directory
4
5  echo 'Please enter a word:'
6
7  read userIn
8
9  num=$(grep -owi ${userIn} ./myexamfile.txt | wc -l)
10
11 if [ $num \> 0 ]
12 then
13     echo "There are ${num} occurrences of '${userIn}' in this file"
14 else
15     echo "There are no occurrences of '${userIn}' in this file"
16 fi
17
18
```

Terminal Output:

```
1: bash
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./countoccurrence.sh
Please enter a word:
buffalo
There are 96 occurrences of 'buffalo' in this file
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./countoccurrence.sh
Please enter a word:
bison
There are 17 occurrences of 'bison' in this file
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./countoccurrence.sh
Please enter a word:
wikipedia
There are 1 occurrences of 'wikipedia' in this file
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./countoccurrence.sh
Please enter a word:
asdf
There are no occurrences of 'asdf' in this file
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./countoccurrence.sh
Please enter a word:
homonyms
There are 1 occurrences of 'homonyms' in this file
[tmichaels1@gsuad.gsu.edu@snowball midterm]$
```

The status bar at the bottom indicates the current file is "countoccurrence.sh" at line 18, column 1, with 4 spaces, UTF-8 encoding, LF line endings, and Shell Script language.

File Edit Selection View Go Run Terminal Help myexamfile.txt - tmichaels1 [SSH: snowball.cs.gsu.edu] - Visual Studio C...

EXPLORER

OPEN EDITORS

TMICHAELS1 [SSH: SNOW...]

- > .conig
- > .vscode-server
- > awk
- > csc3320
- > hw3
- > Lab3
- > Lab4
- > lab6
 - checkError_correcte...
 - checkError.sh
 - foo.class
 - foo.java
 - foo.sh
 - hello
 - hello.c
 - hello.sh
 - myName
 - myName.c
 - Result
 - Result.txt
- midterm
 - countoccurrence.sh
 - helpme.sh
 - mandatabase.txt
 - myexamfile.txt
- > others
- > public
- > sandbox
- > sh
- > test
- > txt
- .bash_history
- .bash_logout
- .bash_profile
- .bashrc
- .viminfo
- a.out
- archived_directories.tar
- awk.tar
- checkTsize
- checkTsize.c

midterm > myexamfile.txt

```
1 Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo
2 From Wikipedia, the free encyclopedia
3 Jump to navigationJump to search
4
5 Simplified parse tree
6
7 S = sentence
8 NP = noun phrase
9 RC = relative clause
10 VP = verb phrase
11 PN = proper noun
12 N = noun
13 V = verb
14 "Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo" is a gram
15
16 The sentence employs three distinct meanings of the word buffalo:
17
18 as a proper noun to refer to a specific place named Buffalo, the city of B
19 as a verb (uncommon in regular usage) to buffalo, meaning "to bully, haras
20 as a noun to refer to the animal, bison (often called buffalo in North Ame
21 An expanded form of the sentence which preserves the original word order i
22
23
24 Contents
25 1 Sentence construction
26 1.1 Usage
27 2 Origin
28 3 See also
29 4 References
30 5 External links
31 Sentence construction
32
33 Reed-Kellogg diagram of the sentence
34 The sentence is unpunctuated and uses three different readings of the word
35
36 a. a city named Buffalo. This is used as a noun adjunct in the sentence;
37 n. the noun buffalo (American bison), an animal, in the plural (equivalent
38 v. the verb "buffalo" meaning to outwit, confuse, deceive, intimidate, or l
39 The sentence is syntactically ambiguous; however, one possible parse (mark
40
41 | Buffaloa buffalon Buffaloa buffalon buffalov buffalov Buffaloa buffal
42
43 When grouped syntactically, this is equivalent to: [(Buffalonian bison) (B
44
45 The sentence uses a restrictive clause, so there are no commas, nor is the
```

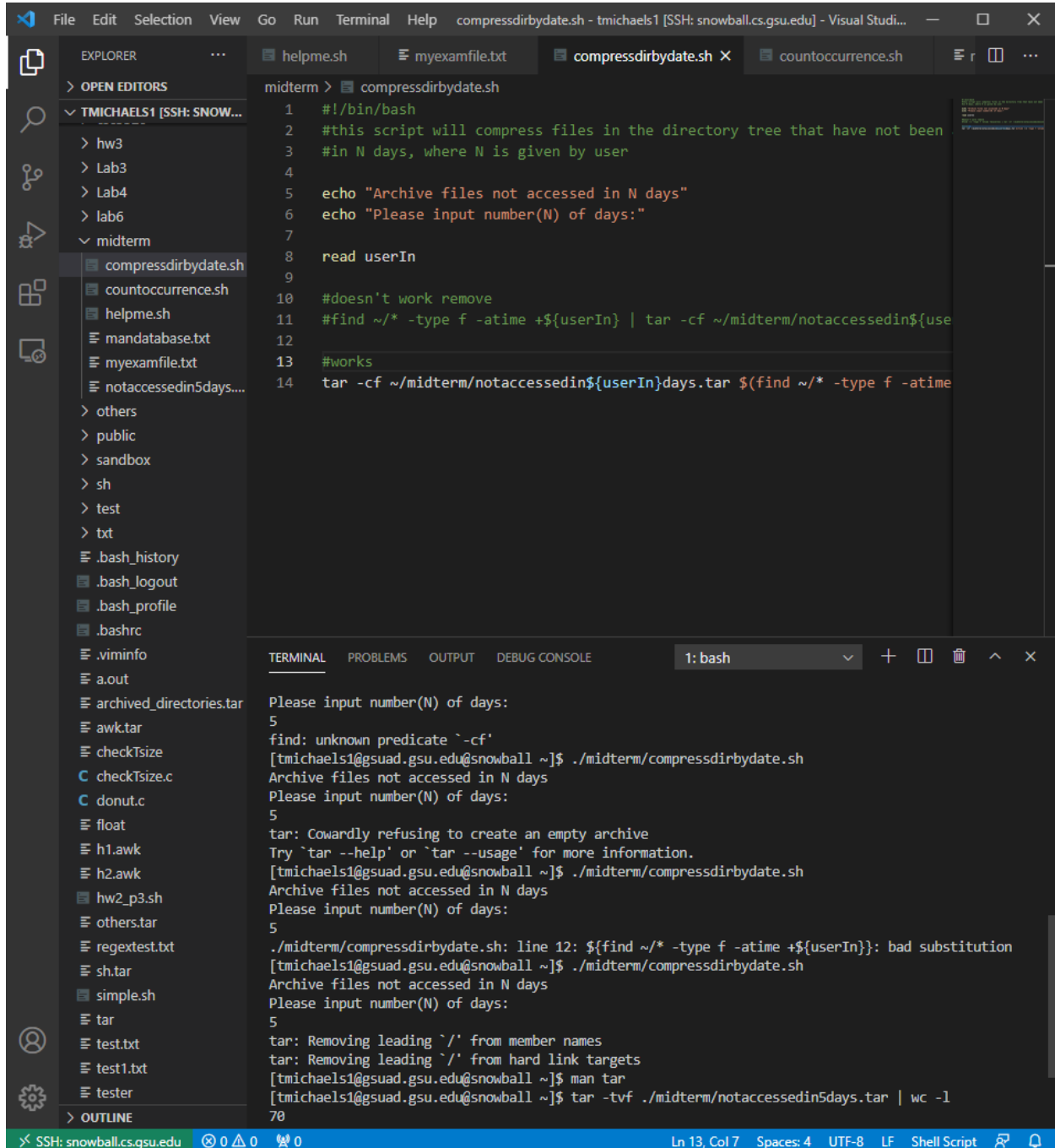
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: bash

There are 1 occurrences of 'homonyms' in this file
[tmichaels1@gsuad.gsu.edu@snowball midterm]\$

SSH: snowball.cs.gsu.edu 0 0 0 Ln 69, Col 208 Spaces: 4 UTF-8 LF Plain Text

3) compresses all files that have not been accessed in the last N days which is supplied by the user

In the output in the screenshot I counted the number of files that were saved in the .tar file and compared that to the number of files that have not been accessed in the last 5 days and they match so I'm positive that it got them all.



The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like `compressdirbydate.sh`, `countoccurrence.sh`, `helpme.sh`, `mandatabase.txt`, `myexamfile.txt`, `notaccessedin5days...`, `others`, `public`, `sandbox`, `sh`, `test`, `txt`, `.bash_history`, `.bash_logout`, `.bash_profile`, `.bashrc`, `.viminfo`, `a.out`, `archived_directories.tar`, `awk.tar`, `checkTsize`, `checkTsize.c`, `donut.c`, `float`, `h1.awk`, `h2.awk`, `hw2_p3.sh`, `others.tar`, `regextest.txt`, `sh.tar`, `simple.sh`, `tar`, `test.txt`, `test1.txt`, and `tester`. The code editor shows the script `compressdirbydate.sh` with the following content:

```
1 #!/bin/bash
2 #this script will compress files in the directory tree that have not been
3 #in N days, where N is given by user
4
5 echo "Archive files not accessed in N days"
6 echo "Please input number(N) of days:"
7
8 read userIn
9
10 #doesn't work remove
11 #find ~/* -type f -atime +${userIn} | tar -cf ~/midterm/notaccessedin${use
12
13 #works
14 tar -cf ~/midterm/notaccessedin${userIn}days.tar $(find ~/* -type f -atime
```

The terminal output shows the execution of the script with the following messages:

```
1: bash
Please input number(N) of days:
5
find: unknown predicate `~cf'
[tmichaels1@gsuad.gsu.edu@snowball ~]$ ./midterm/compressdirbydate.sh
Archive files not accessed in N days
Please input number(N) of days:
5
tar: Cowardly refusing to create an empty archive
Try `tar --help' or `tar --usage' for more information.
[tmichaels1@gsuad.gsu.edu@snowball ~]$ ./midterm/compressdirbydate.sh
Archive files not accessed in N days
Please input number(N) of days:
5
./midterm/compressdirbydate.sh: line 12: ${find ~/* -type f -atime +${userIn}}: bad substitution
[tmichaels1@gsuad.gsu.edu@snowball ~]$ ./midterm/compressdirbydate.sh
Archive files not accessed in N days
Please input number(N) of days:
5
tar: Removing leading `/' from member names
tar: Removing leading `/' from hard link targets
[tmichaels1@gsuad.gsu.edu@snowball ~]$ man tar
[tmichaels1@gsuad.gsu.edu@snowball ~]$ tar -tvf ./midterm/notaccessedin5days.tar | wc -l
70
```

The status bar at the bottom shows the file path `SSH: snowball.cs.gsu.edu`, the file encoding `UTF-8`, and the line and column numbers `Ln 13, Col 7`.

4)

```
File Edit Selection View Go Run Terminal Help phonebook.sh - tmichaels1 [SSH: snowball.cs.gsu.edu] - Visual Studio C...
EXPLORER
> OPEN EDITORS
TMICHAELS1 [SSH: SNOW...
> .cache
> .config
> .vscode-server
> awk
> csc3320
> hw3
> Lab3
> Lab4
> lab6
> midterm
  compressdirbydate.sh
  countoccurrence.sh
  helpme.sh
  mandatabase.txt
  myexamfile.txt
  notaccessedin5days....
  phonebook.sh
  phonebookdb.txt
> others
> public
> sandbox
> sh
> test
> txt
  .bash_history
  .bash_logout
  .bash_profile
  .bashrc
  .viminfo
  a.out
  archived_directories.tar
  awk.tar
  checkTsize
  checkTsize.c
  donut.c
  float
  h1.awk
  h2.awk
  hw2_p3.sh
  others.tar
> OUTLINE

midterm > phonebook.sh
1  #!/bin/bash
2  #this script creates and maintains a phonebook that can be edited by user
3
4  #creates files if it doesn't already exists
5  PBDB=./phonebookdb.txt
6  if [ ! -f $PBDB ]
7  then
8      > phonebookdb.txt
9  fi
10
11  chmod 700 $PBDB
12
13  #prompt user
14  echo "Please select an option"
15  echo "type corresponding number below to select"
16  echo "1) create new contact"
17  echo "2) edit existing entry"
18  echo "3) remove entry"
19  echo "4) view entries"
20
21  read userSelection
22
23  case $userSelection in
24      1)
25          echo "First Name:"
26          read firstName
27          echo "Last Name:"
28          read lastName
29          echo "Address:"
30          read address
31          echo "Phone:"
32          read phone
33          echo "Adding contact: $firstName $lastName, $address, $phone"
34          echo "Contact added successfully"
35          ;;
36      2)
37          echo "Please select an option"
38          echo "type corresponding number below to select"
39          echo "1) create new contact"
40          echo "2) edit existing entry"
41          echo "3) remove entry"
42          echo "4) view entries"
43          read userSelection
44          ;;
45      3)
46          echo "Please select an option"
47          echo "type corresponding number below to select"
48          echo "1) create new contact"
49          echo "2) edit existing entry"
50          echo "3) remove entry"
51          echo "4) view entries"
52          read userSelection
53          ;;
54      4)
55          echo "Viewing contacts"
56          cat $PBDB
57          ;;
58      *)
59          echo "Invalid option"
60          ;;
61  esac
```

```
1: bash
[tmichaels1@gsuad.gsu.edu@snowball ~]$ cd midterm/
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./phonebook.sh
Please select an option
type corresponding number below to select
1) create new contact
2) edit existing entry
3) remove entry
4) view entries
1
First Name:
Tracy
Last Name:
Michaels
Address:
433 Eagle Tiff Dr.
Phone:
6789946252
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./phonebook.sh
Please select an option
type corresponding number below to select
1) create new contact
2) edit existing entry
3) remove entry
4) view entries
4
1 Tracy, Michaels, 433 Eagle Tiff Dr., 6789946252
[tmichaels1@gsuad.gsu.edu@snowball midterm]$
```

SSH: snowball.cs.gsu.edu 0 0 0 Ln 71, Col 1 Spaces: 4 UTF-8 LF Shell Script

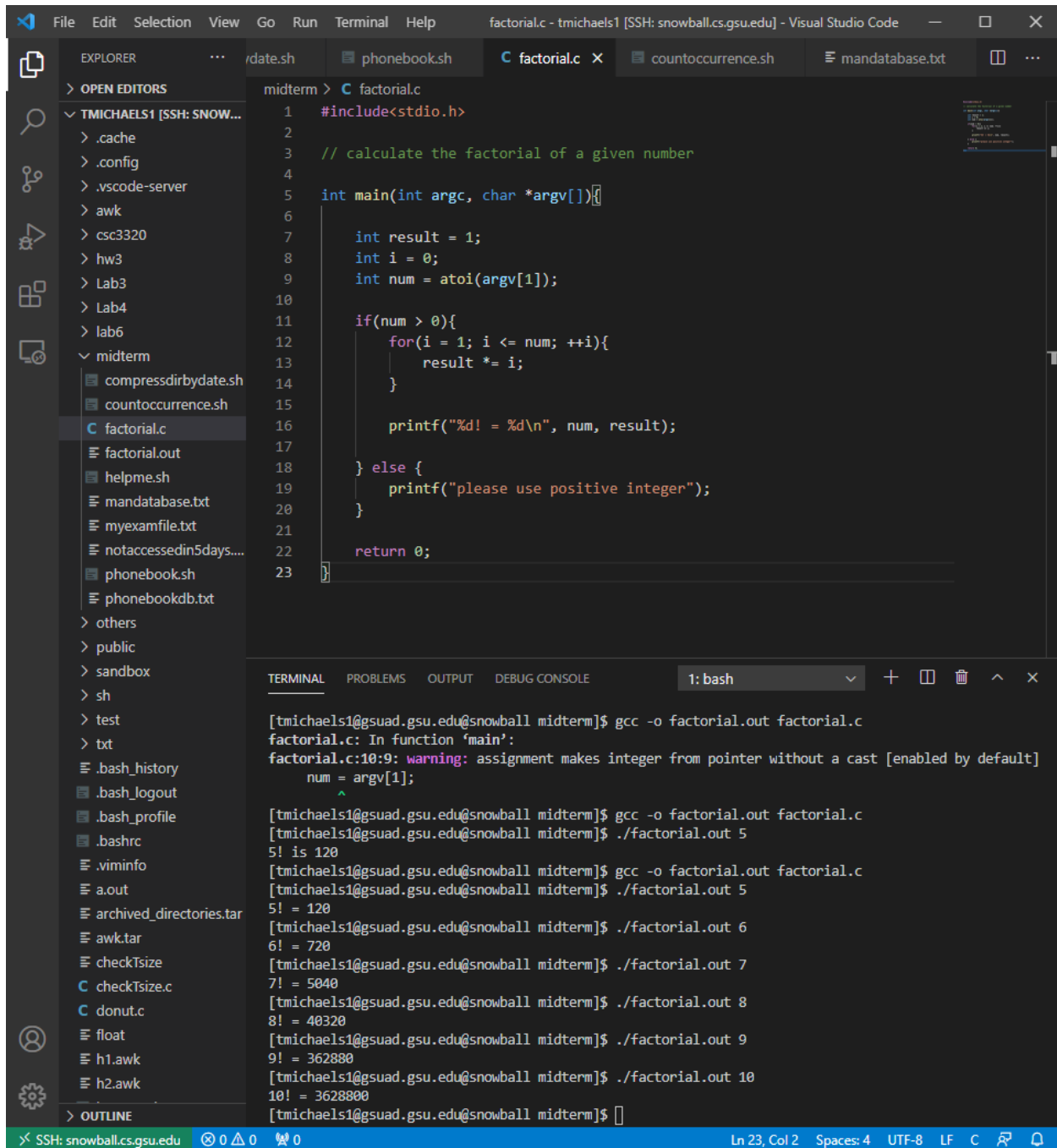
5)

A) to compile this code I used the command `gcc -o factorial.out factorial.c`

This creates a file called `factorial.out`

This program accepts a command line argument for the input so to run this program I typed `./factorial.out 5`

Which gave the desired output of $5! = 120$



The screenshot shows the Visual Studio Code editor with the `factorial.c` file open. The file contains the following C code:

```
1 #include<stdio.h>
2
3 // calculate the factorial of a given number
4
5 int main(int argc, char *argv[])
6 {
7     int result = 1;
8     int i = 0;
9     int num = atoi(argv[1]);
10
11     if(num > 0){
12         for(i = 1; i <= num; ++i){
13             result *= i;
14         }
15
16         printf("%d! = %d\n", num, result);
17     } else {
18         printf("please use positive integer");
19     }
20 }
21
22 return 0;
23 }
```

The terminal output shows the compilation and execution of the program:

```
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ gcc -o factorial.out factorial.c
factorial.c: In function 'main':
factorial.c:10:9: warning: assignment makes integer from pointer without a cast [enabled by default]
    num = argv[1];
    ^
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ gcc -o factorial.out factorial.c
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 5
5! is 120
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ gcc -o factorial.out factorial.c
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 5
5! = 120
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 6
6! = 720
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 7
7! = 5040
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 8
8! = 40320
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 9
9! = 362880
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 10
10! = 3628800
[tmichaels1@gsuad.gsu.edu@snowball midterm]$
```

B) compiled this program using `gcc -o bitshift.out bitshift.c`

Bitshift.out accepts a command line argument of an integer to perform it's operation on

The screenshot shows a Visual Studio Code editor window with the following components:

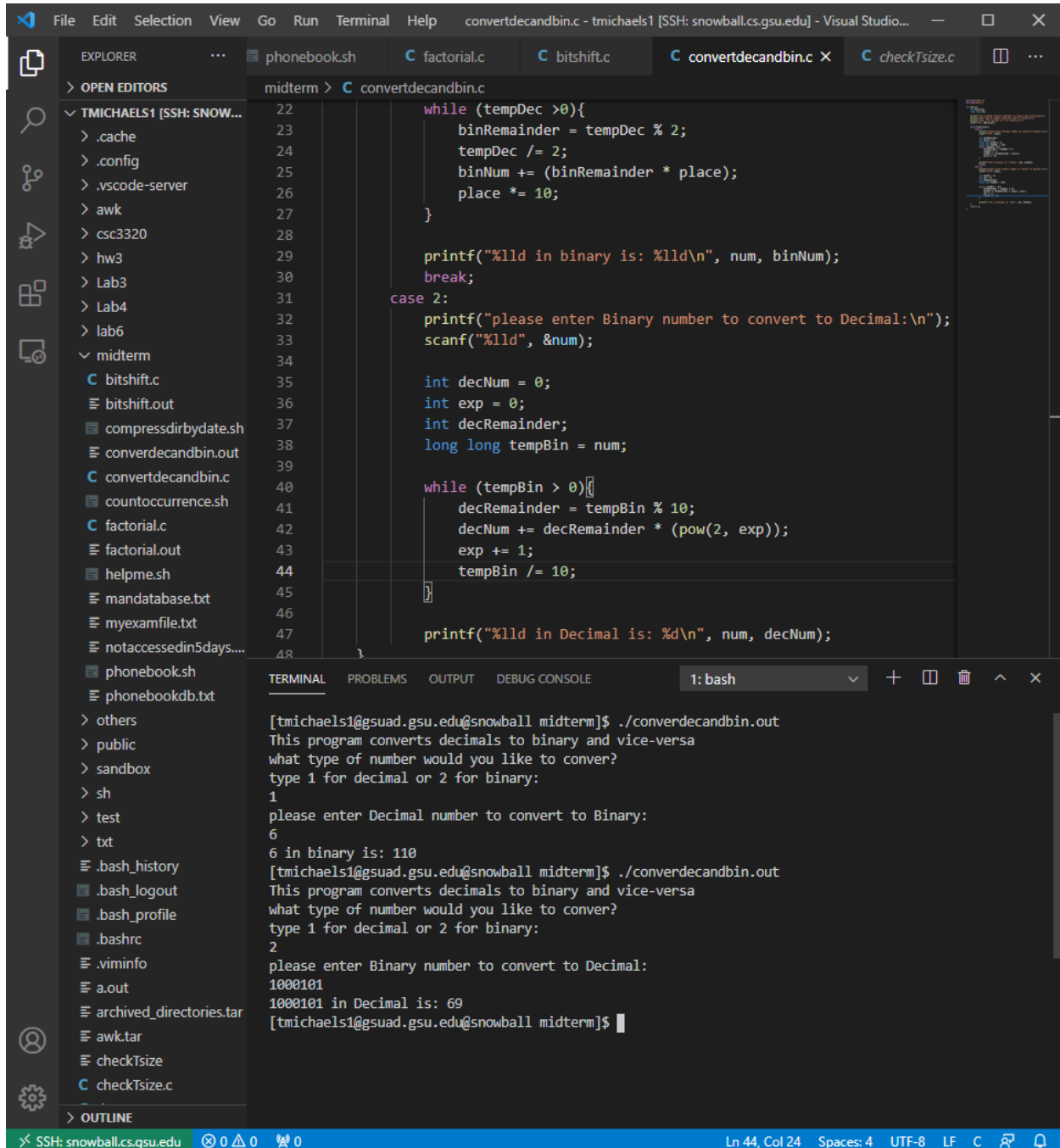
- Explorer Panel:** Displays the file structure of the project, including files like `date.sh`, `phonebook.sh`, `factorial.c`, `bitshift.c`, `countoccurrence.sh`, and various shell scripts and text files.
- Editor Panel:** Shows the source code for `bitshift.c`. The code is as follows:

```
1 #include<stdio.h>
2
3 //this program bit shifts an integer to the left by 3
4 // and then adds it to the complement
5
6 int main(int argc, char *argv[])
7 {
8     int num = atoi(argv[1]);
9     int result;
10    result = (num<<3) + ~num;
11    printf("%d bit-shifted by 3 plus it's complement is: %d\n", num, result);
12 }
```
- Terminal Panel:** Shows the execution of the program. It first runs a series of tests for a factorial program, then compiles and runs the `bitshift` program with the argument `10`. The output is:

```
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ gcc -o factorial.out factorial.c
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 5
5! is 120
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ gcc -o factorial.out factorial.c
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 5
5! = 120
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 6
6! = 720
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 7
7! = 5040
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 8
8! = 40320
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 9
9! = 362880
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./factorial.out 10
10! = 3628800
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ gcc -o bitshift.out bitshift.c
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./bitshift.out 10
10 bit-shifted by 3 plus it's complement is: 69
[tmichaels1@gsuad.gsu.edu@snowball midterm]$
```

Bonus 1) compiled with `gcc -o converdecandbin.out convertdecandbin.c -lm`

Had to use `-lm` to link math



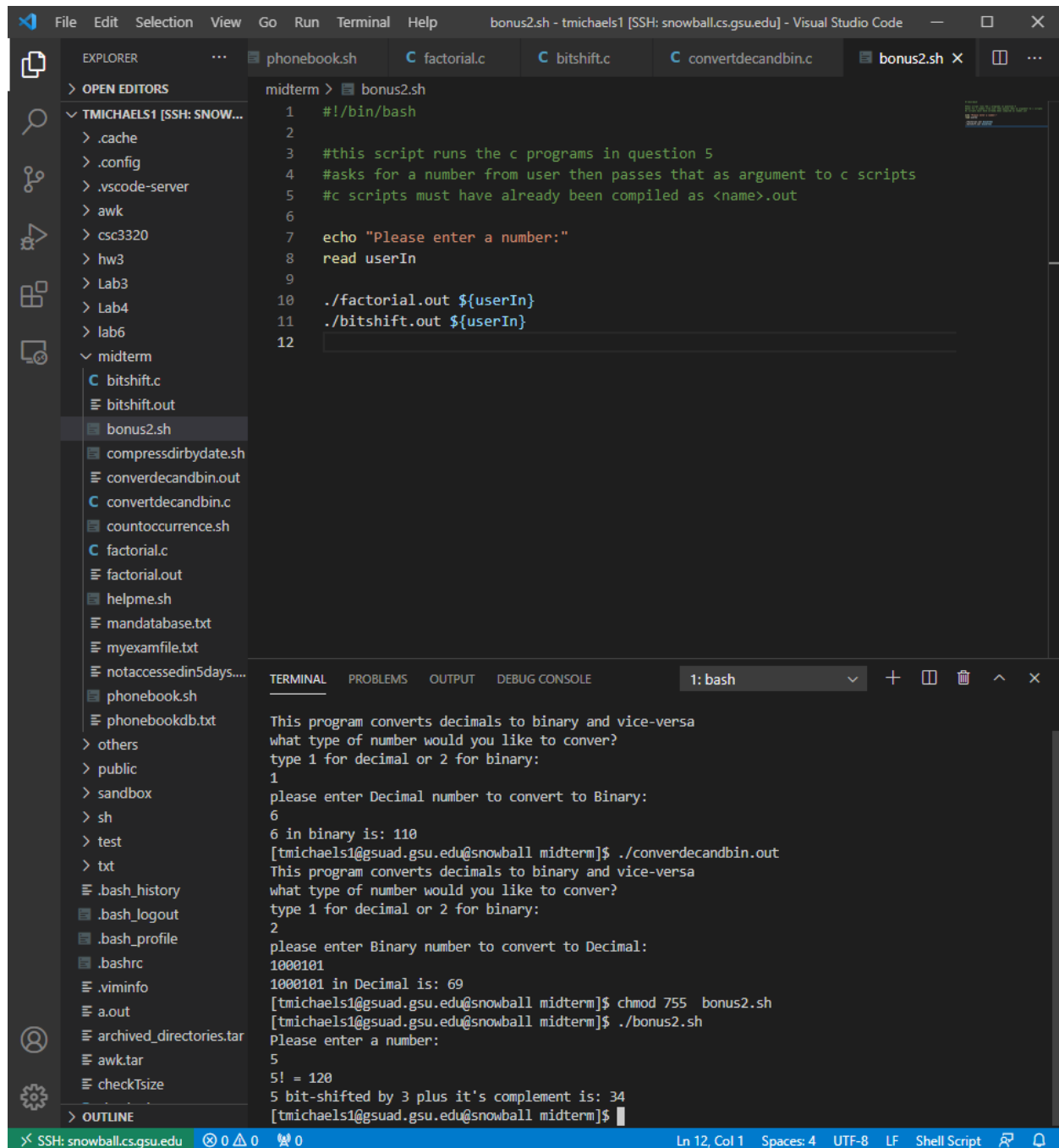
The screenshot shows the Visual Studio Code interface with the file `convertdecandbin.c` open. The code implements a program that converts decimal numbers to binary and vice versa. It uses a `while` loop to convert decimal to binary and a `while` loop to convert binary to decimal. The program prompts the user to enter a number and then asks for the type of conversion (1 for decimal to binary, 2 for binary to decimal).

```
22 while (tempDec > 0){
23     binRemainder = tempDec % 2;
24     tempDec /= 2;
25     binNum += (binRemainder * place);
26     place *= 10;
27 }
28
29 printf("%lld in binary is: %lld\n", num, binNum);
30 break;
31 case 2:
32     printf("please enter Binary number to convert to Decimal:\n");
33     scanf("%lld", &num);
34
35     int decNum = 0;
36     int exp = 0;
37     int decRemainder;
38     long long tempBin = num;
39
40     while (tempBin > 0){
41         decRemainder = tempBin % 10;
42         decNum += decRemainder * (pow(2, exp));
43         exp += 1;
44         tempBin /= 10;
45     }
46
47     printf("%lld in Decimal is: %d\n", num, decNum);
48 }
```

The terminal output shows the program being executed and the results of the conversions:

```
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./converdecandbin.out
This program converts decimals to binary and vice-versa
what type of number would you like to conver?
type 1 for decimal or 2 for binary:
1
please enter Decimal number to convert to Binary:
6
6 in binary is: 110
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./converdecandbin.out
This program converts decimals to binary and vice-versa
what type of number would you like to conver?
type 1 for decimal or 2 for binary:
2
please enter Binary number to convert to Decimal:
1000101
1000101 in Decimal is: 69
[tmichaels1@gsuad.gsu.edu@snowball midterm]$
```

Bonus 2)



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal is running a shell script named `bonus2.sh` in a directory named `midterm`. The script's content is visible in the editor above the terminal.

Script Content (bonus2.sh):

```
1  #!/bin/bash
2
3  #this script runs the c programs in question 5
4  #asks for a number from user then passes that as argument to c scripts
5  #c scripts must have already been compiled as <name>.out
6
7  echo "Please enter a number:"
8  read userIn
9
10 ./factorial.out ${userIn}
11 ./bitshift.out ${userIn}
12
```

Terminal Output:

```
1: bash
This program converts decimals to binary and vice-versa
what type of number would you like to convert?
type 1 for decimal or 2 for binary:
1
please enter Decimal number to convert to Binary:
6
6 in binary is: 110
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./converdecandbin.out
This program converts decimals to binary and vice-versa
what type of number would you like to convert?
type 1 for decimal or 2 for binary:
2
please enter Binary number to convert to Decimal:
1000101
1000101 in Decimal is: 69
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ chmod 755 bonus2.sh
[tmichaels1@gsuad.gsu.edu@snowball midterm]$ ./bonus2.sh
Please enter a number:
5
5! = 120
5 bit-shifted by 3 plus it's complement is: 34
[tmichaels1@gsuad.gsu.edu@snowball midterm]$
```

The status bar at the bottom indicates the file is `Ln 12, Col 1`, using `UTF-8` encoding, `LF` line endings, and is a `Shell Script`.

I compressed all files in the midterm directory and will upload that as well