

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: Hakan Gunerli

Campus ID: hgunerli1

Panther #: 002504797

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)

Program1:

Run using ./helpme.sh

There are 10 commands in the application, these are
wc,sed,mkdir,cp,tar,kill,sort,echo ,ssh,and sudo

```
#!/bin/bash
x=1
counter="0"

echo "enter the command you want to be searched"
read searched \c
match_condition="$(grep -i ^$searched\ ( mandatabase.txt | wc -l)"
if [ $match_condition -ge $x ]
then

while read ln; do
    if [[ $ln == ${searched^^}"("*" ] ]
    then
        echo "$ln"
        counter="1"
    elif [[ "$counter" -eq "1" && $ln == *${searched^^}"(1)" ]]
    then
        echo "$ln"
        counter="0"
    elif [[ "$counter" -eq "1" && $ln != *${searched^^}"(1)" ]]
    then
        echo "$ln"
    fi

done < mandatabase.txt

else
    echo "sorry, I cannot help you"
fi
```

Hakan Gunerli

```
hakangangerli@DELL-CAN:/mnt/d/Users/hakan/Desktop/program1$ ./helpme.sh
enter the command you want to be searched
wc
wc(1)
User Commands
WC(1)

NAME
wc - print newline, word, and byte counts for each file

SYNOPSIS
wc [OPTION]... [FILE]...
wc [OPTION]... --files0-from=F

DESCRIPTION
Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified. A word is a non-zero-length sequence of
characters delimited by white space.

With no FILE, or when FILE is -, read standard input.

The options below may be used to select which counts are printed, always in the following order: newline, word, character, byte, maximum line length.

-c, --bytes
print the byte counts

-m, --chars
print the character counts

-l, --lines
print the newline counts

--files0-from=F
read input from the files specified by NUL-terminated names in file F; If F is - then read names from standard input

-L, --max-line-length
print the maximum display width

-w, --words
print the word counts

--help display this help and exit

--version
output version information and exit

AUTHOR
Written by Paul Rubin and David MacKenzie.

REPORTING BUGS
GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Report wc translation bugs to <https://translationproject.org/team/>

COPYRIGHT
Copyright © 2018 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO
Full documentation at: <https://www.gnu.org/software/coreutils/wc>
or available locally via: info '(coreutils) wc invocation'

GNU coreutils 8.30
September 2019
WC(1)

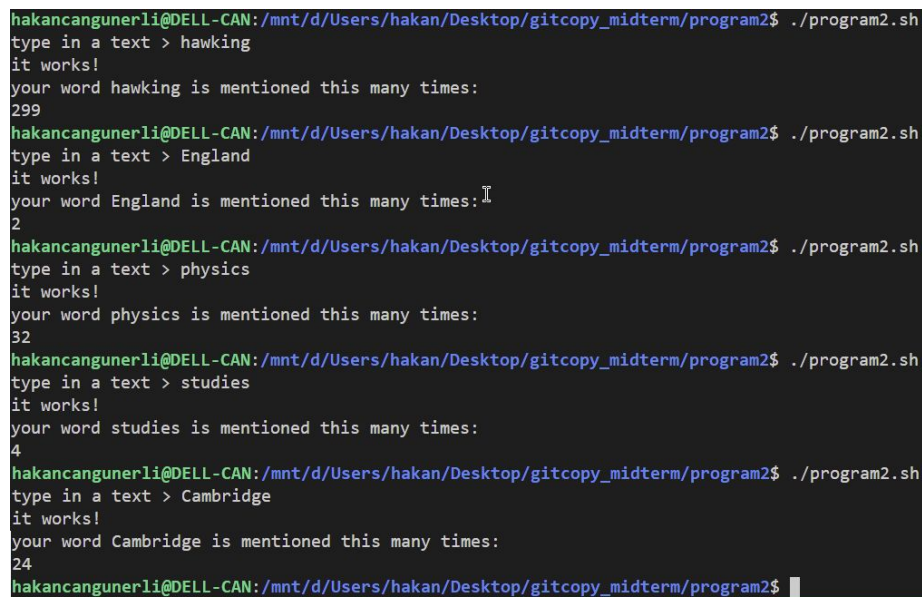
hakangangerli@DELL-CAN:/mnt/d/Users/hakan/Desktop/program1$ ./helpme.sh
enter the command you want to be searched
sd
sorry, I cannot help you
hakangangerli@DELL-CAN:/mnt/d/Users/hakan/Desktop/program1$
```

DESCRIPTION: ./helpme.sh will match between the input and input(1). If it does not match then it will say sorry, I cannot help you.

Program2:

```
read -p "type in a text > " input

if grep -q "$input" myexamfile.txt; then
    echo "it works!"
    echo "your word \"$input\" is mentioned this many times:"
    grep -o -i "$input" myexamfile.txt | wc -w
else
    echo "i do not understand, please try again."
fi
```



```
hakancangunerli@DELL-CAN:/mnt/d/Users/hakan/Desktop/gitcopy_midterm/program2$ ./program2.sh
type in a text > hawking
it works!
your word hawking is mentioned this many times:
299
hakancangunerli@DELL-CAN:/mnt/d/Users/hakan/Desktop/gitcopy_midterm/program2$ ./program2.sh
type in a text > England
it works!
your word England is mentioned this many times:
2
hakancangunerli@DELL-CAN:/mnt/d/Users/hakan/Desktop/gitcopy_midterm/program2$ ./program2.sh
type in a text > physics
it works!
your word physics is mentioned this many times:
32
hakancangunerli@DELL-CAN:/mnt/d/Users/hakan/Desktop/gitcopy_midterm/program2$ ./program2.sh
type in a text > studies
it works!
your word studies is mentioned this many times:
4
hakancangunerli@DELL-CAN:/mnt/d/Users/hakan/Desktop/gitcopy_midterm/program2$ ./program2.sh
type in a text > Cambridge
it works!
your word Cambridge is mentioned this many times:
24
hakancangunerli@DELL-CAN:/mnt/d/Users/hakan/Desktop/gitcopy_midterm/program2$
```

The examfile is the wikipedia page for Stephen Hawking.

Description: when the program is executed using ./program2.sh, the program will read the file that contains a sample wikipedia page, and using grep we print the results as piped into the amount that word is contained within our text file.

Program3:

```
read -p "how many days since passed? > " dayamount
read -p "name of output zipfile > " output
#find for all extensions, put them in, and feed that to the pipe, zip it
and take it into the filename using -@
find ~ -iname "*" -atime +$dayamount -type f|zip $output -@
# i did with txt to show, you can change it to anything really.

#bear in mind that, once this is zipped, it is zipped.
```

```
[hgunerli1@gsuad.gsu.edu@snowball program3]$ chmod 777 program3.sh
[hgunerli1@gsuad.gsu.edu@snowball program3]$ ls
program3.sh
[hgunerli1@gsuad.gsu.edu@snowball program3]$ ./program3.sh
how many days since passed? > 3
name of output zipfile > test
  adding: home/hgunerli1/csc3320/Lab3/Try.c (deflated 35%)
  adding: home/hgunerli1/csc3320/Lab3/RealEstate.csv (deflated 77%)
  adding: home/hgunerli1/csc3320/lab2/myLab2.txt (stored 0%)
  adding: home/hgunerli1/dead.letter (stored 0%)
  adding: home/hgunerli1/checkError.sh (deflated 41%)
  adding: home/hgunerli1/hello.sh (deflated 36%)
  adding: home/hgunerli1/Lab4/CSC_Course.txt (deflated 75%)
  adding: home/hgunerli1/Lab4/newList.txt (deflated 60%)
  adding: home/hgunerli1/Lab4/mountainList.txt (deflated 49%)
  adding: home/hgunerli1/homeworks/homework_instructions.txt (deflated 50%)
  adding: home/hgunerli1/simple.sh (deflated 16%)
  adding: home/hgunerli1/history.txt (deflated 66%)
  adding: home/hgunerli1/lab6/a.out (deflated 72%)
  adding: home/hgunerli1/lab6/myName.c (deflated 11%)
  adding: home/hgunerli1/lab6/foo.sh (deflated 18%)
  adding: home/hgunerli1/lab6/hello (deflated 73%)
  adding: home/hgunerli1/lab6/foo.java (deflated 47%)
  adding: home/hgunerli1/lab6/foo.class (deflated 27%)
  adding: home/hgunerli1/lab6/hello.c (stored 0%)
  adding: home/hgunerli1/.bash_logout (stored 0%)
  adding: home/hgunerli1/.homework_instructions.txt.swo (deflated 98%)
  adding: home/hgunerli1/.homework_instructions.txt.swn (deflated 93%)
  adding: home/hgunerli1/.viminfo (deflated 80%)
[hgunerli1@gsuad.gsu.edu@snowball program3]$ ls
program3.sh  test.zip
[hgunerli1@gsuad.gsu.edu@snowball program3]$
```

Description: a zip file will be created from the input of date and name of the zip file.

Program4:

Addition

```
#!/bin/sh
BOOK="address-book.txt"

read -p "name to add: " name
read -p "phone number: " phone

echo "is this correct?:"
echo -e "$name ; $phone \n"
echo -n "y/n: "
read answer

if [ "$answer" == "y" ]
then
# add it to the book.
    echo "$name ; $phone" >>$BOOK
else
# if they say no, run this.
    echo "$name ; $phone NOT written to $BOOK"
fi

exit 0
```

Deletion

```
#!/bin/sh
BOOK="address-book.txt"

# which line to delete
    read -p "Which name should I delete: " name
```


Hakan Gunerli

```
sed -e "$name/d" temp.txt | tee $BOOK
```

Find

```
#!/bin/sh
BOOK="address-book.txt"
read -p "find name: " result
fgrep -w "${result}" $BOOK
```

List

```
#!/bin/sh

BOOK="address-book.txt"

# Display the format before the entries
echo "Line Number:  Name  ;  Phone Number"

# Print the book with line numbers
nl --number-separator=";  " $BOOK
```

Phone

```
#!/bin/sh
exit=0 # this is to send an exit after.
# Name of address book
BOOK="address-book.txt"

while [ $exit -ne 1 ]
do
    echo "What operation do you want?"
    echo "available commands, add, find, del, and exit."
    read -p "type in a command > " answer

    if [ "$answer" = "add" ]
    then
        ./addition.sh
        sort -o temp.txt address-book.txt
```

```
        cp temp.txt address-book.txt
elif [ "$answer" = "list" ]
then
    ./list.sh
    sort -o temp.txt address-book.txt
    cp temp.txt address-book.txt
elif [ "$answer" = "find" ]
then
    ./find.sh
elif [ "$answer" = "del" ]
then
    ./deletion.sh
    sort -o temp.txt address-book.txt | cat temp.txt
    cp temp.txt address-book.txt
elif [ "$answer" = "exit" ]
then
    exit=1
else
    echo "I do not understand the command."
fi
done

exit 0
```

```
hakancangunerli@DELL-CAN: /mnt/d/Users/hakan/Desktop/gitcopy_midterm/code/program4$ ./phone.sh
What operation do you want?
available commands, add, find, list, del, and exit.
type in a command > add
name to add: can
phone number: 123-124-12412
is this correct?:
can ; 123-124-12412

y/n: y
What operation do you want?
available commands, add, find, list, del, and exit.
type in a command > add
name to add: abc
phone number: 583-423-2342
is this correct?:
abc ; 583-423-2342

y/n: y
What operation do you want?
available commands, add, find, list, del, and exit.
type in a command > add
name to add: jkl
phone number: 544-234-2342
is this correct?:
jkl ; 544-234-2342

y/n: y
What operation do you want?
available commands, add, find, list, del, and exit.
type in a command > add
name to add: fgh
phone number: 142-152-1231
is this correct?:
fgh ; 142-152-1231

y/n: y
What operation do you want?
available commands, add, find, list, del, and exit.
type in a command > list
Line Number:   Name   ;   Phone Number
1;   abc   ; 583-423-2342
2;   can   ; 123-124-12412
3;   fgh   ; 142-152-1231
4;   jkl   ; 544-234-2342
What operation do you want?
available commands, add, find, list, del, and exit.
type in a command > find can
I do not understand the command.
What operation do you want?
available commands, add, find, list, del, and exit.
type in a command > find
find name: can
can ; 123-124-12412
What operation do you want?
available commands, add, find, list, del, and exit.
type in a command > del
Which name should I delete: can
abc ; 583-423-2342
fgh ; 142-152-1231
jkl ; 544-234-2342
abc ; 583-423-2342
can ; 123-124-12412
fgh ; 142-152-1231
jkl ; 544-234-2342
What operation do you want?
available commands, add, find, list, del, and exit.
type in a command > exit
hakancangunerli@DELL-CAN: /mnt/d/Users/hakan/Desktop/gitcopy_midterm/code/program4$
```

Description: the ./phone.sh essentially acts as driver code to our system, and allows us to store the .txt file once an output has been added or deleted. Above is all the references made at the same time to prove functionality, and even an error is shown to show what the application does when an invalid statement is provided.

Chmod 700 ./phone.sh is executed to ensure that only the user can read, write and execute the file.

Program5:

BITWISE OPERATIONS CODE

```
#include <stdio.h>
int main(int argc, char const *argv[])// i come from a java background,
please excuse that ;p
{
    unsigned int input; // can't be a negative, it'll ruin the script.

    printf("enter an input: ");
    scanf("%d", &input);
    int shifted = (input <<3)+ ~input;

    printf("3 bit one's complement of %d is %d\n", input, shifted);
    printf("goodbye.");
    return 0;
}
```

FACTORIAL CODE

```
#include <stdio.h>
int main()
{
    int user_input;

    unsigned long long factorial = 1;
    printf("Enter an integer: ");
    scanf("%d", &user_input);
```

```
if (user_input < 0)
{
printf("Error! Factorial of a negative number doesn't exist.");
}
else if (user_input == 0)
{
printf("factorial of zero is 1.");
}

else
{
int i; // used for iterating for the loop.
i=1;
for (i; i <= user_input; i++)
{
factorial *= i;
}
printf("factorial of the given number %d is %llu.\n ", user_input,
factorial);
}

return 0;
}
```

// note: this application will not work after 20, simply due to the size in which unsigned long long can hold.



The screenshot shows a code editor with a file explorer on the left containing files like bin2decNdec2bin.c, bitwise, bitwise.c, factorial, factorial.c, main.sh, and readme.md. The main editor displays a C program that calculates the factorial of a user input and prints it in decimal and binary. The program also prints the number of bits in the binary representation. The terminal window at the bottom shows the program being executed in a WSL environment. The user enters '10' and the program outputs 'factorial of the given number 10 is 3628800.' and 'enter an input: 3 bit one's complement of 10 is 69'. The user then enters '3' and the program outputs 'factorial of the given number 3 is 6' and 'enter an input: 3 bit one's complement of 3 is 4'.

```
int i = 0;
for(i;n>0;i++){
    a[i]=n%2; // divisible by two.
    n=n/2;
}
printf("your input roughly in binary is: ");
i = i-1; // condition, i
for(i;i>=0;i--){
    printf("%d",a[i]);
}
printf(" goodbye.");
```

hakan@DELL-CAN: /mnt/d/Users/hakan/Desktop/gitcopy_midterm/code\$ cd program5/
hakan@DELL-CAN: /mnt/d/Users/hakan/Desktop/gitcopy_midterm/code/program5\$./main.sh
give me a number 10
Enter an integer: factorial of the given number 10 is 3628800.
enter an input: 3 bit one's complement of 10 is 69
goodbye.hakan@DELL-CAN: /mnt/d/Users/hakan/Desktop/gitcopy_midterm/code/program5\$

Description: ./main.sh gets the input from the user, and posts that to both bitwise and factorial operations. This output example shows the files created in green and the outputs in the terminal.

BONUS POINTS

MAIN.SH BONUS

```
read -p "give me a number " value
gcc factorial.c -o factorial
gcc bitwise.c -o bitwise
```

```
./factorial <<< $value
./bitwise <<< $value
```

```
# gcc -o bin bin2decNdec2bin.c -lm
```

Description: Please refer to the output for bitwise and factorial for how the program works.

BIN TO DEC / DEC TO BIN CONVERTED CODE BONUS

```
#include <stdio.h>
#include <math.h>
int main(){
    char letter;
    printf("target conversion, (b for binary, d for decimal) ");
    scanf("%s", &letter);
    if(letter == 'b'){
        int a[10], n; // i'll use these to store the results from
        binary inside.
        printf("input decimal value: ");
        scanf("%d", &n);
```

```

int i = 0;
for(i;n>0;i++){
    a[i]=n%2; // divisible by two.
    n=n/2;
}
printf("your input roughly in binary is: ");
i = i-1; // condition,

for(i;i>=0;i--){
    printf("%d",a[i]);
}
printf(" goodbye.");

}else if (letter == 'd'){
    int rem;
    long long bin;
    int dec = 0;
    int j = 0;
    printf("input binary value ");
    scanf("%lld", &bin);
    while(bin != 0){
        rem = bin % 10;
        bin /= 10;
        dec += rem * pow(2, j);
        j += 1;
    }
    printf("value in decimal is %d\n", dec);
    printf("goodbye.");
}else{
    printf("i do not understand your input, please try
again.");
}
return 0;
}

```

The screenshot shows a Visual Studio Code editor with a file explorer on the left containing files like bin, bin2decNdec2bin.c, bitwise, bitwise.c, factorial, factorial.c, main.sh, and readme.md. The main editor area displays the C code from the previous block. The terminal window at the bottom shows the execution output:

```

1: wsl
input binary value 1010
value in decimal is 10
goodbye.hakancangunerli@DELL-CAN: /mnt/d/Users/hakan/Desktop/gitcopy_midterm/code/program5$ ./bin
target conversion, (b for binary, d for decimal) b
input decimal value: 12
your input roughly in binary is: 1100 goodbye.hakancangunerli@DELL-CAN: /mnt/d/Users/hakan/Desktop/gitcopy_midterm/code/prog
hakancangunerli@DELL-CAN: /mnt/d/Users/hakan/Desktop/gitcopy_midterm/code/program5$

```

Hakan Gunerli

Description:When the code that is commented in ./main.sh is executed(`gcc -o bin bin2decNdec2bin.c -lm`) , the gcc compiler generates a file. This generated file could be run as ./bin and a sample output is attached as a picture.