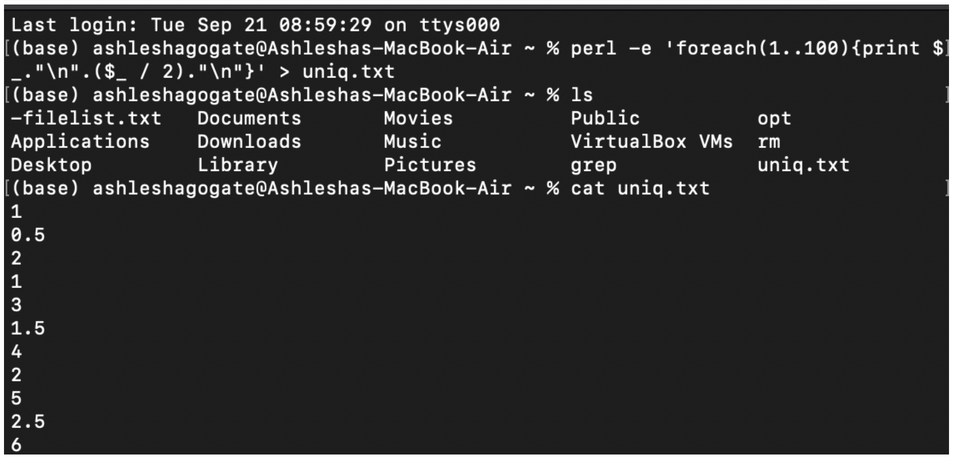
Exercises

We start our shell scripting discussion this week. Most of the concepts in this exercise will be covered in the class. There are a couple of additional commands that are used in this exercise but will not be discussed in the class. Part of the homework will be understanding what these commands are and how to use them.

sort and uniq

1. More piping with sort and `uniq` (This is a repeat from a previous assignment, just a little refresher)
    1. Create the following file:
       `perl -e 'foreach(1..100){print $_."\n".($_ / 2)."\n"}' > uniq.txt`



```
Last login: Tue Sep 21 08:59:29 on ttys000
(base) ashleshagogate@Ashleshas-MacBook-Air ~ % perl -e 'foreach(1..100){print $
_."\n".($_ / 2)."\n"}' > uniq.txt
(base) ashleshagogate@Ashleshas-MacBook-Air ~ % ls
-filelist.txt   Documents       Movies          Public          opt
Applications    Downloads       Music           VirtualBox VMs  rm
Desktop         Library         Pictures        grep            uniq.txt
(base) ashleshagogate@Ashleshas-MacBook-Air ~ % cat uniq.txt
1
0.5
2
1
3
1.5
4
2
5
2.5
6
```

    2. sort the file numerically and output it to another file without using `>` and count the number of lines

ANS: sort -n uniq.txt|tee result.txt|wc -l

The number of lines are 200.

    3. Pipe the result of the sort to `uniq`. What happened? How many lines are there?

ANS: sort -n uniq.txt|uniq|wc -l

The number of lines are now 150 because uniq got rid of duplicate values in its viscinity.

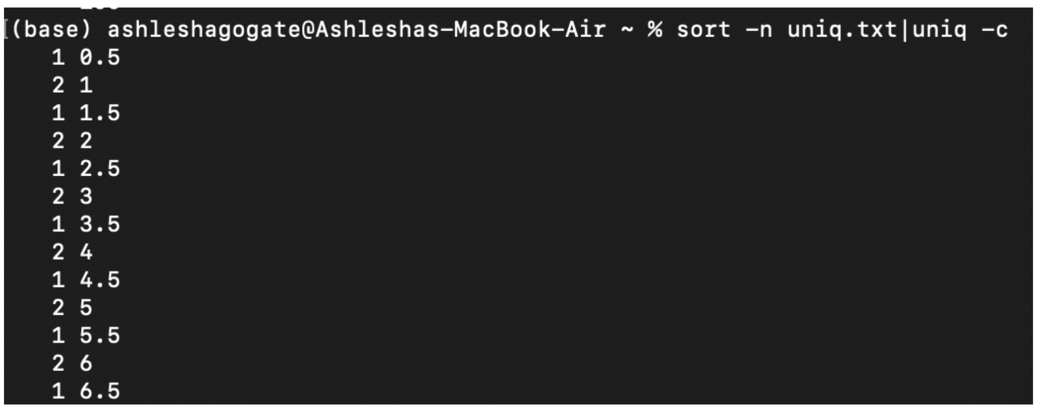
    4. Pipe the result of the sort to `uniq` and discard all lines that appear more than once. I.e., I don't want the lines (e.g., 1, 2, etc.) which occur more than once.

ANS: sort -n uniq.txt|uniq -u

    5. Pipe the result of the sort to `uniq` and count the number of times each number appear

ANS: sort -n uniq.txt|uniq -c

This ouputs two columns. The first column is the count of the number of times each entry occurs and the second is the entry itself.

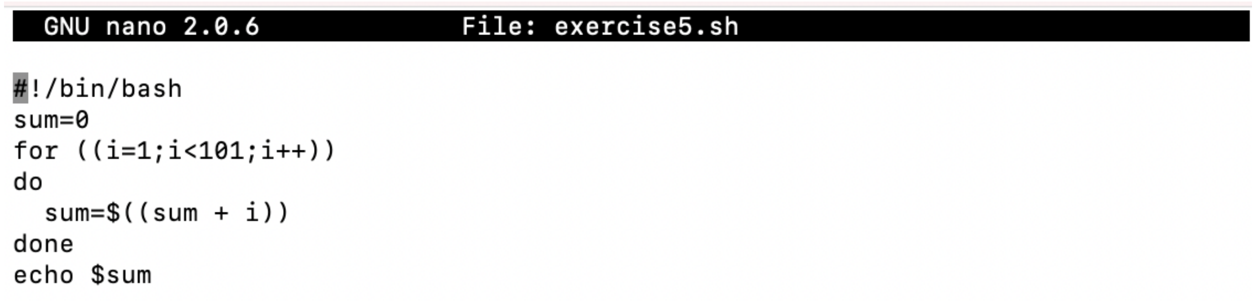


```
(base) ashleshagogate@Ashleshas-MacBook-Air ~ % sort -n uniq.txt|uniq -c
   1 0.5
   2 1
   1 1.5
   2 2
   1 2.5
   2 3
   1 3.5
   2 4
   1 4.5
   2 5
   1 5.5
   2 6
   1 6.5
```

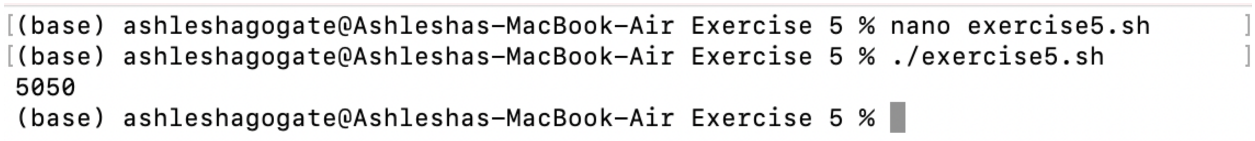
Basic Shell scripting

2. Write shell scripts for the following:
    1. Write a shell loop that adds up all the numbers from 1-100

ANS:



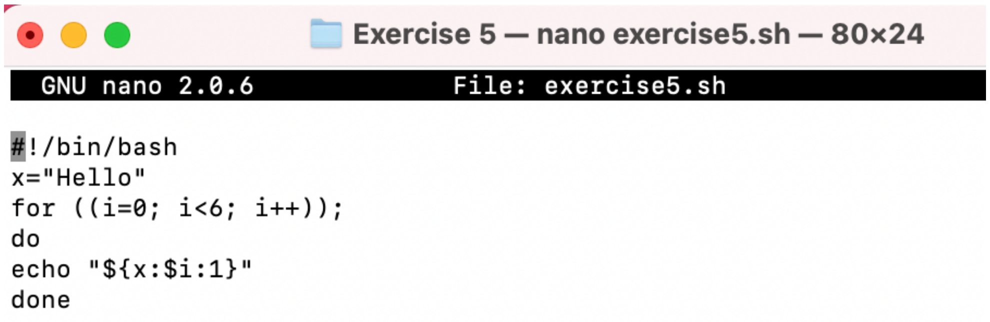
```
 GNU nano 2.0.6                File: exercise5.sh

#!/bin/bash
sum=0
for ((i=1;i<101;i++))
do
  sum=$((sum + i))
done
echo $sum
```



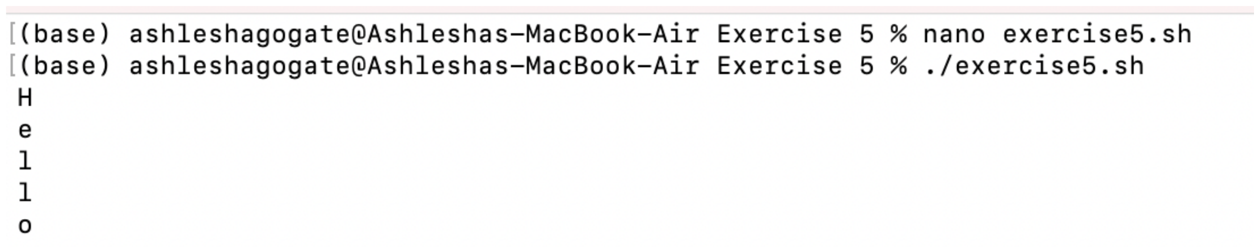
```
(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 % nano exercise5.sh
(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 % ./exercise5.sh
5050
(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 %
```

2. Write a shell loop that prints each letter in the word Hello separately. This can be done using a for loop iterating through the letters H e l l o. This is a 5 line script; don't make it harder than that.

ANS:



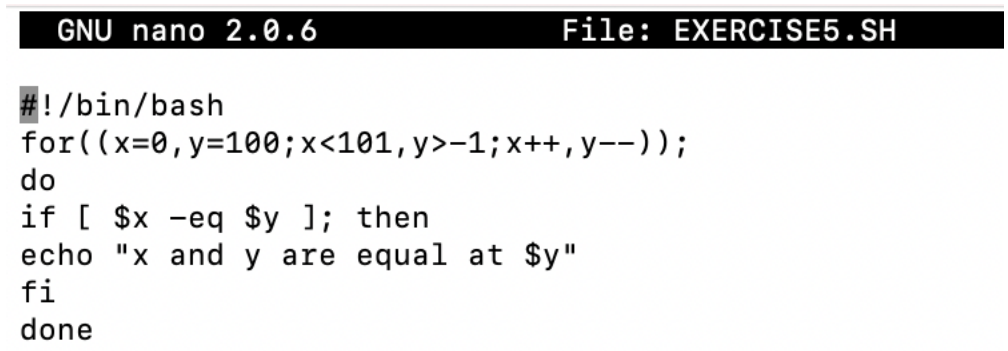
```
Exercise 5 — nano exercise5.sh — 80×24
 GNU nano 2.0.6                File: exercise5.sh

#!/bin/bash
x="Hello"
for ((i=0; i<6; i++));
do
echo "${x:$i:1}"
done
```



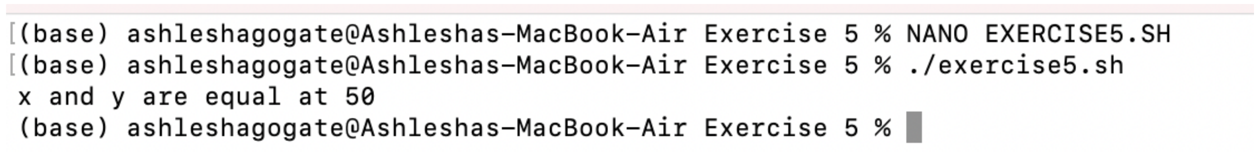
```
[(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 % nano exercise5.sh
[(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 % ./exercise5.sh
H
e
l
l
o
```

3. Have variable x increase from 0 to 100 and variable y decrease from 100 to 0 by 1 unit at a time and print "x and y are equal" when the variables are equal

ANS:



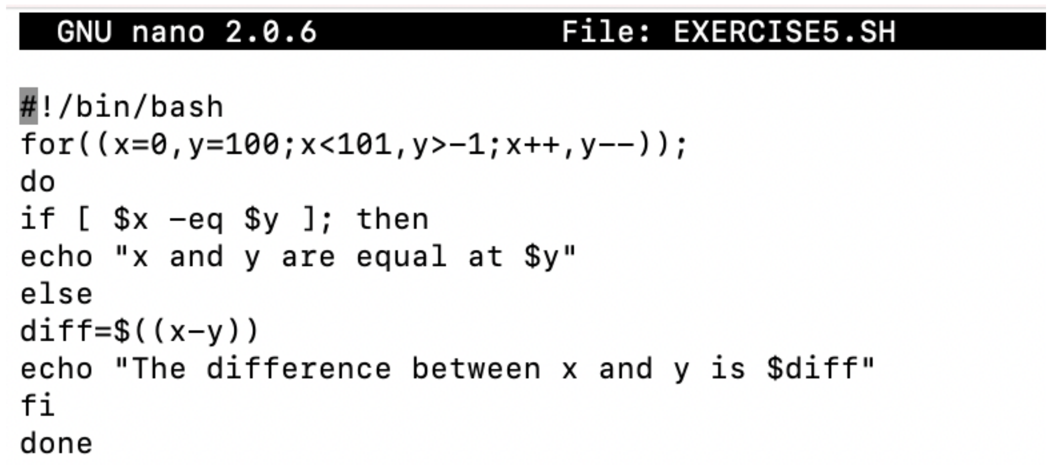
```
GNU nano 2.0.6                File: EXERCISE5.SH

#!/bin/bash
for((x=0,y=100;x<101,y>-1;x++,y--));
do
if [ $x -eq $y ]; then
echo "x and y are equal at $y"
fi
done
```



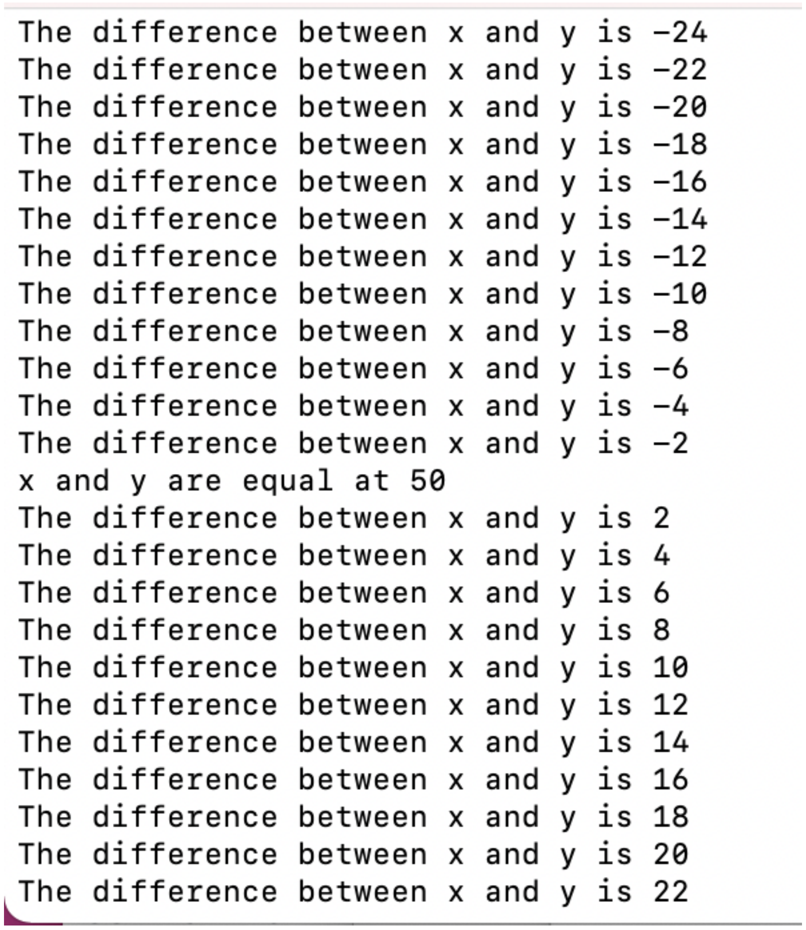
```
[(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 % NANO EXERCISE5.SH
[(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 % ./exercise5.sh
x and y are equal at 50
(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 %
```

4. Repeat the above (c) but this time print the difference between x and y ($x-$y) when they are not equal and print "x and y are equal" when they are equal

ANS:



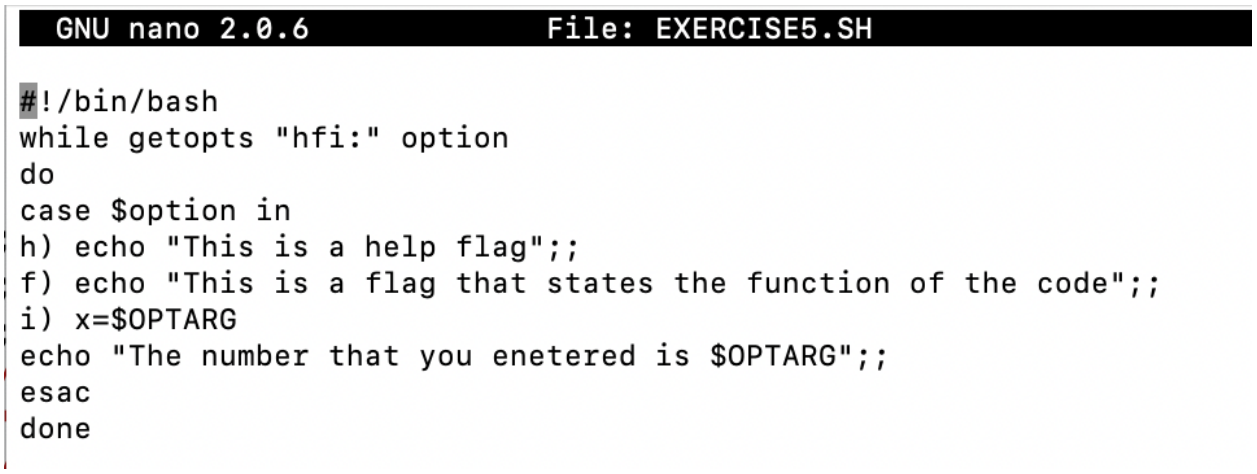
```
GNU nano 2.0.6                File: EXERCISE5.SH

#!/bin/bash
for((x=0,y=100;x<101,y>-1;x++,y--));
do
if [ $x -eq $y ]; then
echo "x and y are equal at $y"
else
diff=$((x-y))
echo "The difference between x and y is $diff"
fi
done
```



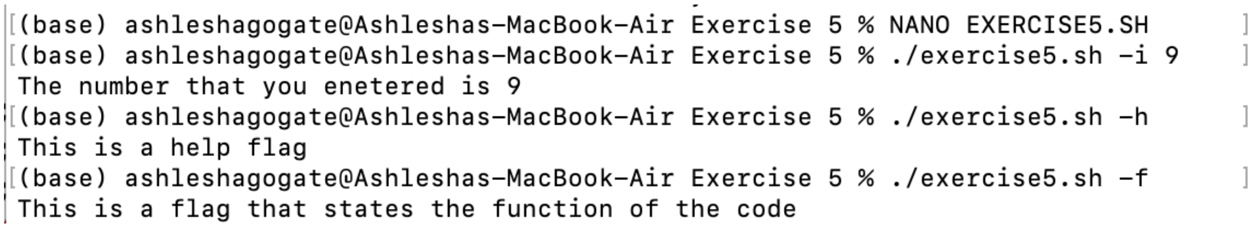
```
The difference between x and y is -24
The difference between x and y is -22
The difference between x and y is -20
The difference between x and y is -18
The difference between x and y is -16
The difference between x and y is -14
The difference between x and y is -12
The difference between x and y is -10
The difference between x and y is -8
The difference between x and y is -6
The difference between x and y is -4
The difference between x and y is -2
x and y are equal at 50
The difference between x and y is 2
The difference between x and y is 4
The difference between x and y is 6
The difference between x and y is 8
The difference between x and y is 10
The difference between x and y is 12
The difference between x and y is 14
The difference between x and y is 16
The difference between x and y is 18
The difference between x and y is 20
The difference between x and y is 22
```

5. Write a **getopts** block with 3 options including one that requires an argument

ANS:



```
  GNU nano 2.0.6                 File: EXERCISE5.SH

#!/bin/bash
while getopts "hfi:" option
do
case $option in
h) echo "This is a help flag";;
f) echo "This is a flag that states the function of the code";;
i) x=$OPTARG
echo "The number that you enetered is $OPTARG";;
esac
done
```



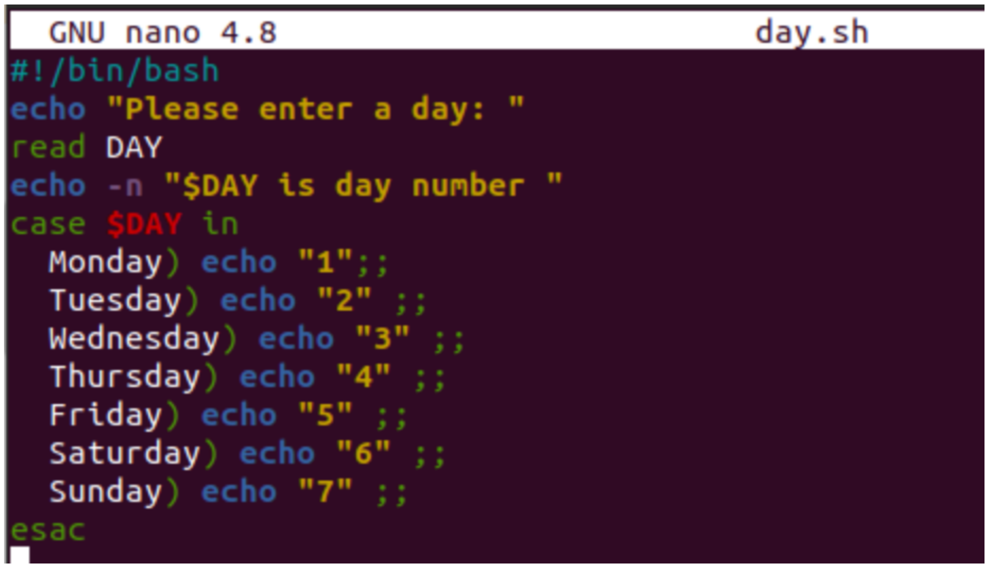
```
(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 % NANO EXERCISE5.SH
(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 % ./exercise5.sh -i 9
The number that you enetered is 9
(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 % ./exercise5.sh -h
This is a help flag
(base) ashleshagogate@Ashleshas-MacBook-Air Exercise 5 % ./exercise5.sh -f
This is a flag that states the function of the code
```

6. Write a script utilizing case that asks the user the day of the week and prints the day number based on user input
   Monday = 1; Tuesday = 2; ...; Sunday = 7

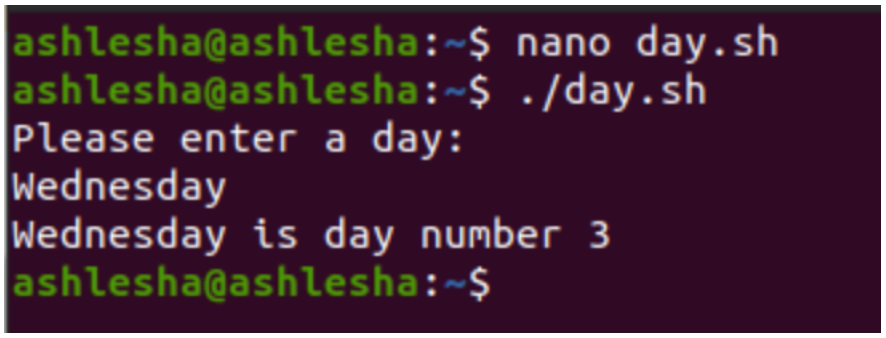
   Example terminal output (lines that start with > are output from the script, lines that start with $ are user input):

```
./scriptName.sh
> Please enter a day:
$ Monday
> Monday is day number 1
```

ANS:



```
  GNU nano 4.8                              day.sh
#!/bin/bash
echo "Please enter a day: "
read DAY
echo -n "$DAY is day number "
case $DAY in
  Monday) echo "1";;
  Tuesday) echo "2" ;;
  Wednesday) echo "3" ;;
  Thursday) echo "4" ;;
  Friday) echo "5" ;;
  Saturday) echo "6" ;;
  Sunday) echo "7" ;;
esac
```



```
ashlesha@ashlesha:~$ nano day.sh
ashlesha@ashlesha:~$ ./day.sh
Please enter a day:
Wednesday
Wednesday is day number 3
ashlesha@ashlesha:~$
```

Mini challenge part 1

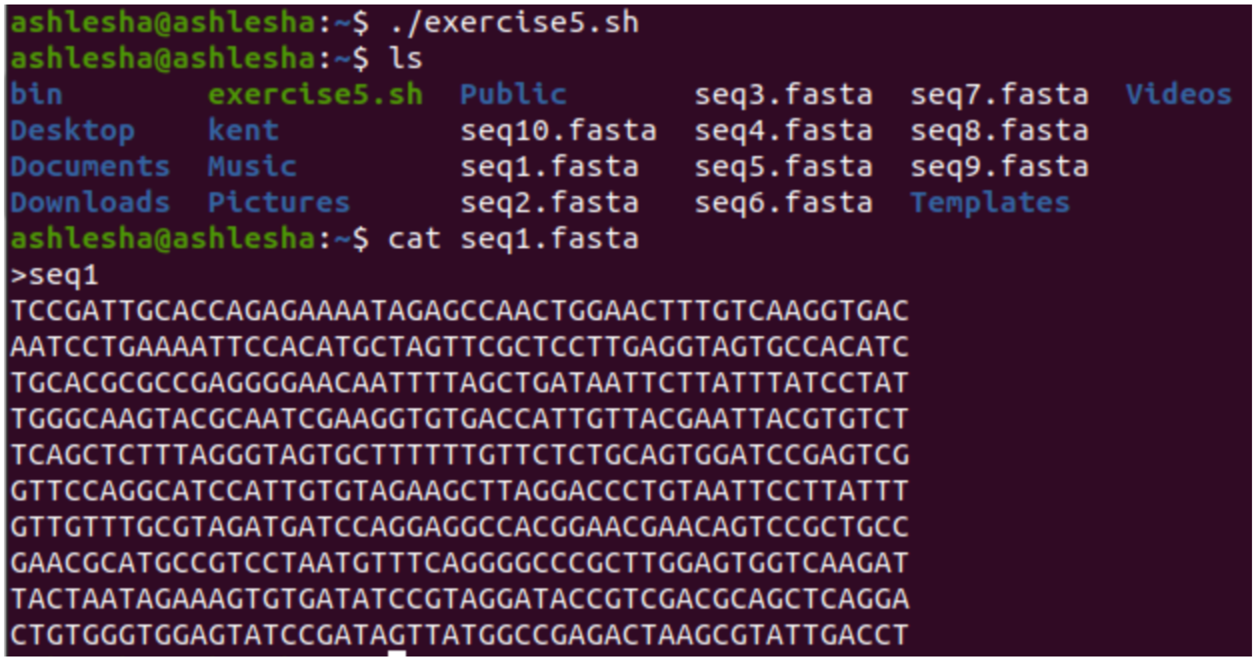
3. We are going to do an exercise for fun, but it will enforce shell concepts and will help you prepare for this week. To make things easier for you, I have provided you with stepwise

instructions on how to proceed. You will add one element each time, making the loop a little more complicated as compared with the step before.

1. Write a `for` loop to create 10 files by the name `seq1.fasta`, `seq2.fasta`, `seq3.fasta`, and so on. You can create a file using the touch command.
2. Modify this loop to now do two more things:
   i. Delete the `seq1.fasta`, `seq2.fasta`, `seq3.fasta`, etc. if they exist
   ii. Create new `seq1.fasta`, `seq2.fasta`, `seq3.fasta`, etc. files with a FASTA description line in it. The FASTA description lines needs to be like this: `>seq1 for seq1.fasta`, `>seq2 for seq2.fasta`, `>seq3 for seq3.fasta`, etc.
3. Add another element to this loop: the loop now also adds a random DNA sequence along with the FASTA description line. The following command will give you a random DNA string of 50x10 letters: cat /dev/urandom | tr -dc 'ACGT' | fold -w 50 | head

ANS:

```
#!/bin/bash
for ((i=1;i<11;i++));
do
touch seq$i.fasta
if [ $i  -le '10' ]; then
rm -r seq$i.fasta
fi
echo ">seq$i" > seq$i.fasta
cat /dev/urandom | tr -dc 'ACGT' | fold -w 50 | head >> seq$i.fasta
done
```



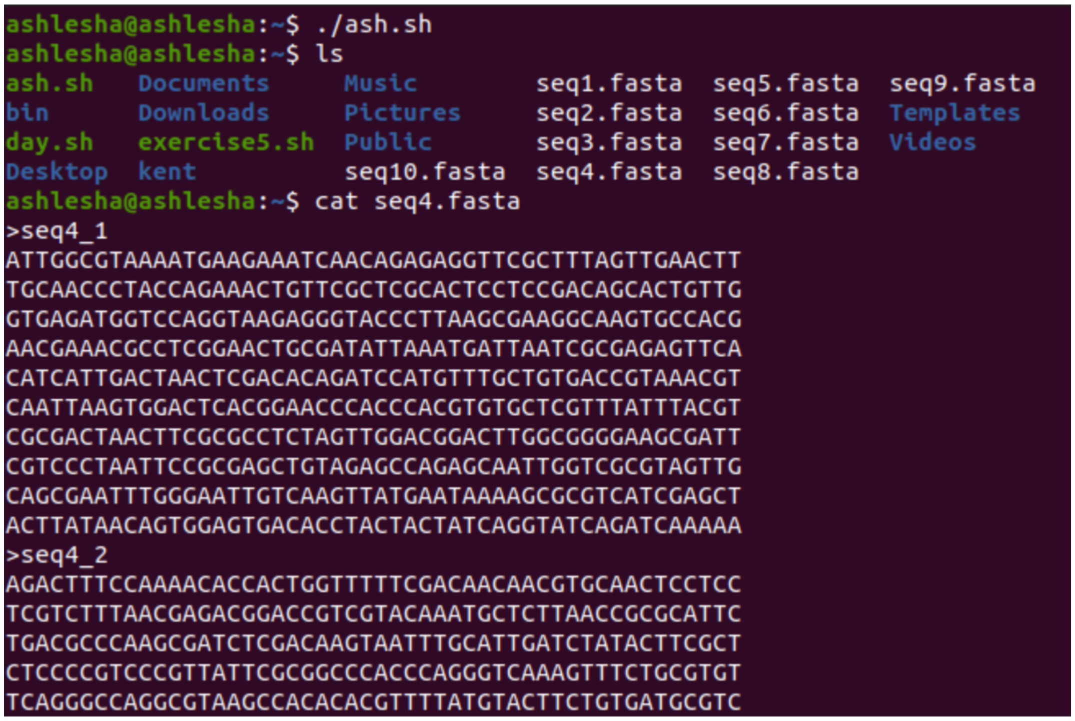
```
ashlesha@ashlesha:~$ ./exercise5.sh
ashlesha@ashlesha:~$ ls
bin        exercise5.sh  Public       seq3.fasta  seq7.fasta  Videos
Desktop    kent          seq10.fasta  seq4.fasta  seq8.fasta
Documents  Music         seq1.fasta   seq5.fasta  seq9.fasta
Downloads  Pictures      seq2.fasta   seq6.fasta  Templates
ashlesha@ashlesha:~$ cat seq1.fasta
>seq1
TCCGATTGCACCAGAGAAAATAGAGCCAACTGGAACTTTGTCAAGGTGAC
AATCCTGAAAATTCCACATGCTAGTTCGCTCCTTGAGGTAGTGCCACATC
TGCACGCGCCGAGGGGAACAATTTTAGCTGATAATTCTTATTTATCCTAT
TGGGCAAGTACGCAATCGAAGGTGTGACCATTGTTACGAATTACGTGTCT
TCAGCTCTTTAGGGTAGTGCTTTTTTGTTCTCTGCAGTGGATCCGAGTCG
GTTCCAGGCATCCATTGTGTAGAAGCTTAGGACCCTGTAATTCCTTATTT
GTTGTTTGCGTAGATGATCCAGGAGGCCACGGAACGAACAGTCCGCTGCC
GAACGCATGCCGTCCTAATGTTTCAGGGGCCCGCTTGGAGTGGTCAAGAT
TACTAATAGAAAGTGTGATATCCGTAGGATACCGTCGACGCAGCTCAGGA
CTGTGGGTGGAGTATCCGATAGTTATGGCCGAGACTAAGCGTATTGACCT
```

Mini challenge part 2: Let’s try nesting the loops

4. Nesting loops is really having a loop within a loop. Instead of creating 10 single sequence files, as in part 1, we will create 10 multiple sequence FASTA files (aka, 10 multi-FASTA files) with 8 sequences in each.

   The FASTA descriptors for file 1 will go like this: `>seq1_1`, `>seq1_2`, `>seq1_3`, …; the FASTA descriptor for file2 will go like this: `>seq2_1`, `>seq2_2`, `>seq2_3`, … ; etc.

```
#!/bin/bash
for ((i=1;i<11;i++));
do
touch seq$i.fasta
if [ $i  -le '10' ]; then
rm -r seq$i.fasta
fi
for ((j=1;j<9;j++));
do
echo ">seq${i}_${j}" >> seq$i.fasta
cat /dev/urandom | tr -dc 'ACGT' | fold -w 50 | head >> seq$i.fasta
done
done
```

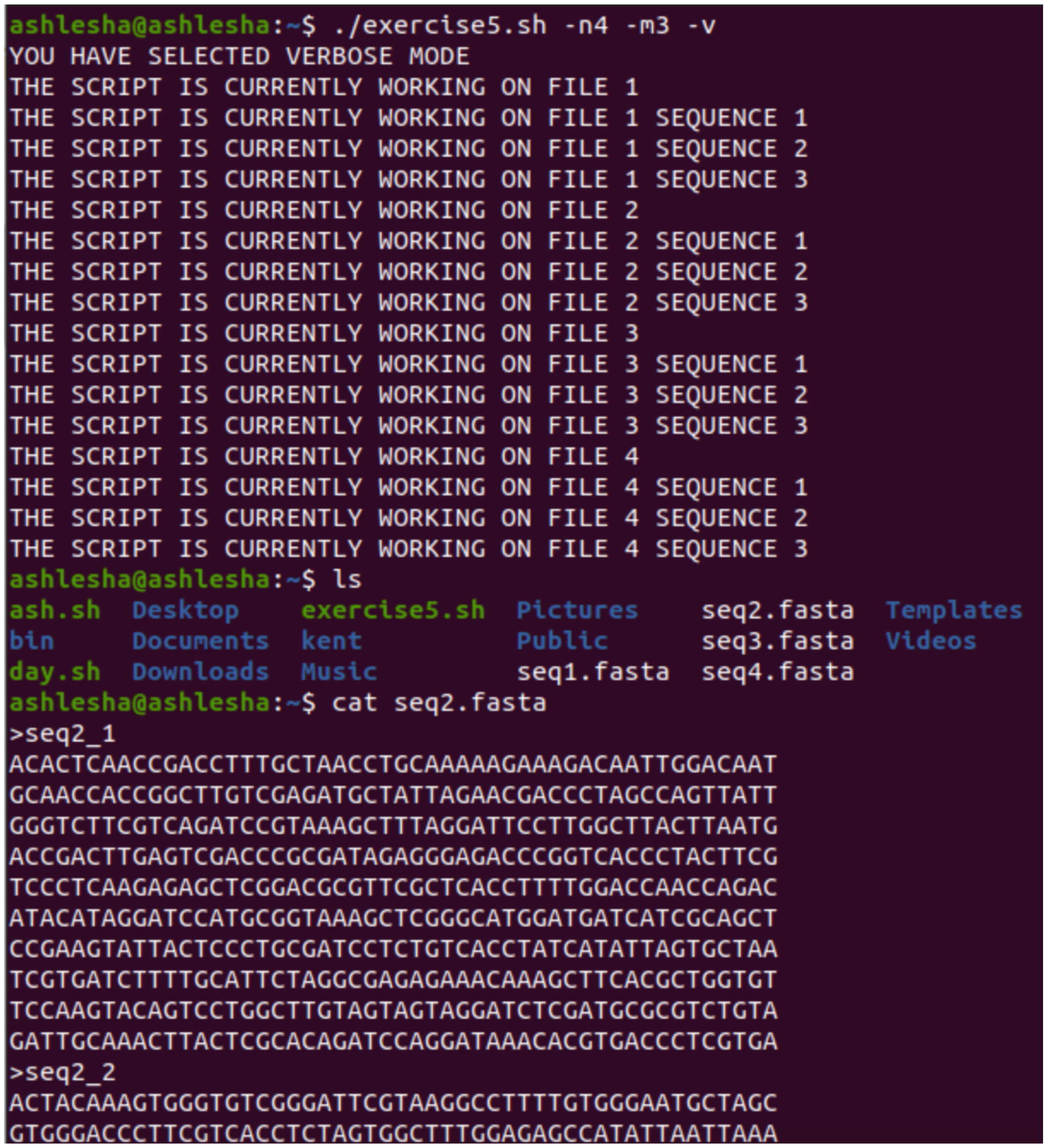


```
ashlesha@ashlesha:~$ ./ash.sh
ashlesha@ashlesha:~$ ls
ash.sh   Documents     Music        seq1.fasta  seq5.fasta  seq9.fasta
bin      Downloads     Pictures     seq2.fasta  seq6.fasta  Templates
day.sh   exercise5.sh  Public       seq3.fasta  seq7.fasta  Videos
Desktop  kent          seq10.fasta  seq4.fasta  seq8.fasta
ashlesha@ashlesha:~$ cat seq4.fasta
>seq4_1
ATTGGCGTAAAATGAAGAAATCAACAGAGAGGTTCGCTTTAGTTGAACTT
TGCAACCCTACCAGAAACTGTTCGCTCGCACTCCTCCGACAGCACTGTTG
GTGAGATGGTCCAGGTAAGAGGGTACCCTTAAGCGAAGGCAAGTGCCACG
AACGAAACGCCTCGGAACTGCGATATTAAATGATTAATCGCGAGAGTTCA
CATCATTGACTAACTCGACACAGATCCATGTTTGCTGTGACCGTAAACGT
CAATTAAGTGGACTCACGGAACCCACCCACGTGTGCTCGTTTATTTACGT
CGCGACTAACTTCGCGCCTCTAGTTGGACGGACTTGGCGGGGAAGCGATT
CGTCCCTAATTCCGCGAGCTGTAGAGCCAGAGCAATTGGTCGCGTAGTTG
CAGCGAATTTGGGAATTGTCAAGTTATGAATAAAAGCGCGTCATCGAGCT
ACTTATAACAGTGGAGTGACACCTACTACTATCAGGTATCAGATCAAAAA
>seq4_2
AGACTTTCCAAAACACCACTGGTTTTTCGACAACAACGTGCAACTCCTCC
TCGTCTTTAACGAGACGGACCGTCGTACAAATGCTCTTAACCGCGCATTC
TGACGCCCAAGCGATCTCGACAAGTAATTTGCATTGATCTATACTTCGCT
CTCCCCGTCCCGTTATTCGCGGCCCACCCAGGGTCAAAGTTTCTGCGTGT
TCAGGGCCAGGCGTAAGCCACACACGTTTTATGTACTTCTGTGATGCGTC
```

Mini challenge part 3: Now add getopts to the beginning of the script

5. For the final part of this series of questions, I want you to add **getopts** to the script so that it takes take three options:
   - Option **n** => will take a number as an input. This argument describes the number of files to be created (which until now was 10)
   - Option **m** => will take a number as an input. This argument describes the number of sequence to be added in each file (which until now was 8)
   - Option **v** => verbose mode, i.e., I want the script to print out every single action it is doing (e.g., what file is the script currently working on? what sequence number?). This is a flag, so no input is expected with it.
     - Again, you should submit a script for the 3rd Mini Challenge as part of your submission on Canvas.

ANS: Uploaded agogate8.sh on canvas



```
ashlesha@ashlesha:~$ ./exercise5.sh -n4 -m3 -v
YOU HAVE SELECTED VERBOSE MODE
THE SCRIPT IS CURRENTLY WORKING ON FILE 1
THE SCRIPT IS CURRENTLY WORKING ON FILE 1 SEQUENCE 1
THE SCRIPT IS CURRENTLY WORKING ON FILE 1 SEQUENCE 2
THE SCRIPT IS CURRENTLY WORKING ON FILE 1 SEQUENCE 3
THE SCRIPT IS CURRENTLY WORKING ON FILE 2
THE SCRIPT IS CURRENTLY WORKING ON FILE 2 SEQUENCE 1
THE SCRIPT IS CURRENTLY WORKING ON FILE 2 SEQUENCE 2
THE SCRIPT IS CURRENTLY WORKING ON FILE 2 SEQUENCE 3
THE SCRIPT IS CURRENTLY WORKING ON FILE 3
THE SCRIPT IS CURRENTLY WORKING ON FILE 3 SEQUENCE 1
THE SCRIPT IS CURRENTLY WORKING ON FILE 3 SEQUENCE 2
THE SCRIPT IS CURRENTLY WORKING ON FILE 3 SEQUENCE 3
THE SCRIPT IS CURRENTLY WORKING ON FILE 4
THE SCRIPT IS CURRENTLY WORKING ON FILE 4 SEQUENCE 1
THE SCRIPT IS CURRENTLY WORKING ON FILE 4 SEQUENCE 2
THE SCRIPT IS CURRENTLY WORKING ON FILE 4 SEQUENCE 3
ashlesha@ashlesha:~$ ls
ash.sh  Desktop    exercise5.sh  Pictures    seq2.fasta  Templates
bin     Documents  kent          Public      seq3.fasta  Videos
day.sh  Downloads  Music         seq1.fasta  seq4.fasta
ashlesha@ashlesha:~$ cat seq2.fasta
>seq2_1
ACACTCAACCGACCTTTGCTAACCTGCAAAAAGAAAGACAATTGGACAAT
GCAACCACCGGCTTGTCGAGATGCTATTAGAACGACCCTAGCCAGTTATT
GGGTCTTCGTCAGATCCGTAAAGCTTTAGGATTCCTTGGCTTACTTAATG
ACCGACTTGAGTCGACCCGCGATAGAGGGAGACCCGGTCACCCTACTTCG
TCCCTCAAGAGAGCTCGGACGCGTTCGCTCACCTTTTGGACCAACCAGAC
ATACATAGGATCCATGCGGTAAAGCTCGGGCATGGATGATCATCGCAGCT
CCGAAGTATTACTCCCTGCGATCCTCTGTCACCTATCATATTAGTGCTAA
TCGTGATCTTTTGCATTCTAGGCGAGAGAAACAAAGCTTCACGCTGGTGT
TCCAAGTACAGTCCTGGCTTGTAGTAGTAGGATCTCGATGCGCGTCTGTA
GATTGCAAACTTACTCGCACAGATCCAGGATAAACACGTGACCCTCGTGA
>seq2_2
ACTACAAAGTGGGTGTCGGGATTCGTAAGGCCTTTTGTGGGAATGCTAGC
GTGGGACCCTTCGTCACCTCTAGTGGCTTTGGAGAGCCATATTAATTAAA
```