# OPS 102 Week 9

Name Chetan Arora ID:100976240

Include screen shot of each of the task1. Shell file should be copied to ~syed.misbahuddin/fall2023/week9” . Choose file name as <studentID_week9_taskn.bash”

Where “n” is the task number.

Task 1:

Part A:

Write a script “mycat” that display the content of a file.

- The program asks user to enter the file name. It displays the message “No such file” if file does not exit
- If the file exist, the program should check if the file is empty or not. If the file is empty, the code displays message “Nothing to display”. Otherwise, it displays the file content on the screen.

```
carora18@mtrx-node01pd:~/ops
> es/186/it-services/wiki/view/1024/vpn
| Instructions on using Employee VPN: https://employees.senecapolytechnic.ca/sp
> aces/77/it-services/wiki/view/3716/vpn
 End of banner message from server
 carora18@matrix.senecacollege.ca's password:
Last login: Sat Mar 16 20:46:13 2024 from 10.29.33.102
[carora18@mtrx-node01pd ~]$ pwd
/home/carora18
[carora18@mtrx-node01pd ~]$ cd ops
[carora18@mtrx-node01pd ops]$ ls
assignmenttask3.bash  assignmenttask4output.txt  assignmenttask6.bash
assignmenttask4.bash  assignmenttask5.bash
[carora18@mtrx-node01pd ops]$ nano 100976240_week9_task1.bash
[carora18@mtrx-node01pd ops]$ nano 100976240_week9_task1.bash
[carora18@mtrx-node01pd ops]$ 100976240_week9_task1.bash
-bash: ./100976240_week9_task1.bash: Permission denied
[carora18@mtrx-node01pd ops]$ chmod u+x 100976240_week9_task.bash
chmod: cannot access '100976240_week9_task.bash': No such file or directory
[carora18@mtrx-node01pd ops]$ ls
100976240_week9_task1.bash  assignmenttask3.bash  assignmenttask4.bash  assignmenttask4output.txt  assignmenttask5.bash  assignmenttask6.bash
[carora18@mtrx-node01pd ops]$ chmod u+x 100976240_week9_task1.bash
[carora18@mtrx-node01pd ops]$ 100976240_week9_task1.bash
Enter the file name: assignmenttask4output.txt
Name: Chetan Arora
College: Seneca College
Program name: Computer Programming
StudentID: 100976240
[carora18@mtrx-node01pd ops]$ cat 100976240_week9_task1.bash
#!/bin/bash

read -p "Enter the file name: " filename

if [ ! -f "$filename" ]; then
    echo "No such file"

fi

if [ ! -s "$filename" ]; then
    echo "Nothing to display"
fi

cat "$filename"

[carora18@mtrx-node01pd ops]$
```

Part B:

Following code accepts only one command line argument. Test following code:

```
if [ $# -ne 1 ]
then
  echo "USAGE: $0 [arg]"
  exit 1
fi
```

Incorporate above command in Part A.

Task2

Using Bash if-elif-else structure, write a code to read numeric grade and displays letter grade according to following scheme:
Above 95 => A+
90 to 94 => A
85 to 89 => B+
80 to 84 => B
75 to 79 => C+
7o to 74 => C
65 to 69 => D+
60 to 64 => D
Below 60 => F

carora18@mtrx-node01pd:~/ops

```
[carora18@mtrx-node01pd ops]$ nano 100976240_week9_task2.bash
[carora18@mtrx-node01pd ops]$ chmod u+x 100976240_week9_task2.bash
[carora18@mtrx-node01pd ops]$ 100976240_week9_task2.bash
Enter numeric grade: 96
Letter grade: A+
[carora18@mtrx-node01pd ops]$ 100976240_week9_task2.bash
Enter numeric grade: 65
Letter grade: D+
[carora18@mtrx-node01pd ops]$ cat 100976240_week9_task2.bash
#!/bin/bash

read -p "Enter numeric grade: " grade

if ((grade >= 95)); then
    echo "Letter grade: A+"
elif ((grade >= 90 && grade <= 94)); then
    echo "Letter grade: A"
elif ((grade >= 85 && grade <= 89)); then
    echo "Letter grade: B+"
elif ((grade >= 80 && grade <= 84)); then
    echo "Letter grade: B"
elif ((grade >= 75 && grade <= 79)); then
    echo "Letter grade: C+"
elif ((grade >= 70 && grade <= 74)); then
    echo "Letter grade: C"
elif ((grade >= 65 && grade <= 69)); then
    echo "Letter grade: D+"
elif ((grade >= 60 && grade <= 64)); then
    echo "Letter grade: D"
else
    echo "Letter grade: F"
fi

[carora18@mtrx-node01pd ops]$ cp 100976240_week9_task2.bash ~syed.misbahuddin/fall2023/week9
[carora18@mtrx-node01pd ops]$
```

Task 3

Test following codes and explain.

Code1

```
read -p "Enter a number: " number
while [ $number -ne 5 ]
do
read -p "Try again. Enter a number: " number
  if [ $number -eq 5 ]
  then
    break
  fi
done
```

```
[carora18@mtrx-node01pd ops]$ nano 100976240_week9_task3.bash
[carora18@mtrx-node01pd ops]$ chmod u+x 100976240_week9_task3.bash
[carora18@mtrx-node01pd ops]$ 100976240_week9_task2.bash
Enter numeric grade: ^[[A^[[A^X^C
[carora18@mtrx-node01pd ops]$ 100976240_week9_task3.bash
Enter a number: 25
Try again. Enter a number: 25
Try again. Enter a number: 1
Try again. Enter a number: 2
Try again. Enter a number: 5
[carora18@mtrx-node01pd ops]$ cat 100976240_week9_task3.bash
#!/bin/bash

read -p "Enter a number: " number
while [ $number -ne 5 ]

do
read -p "Try again. Enter a number: " number

   if [ $number -eq 5 ]
   then
     break

   fi

done

[carora18@mtrx-node01pd ops]$
```

Here's a breakdown of how the code works:

read -p "Enter a number: " number: This line prompts the user to enter a number and stores the input in the variable number.
while [ $number -ne 5 ]: This line starts a while loop that continues as long as the value of number is not equal to 5.
read -p "Try again. Enter a number: " number: Inside the loop, if the user enters a number other than 5, it prompts them again to enter a number, and the new input overwrites the previous value of number.
if [ $number -eq 5 ]: This if statement checks if the value of number is equal to 5.
break: If the value of number is indeed 5, the break statement is executed, which exits the loop.

code 2

```
#!/bin/bash
for i in {1..5}
do
   echo $i
done
```

```
[carora18@mtrx-node01pd ops]$ 100976240_week9_task3.bash
1
2
3
4
5
[carora18@mtrx-node01pd ops]$ cat 100976240_week9_task3.bash
#!/bin/bash
for i in {1..5}
do
    echo $i
done

[carora18@mtrx-node01pd ops]$
```

#!/bin/bash: This is the shebang line that indicates the path to the interpreter for the script, which in this case is /bin/bash.
for i in {1..5}: This line starts a for loop where i takes values from 1 to 5 (inclusive) using brace expansion {1..5}.
do: This keyword indicates the beginning of the code block that will be executed in each iteration of the loop.
echo $i: This line prints the value of i in each iteration of the loop.
done: This keyword indicates the end of the code block to be executed in the loop.

code 3:

```
read -p "Enter a number: " number
while [ $number -ne 5 ] || [ $number -ne 6 ]
do
echo "Not right "
read -p "Enter again " number
if [ $number -eq 5 ] || [ $number -eq 6 ]
then
echo "You got it right"
break
fi
done
echo "Correct guess"
```

```
[carora18@mtrx-node01pd ops]$ nano 100976240_week9_task3.bash
[carora18@mtrx-node01pd ops]$ 100976240_week9_task3.bash
Enter a number: 5
Not right
Enter again 6
You got it right
Correct guess
[carora18@mtrx-node01pd ops]$ 100976240_week9_task3.bash
Enter a number: 4
Not right
Enter again 8
Not right
Enter again 5
You got it right
Correct guess
[carora18@mtrx-node01pd ops]$ cat 100976240_week9_task3.bash

read -p "Enter a number: " number
while [ $number -ne 5 ] || [ $number -ne 6 ]
do
echo "Not right "
read -p "Enter again " number
if [ $number -eq 5 ] || [ $number -eq 6 ]
then
echo "You got it right"
break
fi
done
echo "Correct guess"

[carora18@mtrx-node01pd ops]$
```

Here's a breakdown of how the code works:

read -p "Enter a number: " number: This line prompts the user to enter a number and stores the input in the variable number.

while [ $number -ne 5 ] || [ $number -ne 6 ]: This line starts a while loop that continues as long as the value of number is not equal to 5 or not equal to 6. Note that this condition is logically incorrect. It should use logical AND (&&) instead of logical OR (||) to ensure that the loop continues only if the number is not 5 and not 6.

echo "Not right": Inside the loop, if the number entered by the user is not 5 or 6, it prints "Not right".

read -p "Enter again: " number: It prompts the user to enter a number again and updates the value of number with the new input.

if [ $number -eq 5 ] || [ $number -eq 6 ]: This if statement checks if the value of number is either 5 or 6. If the user guesses the correct number, it prints "You got it right" and breaks out of the loop using the break statement.

echo "Correct guess": After the loop, regardless of whether the user guessed the correct number or not, it prints "Correct guess".

carora18@mtrx-node01pd:~/ops

```
[carora18@mtrx-node01pd ops]$ cat 100976240_week9_task3.bash
read -p "Enter a number: " number
while [ $number -ne 5 ]
do
read -p "Try again. Enter a number: " number
   if [ $number -eq 5 ]
   then
     break
   fi
done


#!/bin/bash
for i in {1..5}
do
    echo $i
done

read -p "Enter a number: " number
while [ $number -ne 5 ] || [ $number -ne 6 ]
do
echo "Not right "
read -p "Enter again " number
if [ $number -eq 5 ] || [ $number -eq 6 ]
then
echo "You got it right"
break
fi
done
echo "Correct guess"

[carora18@mtrx-node01pd ops]$ cp 100976240_week9_task3.bash ~syed.misbahuddin/fall2023/week9
[carora18@mtrx-node01pd ops]$
```