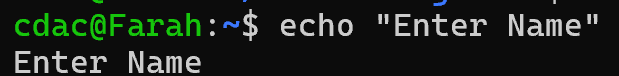
**Assignment 2**

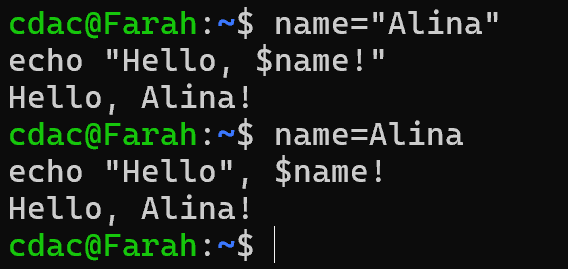
**Part A**

**What will the following commands do?**

• **echo "Hello, World!"**

The echo command in the shell is used to display a line of text or a variable’s value to the terminal.



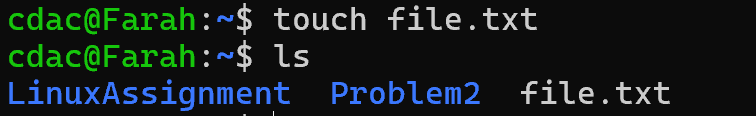


• **name="Productive"**

name is a variable and Productive is the assigned value to it.

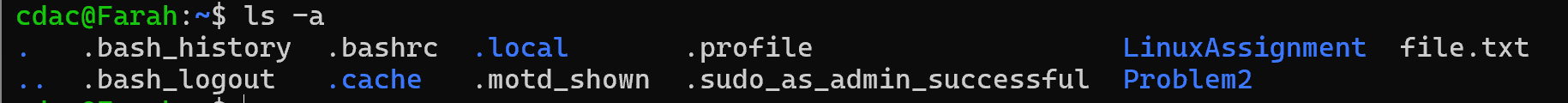
• **touch file.txt**

Used to create file



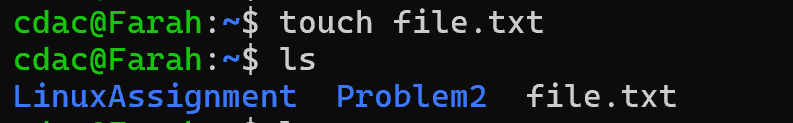
• **ls -a**

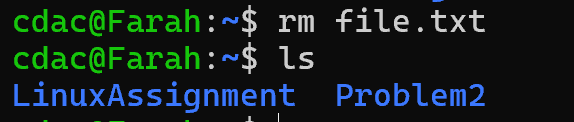
Display all the files and directories of that particular directory including hidden files also that start with dot(.)



**• rm file.txt**

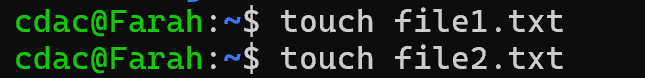
Removes the particular file.

****

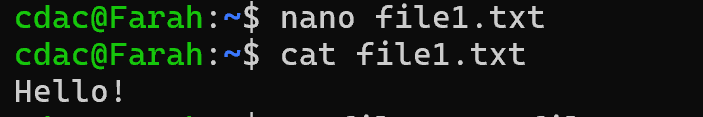
****

**• cp file1.txt file2.txt**

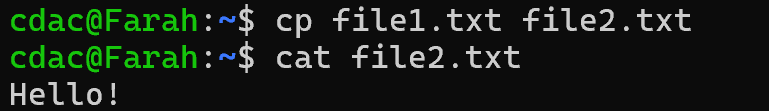
Creating 2 files named file1 and file2 respectively.



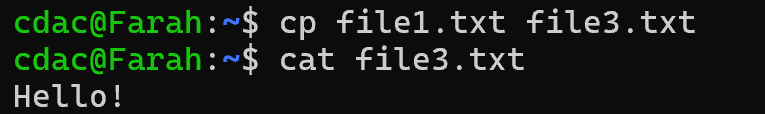
Edited file1 and displaying the content.



Copying file1 content to existing file2.

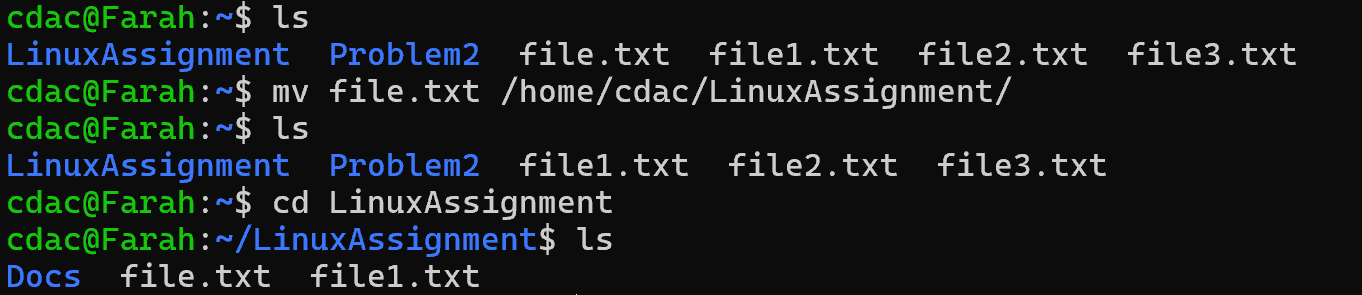


File3 doesnot exist so cp command is creating file3 and then copying the content.

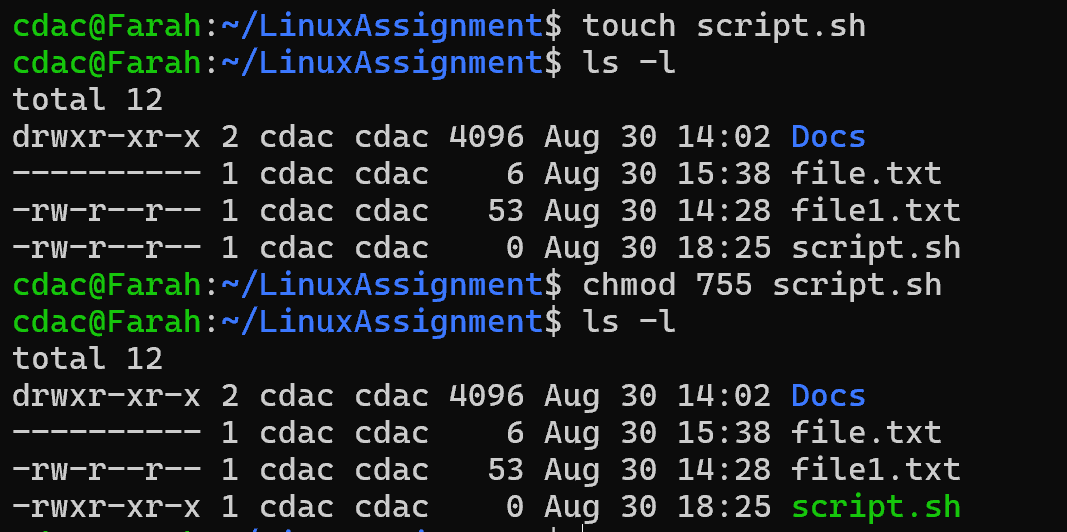


**• mv file.txt /path/to/directory/**

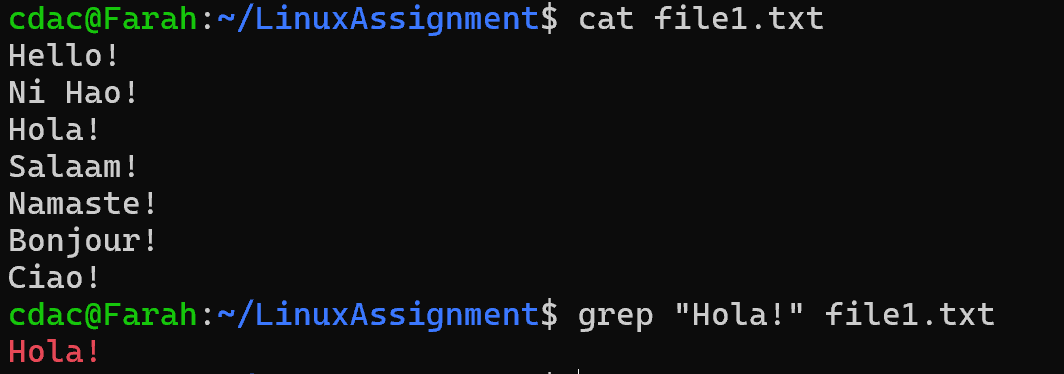
Moving the file from current location to specified directory.



**• chmod 755 script.sh**



**• grep "pattern" file.txt**

****

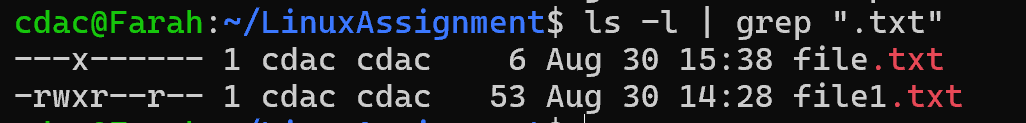
**• kill PID**

Used to terminate the process.

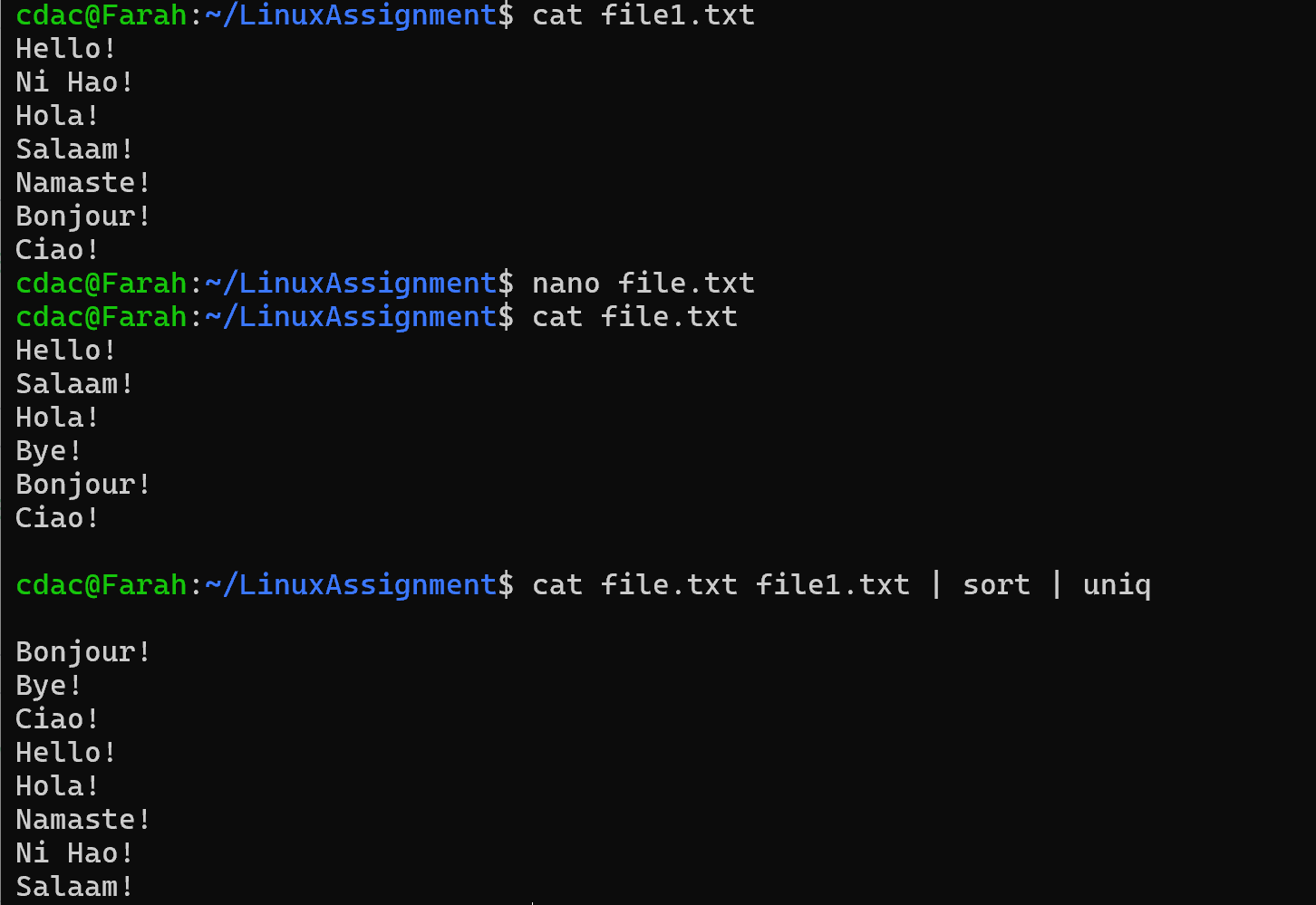
**• mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt**

**• ls -l | grep ".txt"**

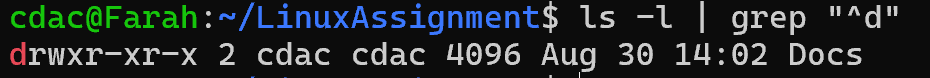
Filter out the .txt file

****

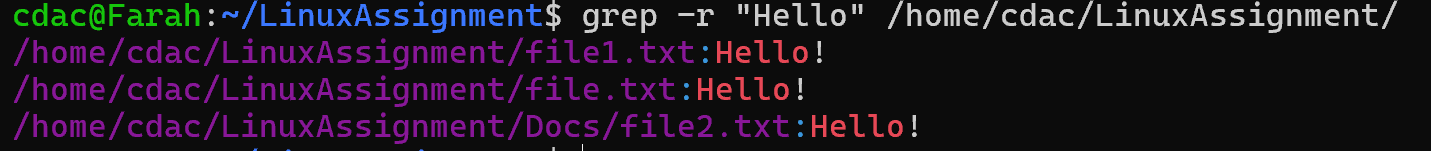
**• cat file.txt file1.txt | sort | uniq**

****

**• ls -l | grep "^d"**

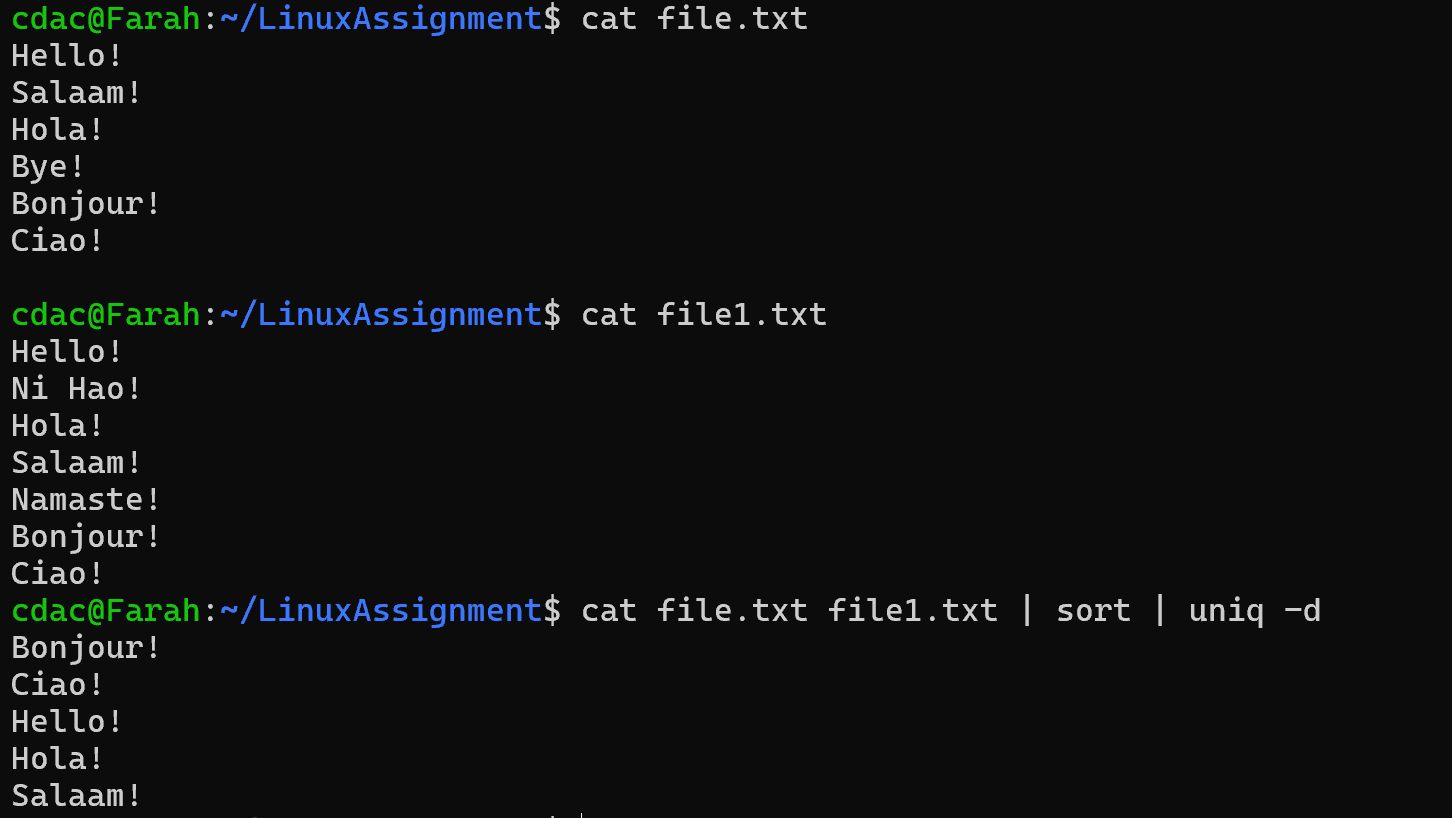
****

**• grep -r "pattern" /path/to/directory/**

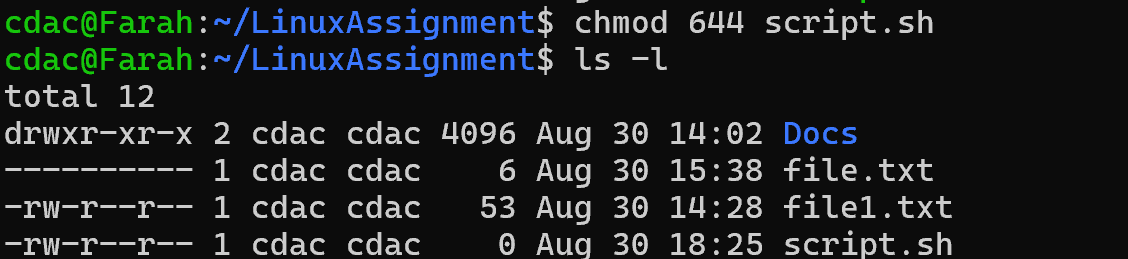
****

**• cat file1.txt file2.txt | sort | uniq –d**

**-d** : Displays only duplicate lines.

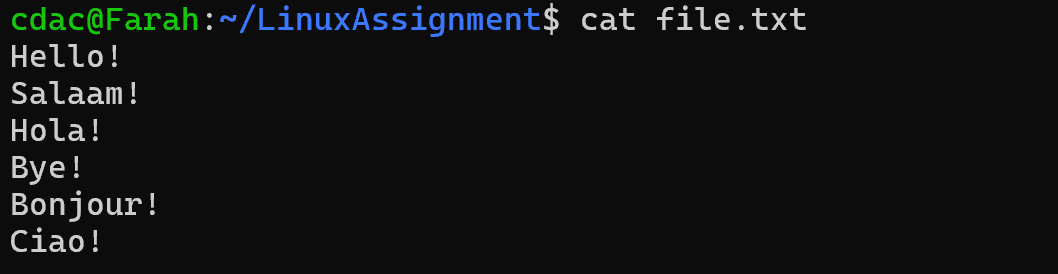
****

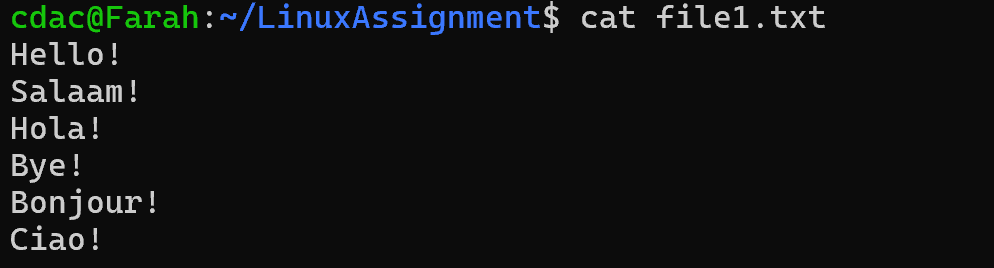
**• chmod 644 file.txt**

****

**• cp -r source\_directory destination\_directory**

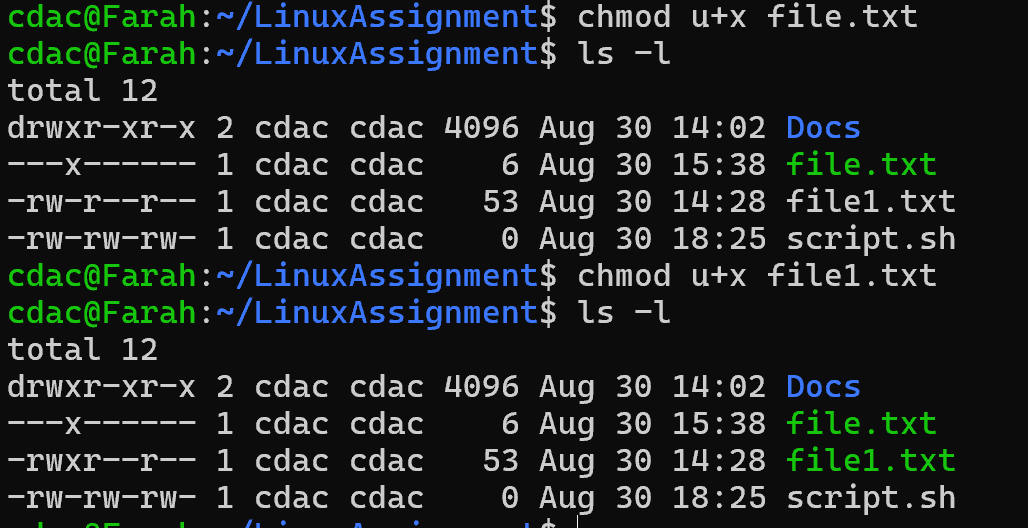
**cp -r file.txt file1.txt** - Copies directories and their contents recursively.

****

****

**• find /path/to/search -name "\*.txt"**

**• chmod u+x file.txt**

****

**• echo $PATH**

**Part B**

**Identify True or False:**

**1. ls** is used to list files and directories in a directory. **- True.**

**2. mv** is used to move files and directories. **- True.**

**3. cd** is used to copy files and directories. - **False.**

**4. pwd** stands for "print working directory" and displays the current directory. **- True.**

**5. grep** is used to search for patterns in files. **- True.**

**6. chmod 755 file.txt** gives read, write, and execute permissions to the owner, and read and execute permissions to group and others. **- True.**

**7. mkdir -p directory1/directory2** creates nested directories, creating directory2 inside directory1 if directory1 does not exist. **- True.**

**8. rm -rf file.txt** deletes a file forcefully without confirmation. **- True.**

**Identify the Incorrect Commands:**

1. **chmodx** is used to change file permissions. **– Incorrect**

* **chmod** is the correct command.

1. **cpy** is used to copy files and directories. **– Incorrect**

* **cp** is the correct command to copy.

1. **mkfile** is used to create a new file. **– Incorrect**

* **touch** command is used to create a new file or nano command is used to create a new file and add content to it and edit later also.

1. **catx** is used to concatenate files. **– Incorrect**

* **cat** is the correct command. It is used to concatenate files.

1. **rn** is used to rename files. **– Incorrect**

* **mv** command is used to rename files.

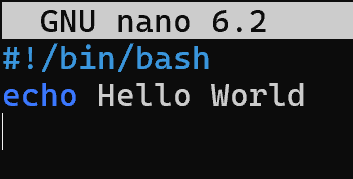
**Part C**

**Question 1:** Write a shell script that prints "Hello, World!" to the terminal.

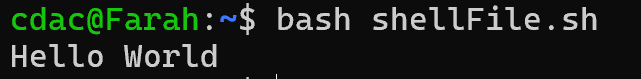
Created a new shell script file.



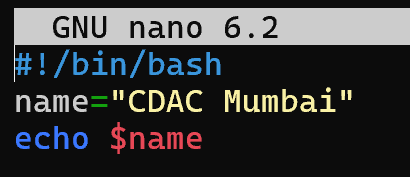
#!/bin/bash - to specify the script interpreter. Used echo to display.

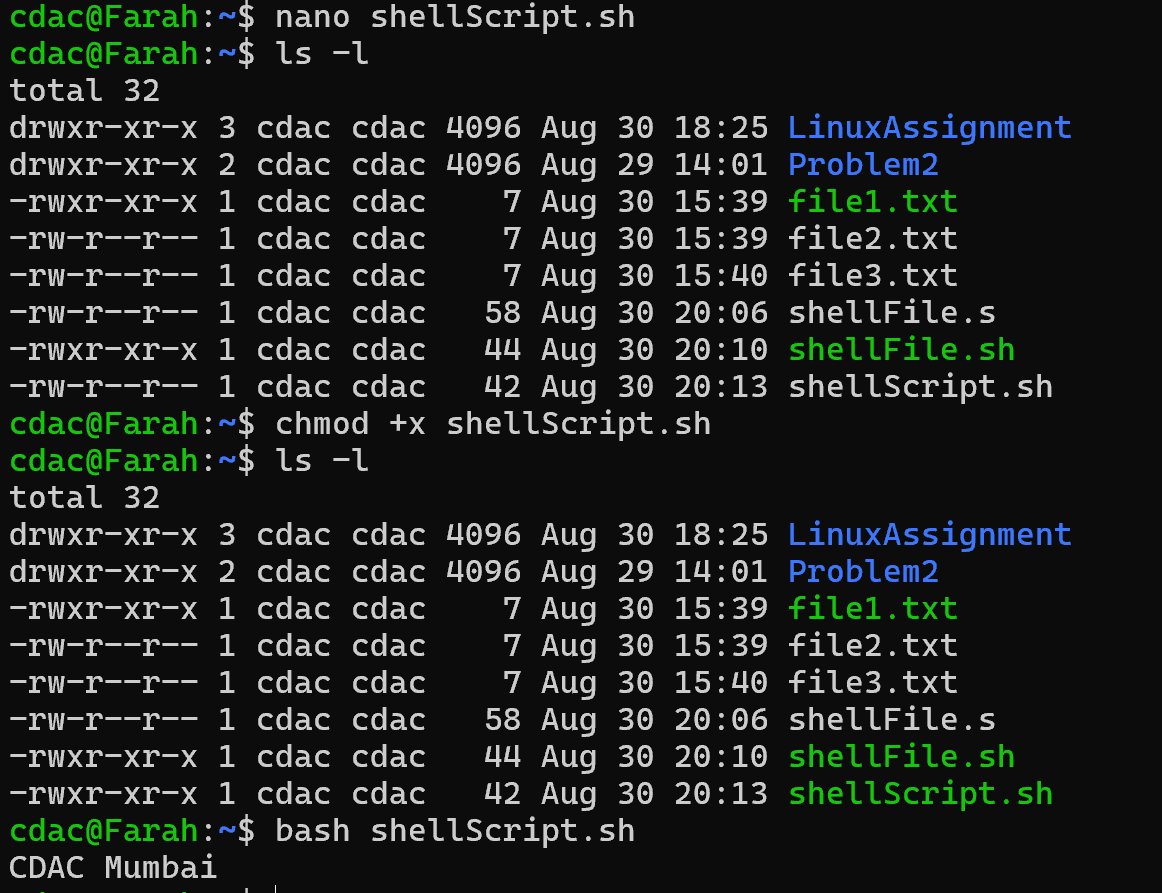


Getting the output.

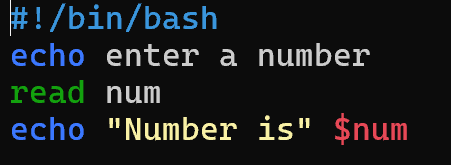


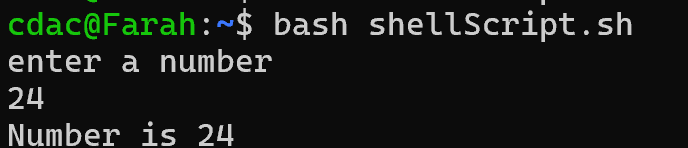
**Question 2:** Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.





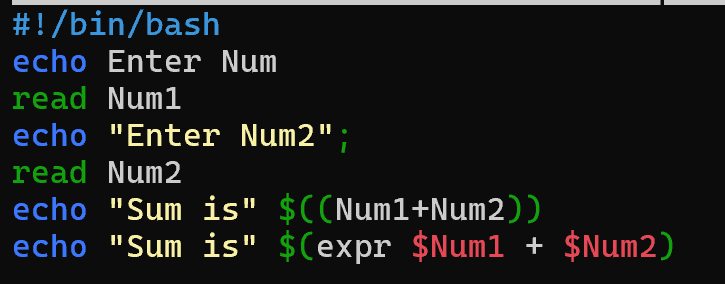
**Question 3:** Write a shell script that takes a number as input from the user and prints it.

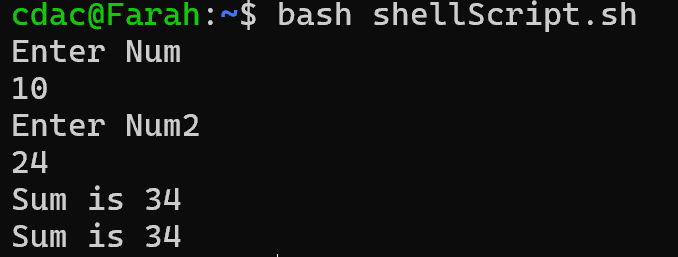




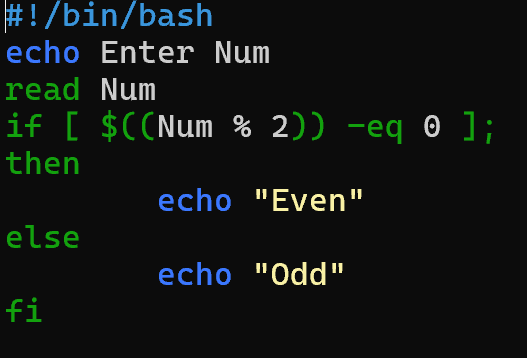
**Question 4:** Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.

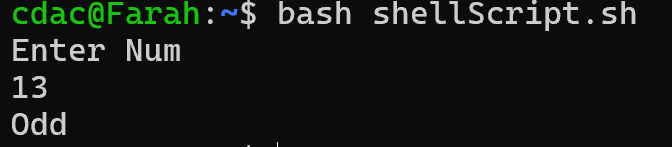
**Two ways to print sum of numbers:-**



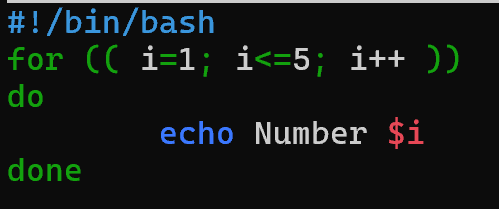


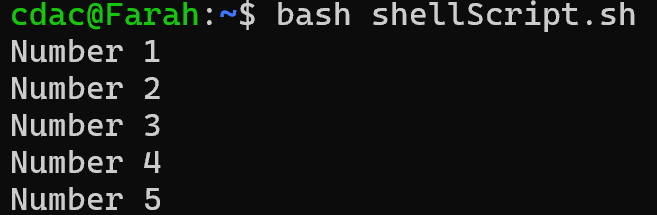
**Question 5:** Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".



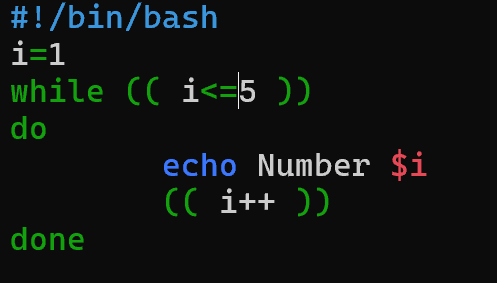


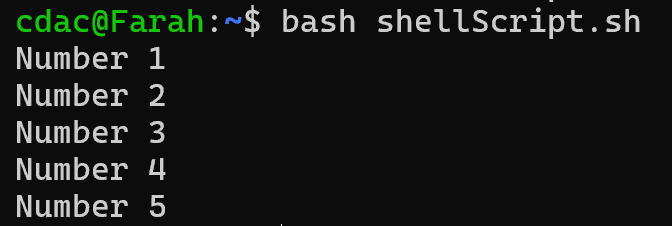
**Question 6:** Write a shell script that uses a for loop to print numbers from 1 to 5.



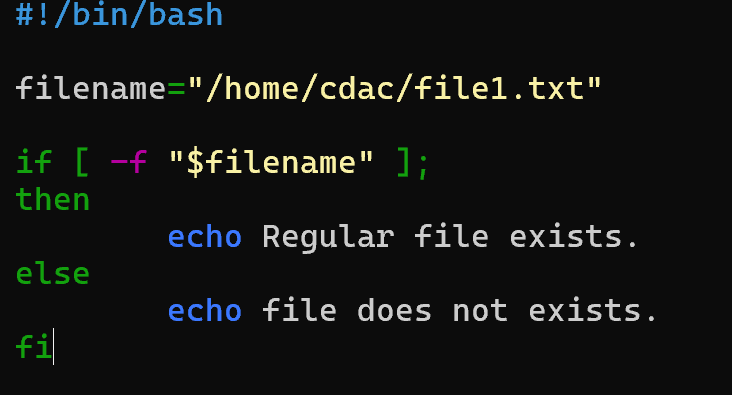


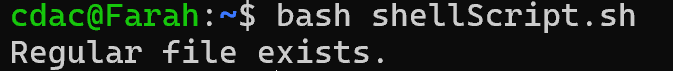
**Question 7:** Write a shell script that uses a while loop to print numbers from 1 to 5.





**Question 8:** Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".



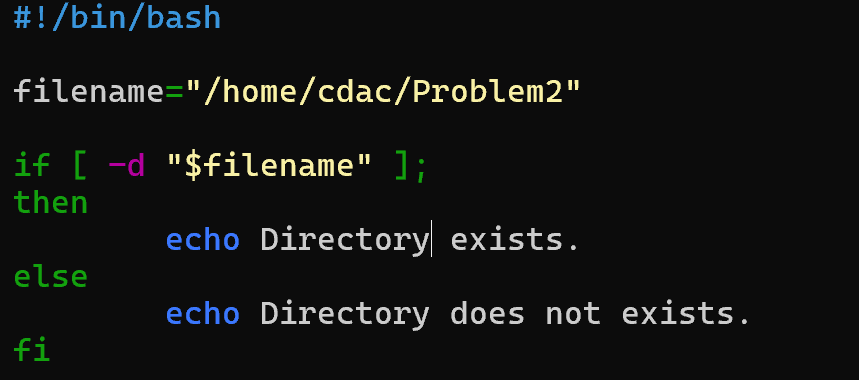


**[ -f "$filename" ]**: Checks if the file is a regular file.

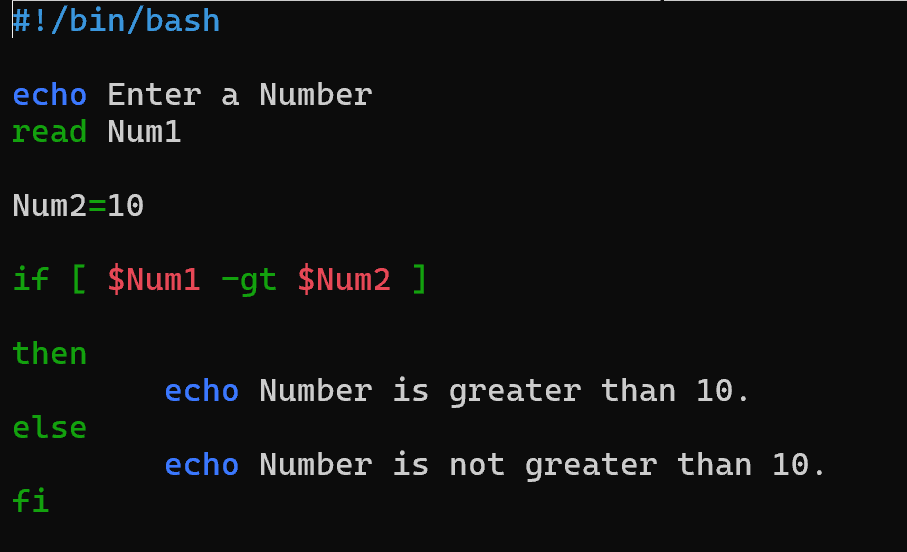
**[ -e "$filename" ]**: Checks if the file exists (any type of file).

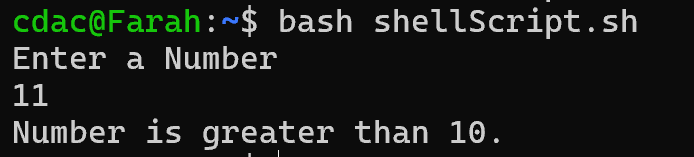
**[ -x "$file" ]**: Checks if the file is executable.

**[ -d "$directory" ]**: Checks if the path is a directory.

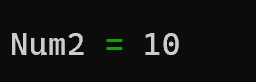


**Question 9:** Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.



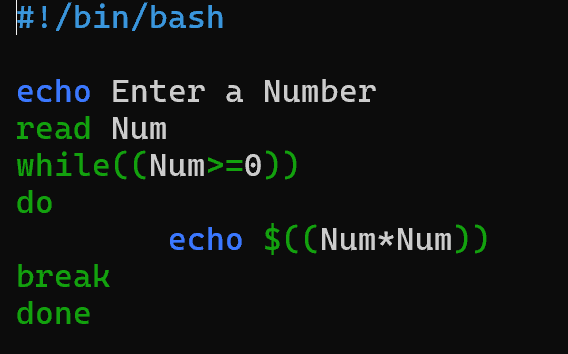


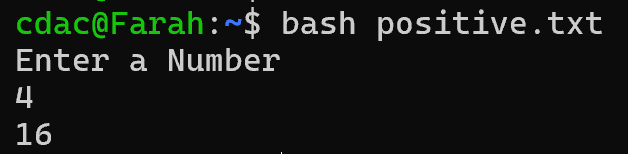
**Don’t give space Num2 = 0 like this. It will give error.**



**Question 10:** Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.

**Question 11:** Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered.





**Part E**

1. Consider the following processes with arrival times and burst times:

| Process | Arrival Time | Burst Time |

|------------|-----------------|------------|

| P1 | 0 | 5 |

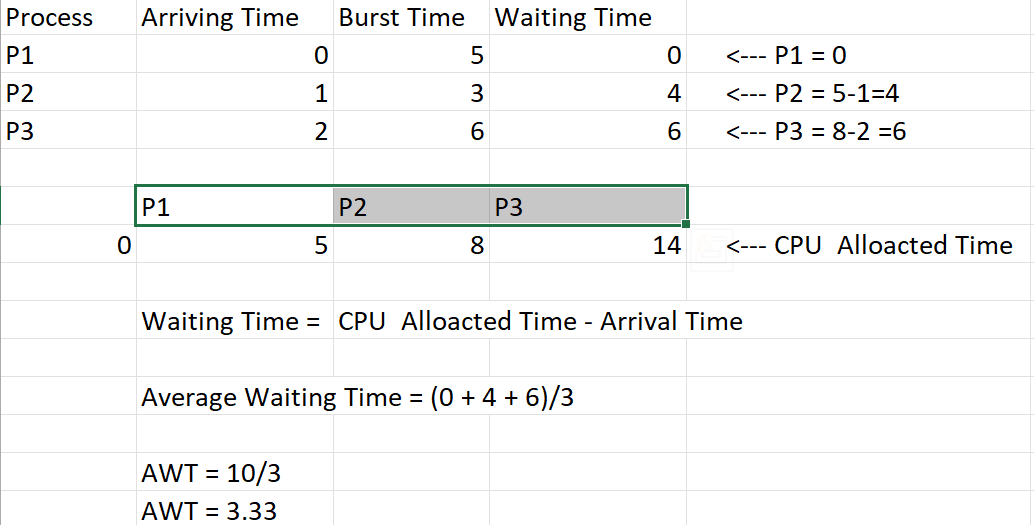
| P2 | 1 | 3 |

| P3 | 2 | 6 |

Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.

**Solution:-**

**First-Served (FCFS) Scheduling -**

****

1. Consider the following processes with arrival times and burst times:

| Process | Arrival Time | Burst Time |

|---------|--------|---------|

| P1 |0 | 3 |

| P2 | 1 | 5 |

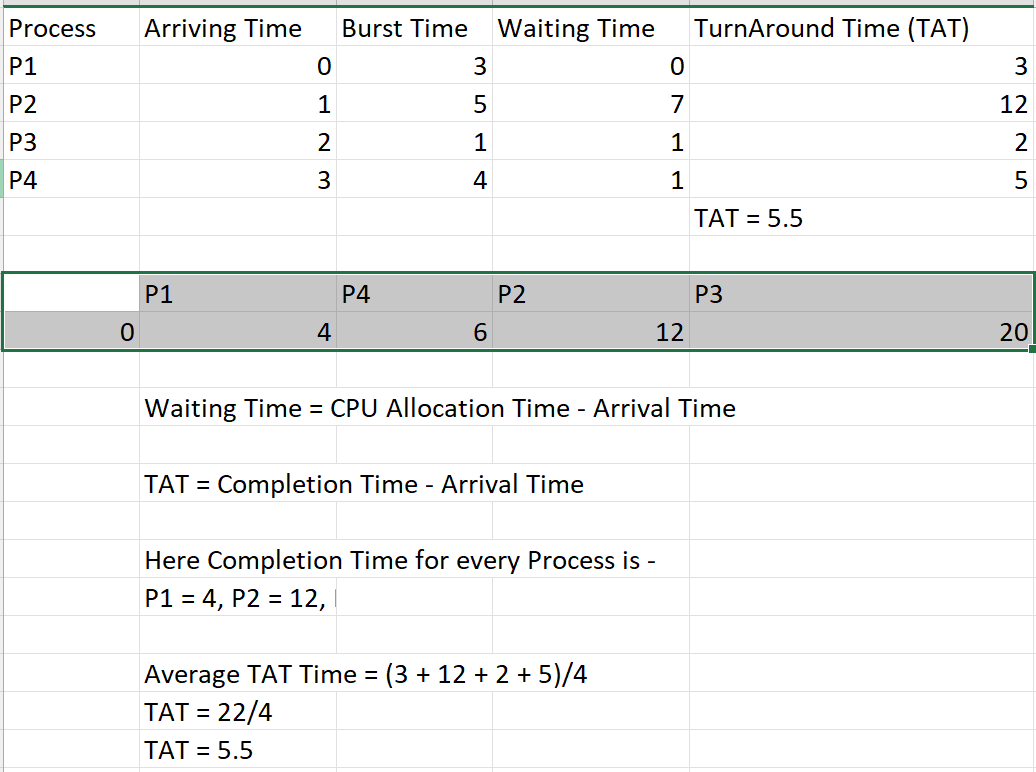
| P3 | 2 | 1 |

| P4 | 3 | 4 |

Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

**Solution:-**

**Shortest Job First (SJF) Scheduling –**



1. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

| Process | Arrival Time | Burst Time | Priority |

|---------|--------|--------|---------|

| P1 | 0 | 6 | 3 |

| P2 | 1 | 4 | 1 |

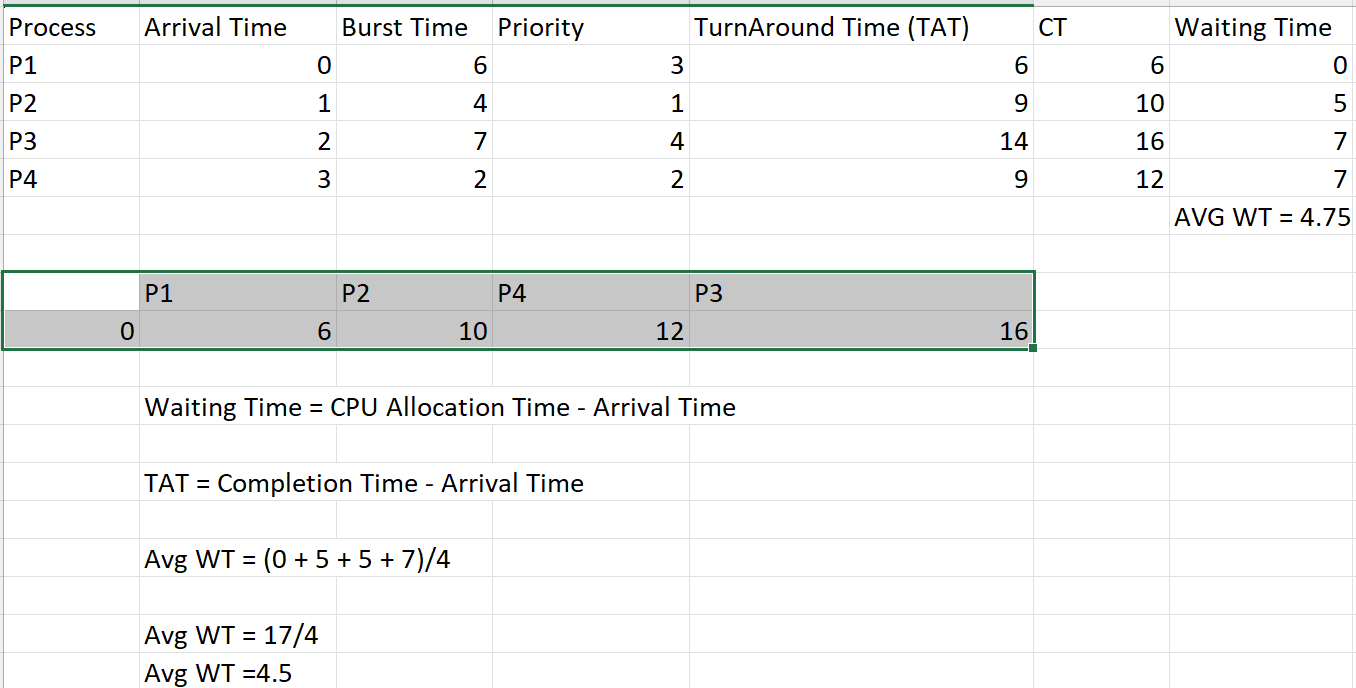
| P3 | 2 | 7 | 4 |

| P4 | 3 | 2 | 2 |

Calculate the average waiting time using Priority Scheduling.

**Solution:-**

**Priority Scheduling -**



1. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

| Process | Arrival Time | Burst Time |

|--------------|--------|--------|

| P1 | 0 | 4 |

| P2 | 1 | 5 |

| P3 | 2 | 2 |

| P4 | 3 | 3 |

Calculate the average turnaround time using Round Robin scheduling.

**Solution:-**

**Round Robin** **Scheduling –**

