Department of Computer Science

Gujarat University

Certificate

Roll No: 31 *Seat No:* _____

This is to certify that Ms. SHUBHAM RATHORE *student of MCA Semester – III has duly completed her term work for the semester ending in December 2020, in the subject of* OPERATING SYSTEM *towards partial fulfillment of her Degree of Masters in Computer Applications.*

10/12/2020
Date of Submission

Internal Faculty

Head of Department

Department Of Computer Science
Rollwala Computer Centre
Gujarat University

MCA –III

Subject: - OPERATING SYSTEM

Name :- SHUBHAM RATHORE H.

Roll No.: - 31 **Exam Seat No.: -**_____

Sr. No.	Contents	Pg. No	Date	Signature
	Assignment-1	1	10/12/2020	
	Glossary			
	Assignment-2	38	10/12/2020	
	Solve Numericals			

Date
Page 1

Assignment-1

1. **Base Address :-**

An address that is used as the origin in the calculation of address in the execution of a computer program.

2. **Batch Processing :-**

Pertaining to the technique of executing a set of computer programs such that the next program of the set is started.

3. 4

3. **Binary Semaphore**

A semaphore that takes on only 0 and 1. A binary semaphore allow only one process or thread to have access to a shared critical resource at a time.

4. **Block :-**

-) A collection of contiguous record that are recorded as a unit, those units are separated by interblock gaps.

-) A group of bits that are transmitted as a Unit.

Date
Page 2

5. Busy waiting :-

The repeated execution of a loop of coded code while waiting for an event to occur.

6. Cache Theory :-

A memory that is smaller and faster than main memory and that is interpose between the processor and main memory. The cache acts as a buffer for recently used memory locations.

7. Central Processing Unit (CPU) :-

That portion of a computer that fetch and executes instructions. It consists of an Arithmatic and logical unit (ALU), a control unit and registers, often simply referred to as a processor.

8. Cluster :-

A group of interconnected, whole computers working together as a unified computing resource that can create the illusion of being one machine. The term whole computer means a system that can run on its own, apart from the cluster.

9. Concurrent:

Pertaining to processes or threads that take place within a common interval of time during which they may have to alternately share common resource.

10. Consumable resource:

→ A resource that can be created and destroyed when a resource is acquired by a process the resource ceases to exist.

→ Example of consumable resources are; Interrupts, Signals, messages and information in I/O, buffers.

11. Deadlock:

→ An impose that occurs when multiple processes are waiting for the availability of a resource that will not become available because it is being held by another process that is in a similar wait state.

→ An impose that occurs when multiple processes are waiting for an action by or a response from another process

that is in a similar wait state.

12. Deadlock detection avoidance :-

→ technique in which requested resources are always granted when available. Periodically, the operating system test for deadlock.

13. Deadlock avoidance :

→ dynamic technique that examines each new resource request for deadlock. If the new request could lead to a deadlock, then the request is denied.

14. Deadlock Prevention :

→ technique that guarantees that a deadlock will not occur, prevention is achieved by assuring that of the ~~no~~ necessary conditions for deadlock is not met.

15. Device Driver :-

→ operating system module that deals directly with device or I/O module.

16. Direct access :=>

= The capability to obtain data from a storage device or to enter data into a storage device, in a sequence independent of their relative position, by means of addresses that indicate the physical location of the data.

17. Direct Memory Access (DMA):

A form of I/O in which a special module, called a DMA module, controls the exchange of data between main memory and an I/O device. The processor sends a request for the transfer of a block of data to the DMA module and is interrupted only after the entire block has been transferred.

18. Disable Interrupt:

A condition, usually created by the operating system, during which the processor will ignore interrupt request signals of a specific class.

19 Disk allocation table:

A table that indicates which blocks on secondary storage are free and available for allocation to files.

20 Distributed Operating System:

A common operating system shared by a network of computers. The distributed operating system provides support for interprocess communication, process migration, mutual exclusion and the prevention or detection of deadlock.

21 Dispatch

To allocate time on a processor to a computer program during execution. So that the program may be executed from a different area of main storage.

22 Dynamic Relocation:-

A process that assign new absolute address to a computer program during execution, so that the program may be executed from a different area of main storage.

23. Enabled interrupt:

A condition, usually created by the Operating system, during which the processor will respond to interrupt request signals of a specified class.

24. External Fragmentation:

-) Occured when memory is divided into variable-size partitions corresponding to the block of data assigned to the memory, gaps will occur between the occupied partitions of memory.

25. Field:

Defined logical data that are part of a record. The elementary Unit of a record that may contains a data item, a data aggregate, a pointer or a link.

26. File:

A set of related records treated as a Unit.

27. File Allocation Table (FAT)

A table that indicates the physical location on secondary storage of the space allocated to a file. There is one file allocation table for each file.

28. File Management System

A set of system software that provides services to users and applications in the use of files including, file access directory ~~access~~ maintenance and access control

29. File organization

→ The physical order of records in a file is as determined by the access method used to store and relatives them.

30. First Come First Served :

Same as FIFO

31. First In First Out :

A queueing technique in which the item

next item to be retrieved is the item that has been in the queue for the longest time.

32. Hash File :-

A File in which records are accessed according to the values of a key field. Hashing is used to locate a record on the basis of id's key value.

33. Hashing:

→ The section of a storage location for an item of data by calculating the address as a function of the ~~con de~~ contents of the data. This technique complicates a storage allocation function but result in rapid random retrieval

34. Hit ratio:-

→ In a two-level memory, the fraction of all memory accesses that are found in the faster memory.

35. Indexed access:

Pertaining to the organization and accessing of the records of a storage structure through a separate index to the location of the stored records.

36. Index File:

A file in which records are accessed according to the ~~required~~ the value of key fields. An index is required that indicates the location of each record on the basis of each key value.

37. Index sequential file:

A file in which records are ordered according to the values of a key field. The main file is supplemented with an index file that contains a partial list of key values, the index provides a lookup capability to quickly reach the vicinity of a desired record.

38. Instruction cycle:

The time period during which one instruction

is fetched from memory and executed when computer is given an instruction in machine language.

39. Internal Fragmentation.

Occurs when memory is divided into fixed size partitions. If a block of data is assigned to one or more partitions then there are may be wasted space in the last partition. This will occur if the last partition data is smaller than last partition.

40. Interrupt:

A suspension of a process such as the execution of a computer program caused by an event external to that process and performed in such a way that the process can be resumed.

41. Interrupt handler:

A routine, generally part of the operating system, when an interrupts occurs, control is transferred to the corresponding interrupt handler which take some action in response to the condition that caused the interrupt.

42. Job.

A set of computational steps packaged to run as a unit.

43. Kernal

→ A portion of the operating system that includes the most heavily used portion of software. Generally in main memory. The kernal runs in a privileged mode & responds to calls from process and interrupts from devices.

44. Kernal mode:

A privileged mode of execution reserved for the kernal of the operating system.

Typically, kernel mode allows access to regions of main memory that are unavailable to process executing in a less privileged mode, and also enables execution of certain machine instructions. Also referred to as system mode or privileged mode.

45 LIFO Last In First Out:

→ A queueing technique in which the next item to be retrieved is the item most recently placed in the queue.

46 Live lock:

A condition in which two or more processes continuously change their state in response to changes in the other process without doing any work. This is similar to deadlock in that no progress made but it differs in that neither deadlock nor process is blocked or waiting for anything.

48. live lock.

49. logical address :

A reference to memory location indepe
ndent of the current assignment
of data to memory. A translation
must be made to a physical address
before the memory access can be
achieved.

50. logical record :-

-> A record is independent of it's
physical environment portion of one
logical record may be located in
different physical records or several
logical records or parts of logical records
may be located in one physical record.

49. Main memory :-

Memory that is internal to the computer
system is program addressable and can
be loaded in to register for subsequent
execution or processing.

Q5 Malicious Software:

→ Any software designed to damage or use up the resources of a target computer. Malicious software can be cancelled with an antivirus or keystroke software.

b1. Memory cycle time:

The time it takes read or write one word to memory. This is the inverse of the rate at which words can be read from or write to memory.

b2. Memory partitioning:

The subdividing of storage into independent sections.

b3. Micro kernal:

A small privileged operating system core that provides process scheduling, memory management and communication. Service and relies on other processes to perform some of the functions traditionally

associated with the operating system kernel.

55 Multiprocessing :-

A mode of operation that provides for the interleaved execution of two or more computer programs by a single processor. The same as multitasking using different terminology.

56 Multi Programming :

A mode of operation that provides for parallel processing by two or more processors of a multiprocessor.

57 Multiprogramming level :

-> The number of processes that are partially or fully resident in main memory.

58 Multitasking :-

A mode of operation that provides for the concurrent performance or interleaved

execution of two or more computer tasks. The same as multiprogramming using different terminology.

5) Mutual exclusion :-

→ A condition in which there is a set of processes, only one of which is able to access a given resource or program at given function at any time, see Critical section.

6) Operating System :-

Software that controls the execution of programs and that provides services such as resource allocation, scheduling input/output control and data management.

6) Page :

In virtual storage, a fixed-length block that has a virtual address and that is transferred as a unit between main memory and

Date
Page 18

secondary memory.

2. Page Fault:

- occurs when the page containing a referenced word is not in main memory. This causes an interrupt and requires that the proper page be brought into main memory.

3. Page Frame:

A fixed size of contiguous block of main memory used to hold a page.

4. Paging:

The transfer of pages between main memory and secondary memory.

5. physical address:

The absolute location of a data in memory.

66 Pipe:

A circular buffer allowing two processes to communicate on the producer consumer model.

67 Preemption:

Reclaiming a resource from a process before the process has finished using it.

68 Prepaging:-

The retrieval of the pages other than the one demanded by a page fault. The hope is that the additional pages will be needed in the near future, conserving the disk I/O, Compare demand paging.

69 Process:

A program in execution A process is controlled and scheduled by the operating system same as task.

Date
Page 20

20 Process Control block

The manifestation of a process in an OS. It is a data structure containing info about the characteristics and the states of the process.

21 Process state:

All of the info that an os needs to manage the process and that the processor needs to properly execute the process.

22 Processor:-

In a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and arithmatic and logical unit.

23 Program Counter:-

Instruction address register.

Date
Page 21

24 Programmed I/O

→ A form of I/O in which the CPU issues an I/O command to an I/O module and must then wait for the operation to be completed before processing.

25 Real time system.

An OS that must schedule and manage real time tasks

26 Registers

High speed memory internal to the CPU. Some registers are user visible that is available to the programmer via the machin instruction set.

27 Relative address

An address calculated as displacement from base address

78 Remote Procedure Call (RPC)

A technique by which two program on different machine interact using remote procedure call / return syntax and semantics.

79 Response time

In a data system, the elapsed time between the end of the transmission of an enquiry message and the beginning of reciepde of response message measured at the inquiry terminal

80 Round Robin:

A scheduling algorithm in which processes are activated in a fixed cyclic order that is all processes are in a circular queue.

81 Scheduling:

To select jobs or tasks that are to be ~~executed~~ dispatched.

82 Secondary memory :→

memory located outside the computer itself.

83 Segment :

→ In a virtual memory a block that has a virtual address. The blocks of a program may be unequal length and may even be of dynamically varying length.

84 Segmentation :

The devision of a program or application into segments as part of a virtual memory scheme.

85 Semaphore :

An integer value that is used for signaling among processes

86 Sequential File

A File in which records are ordered according to the values

of one or more key fields and processed in the same sequence from the beginning of the file.

87. Shell

The portion of the OS that interrupts interactive user commands and job control language commands.

88. Stack

Stack is LIFO

89. Starvation

A condition in which a process is indefinitely delayed because other processes are always given preference.

90. Strong semaphore:

A semaphore in which all processes are waiting on the same semaphore are queued and will eventually processed in the same order as they executed the wait operations.

* Assignment-2 *

Date
Page 25

Banker's Algorithm

Process	Allocation			Max			(work) Available			(max-allo) Need		
	A	B	C	A	B	C	A	B	C			
P0	0	1	0	7	5	3	3	3	2			
P1	2	0	0	3	2	2						
P2	3	0	2	9	0	2						
P3	2	1	1	2	2	2						
P4	0	0	2	4	3	3						

→

Process	Allocation			Max			Available			~~Max~~ Need		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2	7	4	3
P1	2	0	0	3	2	2				1	2	2
P2	3	0	2	9	0	2				6	0	0
P3	2	1	1	2	2	2				0	1	1
P4	0	0	2	4	3	3				4	3	1

→ Need ≤ work ⇒ work = work + allocation

P0 743 ≤ 332 condition ~~true~~ false

work = work + allocation

= 332 + 200

= 532

P1 322 ≤ 332 condition true

work = work + allocation

= 332 + 200

= 532

P2 Need ≤ work

600 ≤ 532 condition false

→ P_3 Need ≤ work

011 ≤ 532 condition true

work = work + allocation

= 532 + 211

= 743

→ P_4 Need ≤ work

431 ≤ 743 condition true

work = work + allocation

= 743 + 002

= 745

→ P_0 Need ≤ work

743 ≤ 745 condition true

work = work + allocation

= 745 + 010

= 755

→ P_2 Need ≤ work

600 ≤ 755 condition true

work = work + allocation

= 755 + 342

= 1052

Safe sequence is → < P1, P3, P4, P0, P2 >

* FIFO

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0 1

Date
Page 20

7	0	1	2	0	3	0	4	2	3	0
7	7	7	2		2	2	4	4	4	0
	0	0	0		3	3	3	2	2	2
		1	1		1	0	0	0	3	3

3	0	3	2	1	7	0	1	2
				0	~~0~~	7	7	0
				1	1	1	0	1
				3	2	2	2	2

0	1	7	0	1
		7	7	7
		1	0	0
		2	2	1

page fault = 15 no of frames = 3

* LRU:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

7	0	1	2	0	3	0	4	2	3	0
7	7	7	2		2		4	4	4	0
	0	0	0		0		0	0	3	3
		1	1		3		3	2	2	2

3	0	3	2	1	2	0	1	7	0	1
				1		1		1		
				3		0		0		
				2		2		7		

no of frames = 3
page fault = 12

Date
Page 28

* Optional:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

7	0	1	2	0	3	0	4	2	3	0
7	7	7	2		2		2			2
	0	0	0		0		4			0
		1	1		3		3			3

3	0	3	2	1	2	0	1	7	0	1
				2				7		
				0				0		
				1				1		

mod frames = 3
page fault = 9

⇒ FIFO
1, 3, 0, 3, 5, 6, 3

1	3	0	3	5	6	3
1	1	1		5	5	5
	3	3		3	6	6
		0		0	0	3

page fault = 6 No of frames
No of frames = 3

Optional:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3.

7	0	1	2	0	3	0	4	2	3	0	3	2	3
7	7	7	7		3		3						
	0	0	0		0		0						
		1	1		1		4						
			2		2		2						

page fault = 6
no of frames = 4

#1. Basic salary of a person is input through the keyboard. His dearness allowance is 40% of basic

#salary and house rent is 20% of basic salary. Write a program to calculate the gross pay.

```
read -p "Enter Your Salary : " salary

dearance=`echo "$salary*40/100"|bc`

houserent=`echo "$salary*20/100"|bc`

gross=`echo "$salary-$houserent+$dearance"|bc`

echo "Dearness Allowance : "$dearance

echo "House Rent : "$houserent

echo "Gross Salaray : "$gross
```

#output:

#13000

#5200

#2600

#5200

****

#2. The distance between two cities is input through the keyboard. (in km). Write a program to

#convert this distance into metres, feet, inches and centimeters and display the results.

```
read -p "Enter Distance between 2 cities in kms : " kilometer

echo "Meters : "$((kilometer*1000))

echo "Feet : " $(echo $kilometer*3280.84 | bc)

echo "Inches : " $(echo $kilometer*39370.08 | bc)
```

```
echo "centimeters : " $((kilometer*100000))


#Enter Distance between 2 cities in kms : 45

#Meters : 45000

#Feet :  147637.80

#Inches :  1771653.60

#centimeters :  4500000

********************************************************************************
********************************************************************************
******************

#3. The length and breadth of a rectangle and radius of a circle are entered through the keyboard,

#calculate the perimeter and area of rectangle and area and circumference of the circle.


PI=3.14159

read -p "Enter Length And Breadth of Rectangle : " length breadth

read -p "Enter Radius of circle : " radius

echo "Perimeter of the Rectangle is : " $((2*length + 2*breadth))

echo "Area of Rectangle is : " $((length*breadth))

echo "Circumference of Circle : " $(echo 2.0*$PI*$radius | bc)

echo "Area of circle : " $(echo $PI*$radius^2 | bc)


#Enter Length And Breadth of Rectangle : 5 6

#Enter Radius of circle : 7

#Perimeter of the Rectangle is :  22

#Area of Rectangle is :  30
```

#Circumference of Circle : 43.98226

#Area of circle : 153.93791

****
****
****

#4. If a five digit number is entered through the keyboard, calculate the sum of its digits.

```
total=0
read -p "Enter Number : " num
length=$(echo -n $num | wc -c)
i=1
while [ $i -le $length ]
do
	remainder=$(echo $num%10 | bc)
	total=$(echo $total+$remainder | bc)
	num=$(echo $num/10 | bc)
	i=$(echo $i+1 | bc)
done
echo "Total : " $total
```

#Output :

#Enter Number : 123

#Total : 6

****
****
****

#5. The file /etc/passwd contains info about all users. Write a program which would receive the

#logname during execution, obtain information about it from the file and display the

#information on screen in some appropriate format. (Hint : use cut)

eg. Logname : , UID : , GID : , Default working directory : , Default working shell :

filepath="/etc/passwd"

cut -d ":" -f 1,3,4,5,6,7 $filepath --output-delimiter=' | '

#Output

#root | 0 | 0 | root | /root | /bin/bash

#daemon | 1 | 1 | daemon | /usr/sbin | /usr/sbin/nologin

#bin | 2 | 2 | bin | /bin | /usr/sbin/nologin

#sys | 3 | 3 | sys | /dev | /usr/sbin/nologin

#sync | 4 | 65534 | sync | /bin | /bin/sync

#games | 5 | 60 | games | /usr/games | /usr/sbin/nologin

#man | 6 | 12 | man | /var/cache/man | /usr/sbin/nologin

#lp | 7 | 7 | lp | /var/spool/lpd | /usr/sbin/nologin

#mail | 8 | 8 | mail | /var/mail | /usr/sbin/nologin

#news | 9 | 9 | news | /var/spool/news | /usr/sbin/nologin

#uucp | 10 | 10 | uucp | /var/spool/uucp | /usr/sbin/nologinv

#proxy | 13 | 13 | proxy | /bin | /usr/sbin/nologin

#www-data | 33 | 33 | www-data | /var/www | /usr/sbin/nologin

#backup | 34 | 34 | backup | /var/backups | /usr/sbin/nologin

#list | 38 | 38 | Mailing List Manager | /var/list | /usr/sbin/nologin

#irc | 39 | 39 | ircd | /var/run/ircd | /usr/sbin/nologin

#gnats | 41 | 41 | Gnats Bug-Reporting System (admin) | /var/lib/gnats | /usr/sbin/nologin

#nobody | 65534 | 65534 | nobody | /nonexistent | /usr/sbin/nologin

#systemd-network | 100 | 102 | systemd Network Management,,, | /run/systemd | /usr/sbin/nologin

#systemd-resolve | 101 | 103 | systemd Resolver,,, | /run/systemd | /usr/sbin/nologin

#systemd-timesync | 102 | 104 | systemd Time Synchronization,,, | /run/systemd | /usr/sbin/nologin

#messagebus | 103 | 106 | | /nonexistent | /usr/sbin/nologin

#syslog | 104 | 110 | | /home/syslog | /usr/sbin/nologin

#_apt | 105 | 65534 | | /nonexistent | /usr/sbin/nologin

#tss | 106 | 111 | TPM software stack,,, | /var/lib/tpm | /bin/false

#uuidd | 107 | 114 | | /run/uuidd | /usr/sbin/nologin

#tcpdump | 108 | 115 | | /nonexistent | /usr/sbin/nologin

#avahi-autoipd | 109 | 116 | Avahi autoip daemon,,, | /var/lib/avahi-autoipd | /usr/sbin/nologin

#usbmux | 110 | 46 | usbmux daemon,,, | /var/lib/usbmux | /usr/sbin/nologin

#rtkit | 111 | 117 | RealtimeKit,,, | /proc | /usr/sbin/nologin

#dnsmasq | 112 | 65534 | dnsmasq,,, | /var/lib/misc | /usr/sbin/nologin

#cups-pk-helper | 113 | 120 | user for cups-pk-helper service,,, | /home/cups-pk-helper | /usr/sbin/nologin

#speech-dispatcher | 114 | 29 | Speech Dispatcher,,, | /run/speech-dispatcher | /bin/false

#avahi | 115 | 121 | Avahi mDNS daemon,,, | /var/run/avahi-daemon | /usr/sbin/nologin

#kernoops | 116 | 65534 | Kernel Oops Tracking Daemon,,, | / | /usr/sbin/nologin

#saned | 117 | 123 | | /var/lib/saned | /usr/sbin/nologin

#nm-openvpn | 118 | 124 | NetworkManager OpenVPN,,, | /var/lib/openvpn/chroot | /usr/sbin/nologin

#hplip | 119 | 7 | HPLIP system user,,, | /run/hplip | /bin/false

#whoopsie | 120 | 125 | | /nonexistent | /bin/false

#colord | 121 | 126 | colord colour management daemon,,, | /var/lib/colord | /usr/sbin/nologin

#geoclue | 122 | 127 | | /var/lib/geoclue | /usr/sbin/nologin

#pulse | 123 | 128 | PulseAudio daemon,,, | /var/run/pulse | /usr/sbin/nologin

#gnome-initial-setup | 124 | 65534 | | /run/gnome-initial-setup/ | /bin/false

#gdm | 125 | 130 | Gnome Display Manager | /var/lib/gdm3 | /bin/false

#shivangi27 | 1000 | 1000 | shivangi,,, | /home/shivangi27 | /bin/bash

#systemd-coredump | 999 | 999 | systemd Core Dumper | / | /usr/sbin/nologin

#vboxadd | 998 | 1 | | /var/run/vboxadd | /bin/false

****
****
****

#6. The script will receive the filename or filename with its full path, the script should

#obtain information about this file as given by "ls -l" and display it in proper format.

eg. Filename : , File access permissions : , Number of links : , Owner of the file : ,

#Group to which belongs : Size of file : , File modification date : , File modification time :

echo "---------- Using stat----------"

stat -c "%A %h %U %G %s %.19y %n" *m

```
printf "\n---------- Using ls -l ----------\n"


ls -l | tr -s ' ' | cut -d ' ' -f 1-9 m -s | awk '{print $1,$2,$3,$4,$5,$6,$7,$8,"\033[32;1m"$9"\033[0m"}'

echo ""

ls -l | tr -s ' ' |cut -d ' ' -f 1-9 -s | awk '{print $1" | "$3" | "$4" | "$5" | "$6,$7" | "$8" |\033[32;1m "$9 "\033[0m"}'


#cut: m: No such file or directory


#total |  |  |  |  |  |

#-rw-rw-r-- | shubham31 | shubham31 | 155 | Dec 2 | 13:44 | circlePerimiter.sh

#-rw-rw-r-- | shubham31 | shubham31 | 61 | Nov 20 | 20:41 | data.dat

#drwxrwxr-x | shubham31 | shubham31 | 4096 | Nov 20 | 20:35 | dir

#drwxr-x--x | shubham31 | shubham31 | 4096 | Nov 20 | 20:52 | dir2

#drwxr-x--x | shubham31 | shubham31 | 4096 | Nov 20 | 20:28 | dir3

#-rw-rw-r-- | shubham31 | shubham31 | 265 | Dec 2 | 09:07 | done

#-rw-rw-r-- | shubham31 | shubham31 | 22 | Dec 2 | 08:56 | f1

#-rw-rw-r-- | shubham31 | shubham31 | 22 | Dec 2 | 08:57 | f3

#-rw-rw-r-- | shubham31 | shubham31 | 86 | Dec 2 | 09:38 | file

#-rw-rw-r-- | shubham31 | shubham31 | 448 | Dec 2 | 09:21 | file3

#-rw-rw-r-- | shubham31 | shubham31 | 505 | Dec 2 | 09:21 | file4

#-rw-rw-r-- | shubham31 | shubham31 | 209 | Dec 1 | 23:30 | grosspay.sh

#-rw-rw-r-- | shubham31 | shubham31 | 0 | Dec 2 | 09:18 | head

#-rw-rw-r-- | shubham31 | shubham31 | 22 | Dec 2 | 08:56 | l1

#-rw-rw-r-- | shubham31 | shubham31 | 25 | Dec 2 | 14:09 | LargestSmallest.sh
```

#-rw-rw-r-- | shubham31 | shubham31 | 741 | Dec 9 | 12:00 | oddeven.sh

#-rw-rw-r-- | shubham31 | shubham31 | 86 | Dec 2 | 09:36 | opfile

#-rw-rw-r-- | shubham31 | shubham31 | 120 | Dec 2 | 09:39 | t1

#-rw-rw-r-- | shubham31 | shubham31 | 49152 | Dec 2 | 14:05 | typescript

#-rw-rw-r-- | shubham31 | shubham31 | 86 | Dec 2 | 09:35 | xaa

****
****
****

#7. If cost price and selling price of an item are entered through the keyboard, write a program

#to determine whether the seller has made profit or loss. Also determine how much profit/loss

#is made.

```
read -p "Enter The Cost Price For Product : " cost
read -p "Enter The Selling price for Product : " selling
if [ $cost -lt $selling ]
then
    echo "Profit : " $((selling-cost))
elif [ $cost -gt $selling ]
then
    echo "Loss : " $((selling-cost))
else
    echo "No Profit No Loss"
fi
```

#OUTPUT

#Enter The Cost Price For Product : 8000

#Enter The Selling price for Product : 9000

#Profit : 1000

****
****
****

#8. Check whether the entered no. is odd or even.

```
read -p "Enter The Number : " num
check=$(echo $num%2 | bc)
if [ $check -eq 0 ]
then
    echo "EVEN"
els
    echo "ODD"
fi
```

#Output :

#Enter The Number : 23

#ODD

****
****
****

#9. Check whether the entered no. is prime or not.

```
read -p "Enter the Number : " prime
i=2
f=0
while test $i -le `expr $prime / 2`
do
        if test `expr $prime % $i` -eq 0
        then
                f=1
        fi
        i=`expr $i + 1`
done

if test $f -eq 1
then
        echo "Not Prime"
else
        echo "Prime"
fi

#Output :
#Enter the Number : 7
#Prime
```

****
****
****

#10. Check whether the entered year is a leap year or not.

```
read -p "Enter The Year : " year
if test `expr $year % 400` -eq 0
then
        echo "Its Leap Year"
elif test `expr $year % 100` -eq 0
then
        echo "Its Not A Leap Year"
elif test `expr $year % 4` -eq  0
then
        echo "Its Leap Year"
else
        echo "Its Not A Leap Year"
fi
```

#Output :

#Enter The Year : 2014

#Its Not A Leap Year

#Enter The Year : 2016

#Its Leap Year

****
****
****

#11. The script receives two file names as arguments, the script must check whether the #files are

#same or not, if they are similar then delete the second file.

```
if [ ! -f $1 ]; then
    echo "$1 not found!"
    exit
fi

if [ ! -f $2 ]; then
    echo "$2 not found!"
    exit
fi

my_var=$(cmp -b $1 $2)
if test -z "$my_var"
then
    echo "Files are same"
    rm $2
    echo $2 "Deleted"
else
    echo "Files are not same"
```

fi

#OUTPUT

#$ ls

#circlePerimiter.sh dir dir3 f1 file file4 head LargestSmallest.sh opfile typescript

#data.dat dir2 done f3 file3 grosspay.sh l1 oddeven.sh t1 xaa

#sh oddeven.sh file file3

#cmp: EOF on file after byte 86, line 4

#Files are same

#file3 Deleted

****

#14. Write a shell script to display the date with the format :- 25th October 2005 is a #Tuesday.

d=`date +%d\ %B\ %Y\ is\ a\ %A`

echo $d

#Output :

#01 December 2020 is a Tuseday

****

#15. Write a shell script to display the appropriate message like : Good Morning / Good #Afternoon

#/ Good Evening

```
hour=`date +%H`
if [ $hour -ge 21 ]
then
    echo "Good Night"
elif [ $hour -ge 16 ]
then
    echo "Good Evening"
elif [ $hour -ge 12 ]
then
    echo "Good Afternoon"
else
    echo "Good Morning"
fi
```

#Output:

#Good Night

****

#16. Write a shell script to display the menu driven interface :-

```
#1) list all files of the current directory
#2) print the current directory
#3) print the date
#4) print the users otherwise
#display "Invalid Option".



echo "1) List all Files"
echo "2) Print Current Directory"
echo "3) Print Date"
echo "4) Print Users"

read choice

case $choice in
    1)  ls
       ;;
    2)  pwd
       ;;
    3)  echo `date +%d-%B-%Y`
       ;;
    4)  awk -F: '{ print $1}' /etc/passwd
       ;;
    *)  echo "Invalid Option"
esac
```

#OUTPUT

#shubham31@shubham:~/mca$ sh oddeven.sh

#1) List all Files

#2) Print Current Directory

#3) Print Date

#4) Print Users

#1

#circlePerimiter.sh dir2 f1 file4 l1 opfile xaa

#data.dat dir3 f3 grosspay.sh LargestSmallest.sh t1

#dir done file head oddeven.sh typescript

#shubham31@shubham:~/mca$ sh oddeven.sh

#1) List all Files

#2) Print Current Directory

#3) Print Date

#4) Print Users

#2

#/home/shivangi27/mca

#shubham31@shubham:~/mca$ sh oddeven.sh

#1) List all Files

#2) Print Current Directory

#3) Print Date

#4) Print Users

#3

#09-December-2020

#shubham31@shubham:~/mca$ sh oddeven.sh

#1) List all Files

#2) Print Current Directory

#3) Print Date

#4) Print Users

#4

#root

#daemon

#bin

#sys

#sync

#games

#man

#lp

#mail

#news

#uucp

#proxy

#www-data

#backup

#list

#irc

#gnats

#nobody

#systemd-network

#systemd-resolve

#systemd-timesync

#messagebus

#syslog

#_apt

#tss

#uuidd

#tcpdump

#avahi-autoipd

#usbmux

#rtkit

#dnsmasq

#cups-pk-helper

#speech-dispatcher

#avahi

#kernoops

#saned

#nm-openvpn

#hplip

#whoopsie

#colord

#geoclue

#pulse

#gnome-initial-setup

#gdm

#shivangi27

#systemd-coredump

#vboxadd

****
****

#17. Create a menu driven calculator which asks for two integers and perform basic #arithmetic

#operations.

```
echo "Enter Number 1"
read a

echo "Enter Number 2"
read b

echo "1) Add"
echo "2) Subtract"
echo "3) Multiply"
echo "4) Divide"
echo "5) Modulo Division"
```

```
read choice


case $choice in
        1)  echo $(($a + $b));;
        2)  echo $(($a - $b));;
        3)  echo $(($a * $b));;
        4)  echo $(($a / $b));;
        5)  echo $(($a % $b));;
        *)  echo "Invalid Option"
esac


#Output
#Enter Number 1
#47
#Enter Number 2
#96
#1) Add
#2) Subtract
#3) Multiply
#4) Divide
#5) Modulo Division
#3
#4512
```

****
****

#18. Find the factorial of any number.

```
read -p "Enter a number: " num
fact=1

while [ $num -gt 1 ]
do
    fact=$((fact*num))
    num=$((num-1))
done
echo $fact
```

#Output:

#Enter a number: 8

#40320

****
****

#19. Display the fibonacci series upto some number.

```
echo "How many number of terms to be generated ?"
read n
x=0
y=1
i=2
echo "Fibonacci Series up to $n terms :"
echo "$x"
echo "$y"
while [ $i -lt $n ]
do
    i=`expr $i + 1 `
    z=`expr $x + $y `
    echo "$z"
    x=$y
    y=$z
done


#Output:


#How many number of terms to be generated ?
#3
#Fibonacci Series up to 7 terms :
#0
#1
#1
```

****
****
****

#20 Two numbers are entered through the keyboard, find the power, one number raised #to another.

```
read -p "Enter base and exponent seperated by space: " base exponent
echo "$base^$exponent" | bc
```

#Output:

#Enter base and exponent seperated by space: 3 7

#2187

****
****
****

#21. Write a script which has the functionality similar to head and tail commands.

```
#! /bin/bash
RED='\033[0;31m'
NC='\033[0m' # No Color

lines=$(wc -l $2 | awk '{print $1;}')
```

```
# Using SED ==================================

startFromLine=$(echo "$lines-$1" | bc)

echo -e "\e[1;36m"  # Changing Color

printf "\n ---------- Using sed ----------\n"

echo -en "\e[0m"  # Changing Color Back

sed -n 1,$1p $2

echo " ..."

tail -$1 $2


#Output:


 #---------- Using sed ----------

#-en

#Hello

#My name is Meet

 #...

#Have a good Day

#Nice to Meet You


********************************************************************************
********************************************************************************
******************

#22. Write a script which reports name and size of all files in a directory. whose sizes #exceed 1000.

#The filenames should be printed in       the descending order of their sizes. The total
```

#no. of files must be reported.

```
ls --sort=size -l | awk '$5 >= 1000 {print $5,$9}'
```

#Output :

#this is file name "hello"

#4096 hello

****
****
****

#24. Print the prime nos. from 1 to 300.

```
checkPrime () {
    n=$1
    if [ $n -le 1 ]
    then
        return 0
    fi

    if [ $n -le 3 ]
    then
        return 1
    fi
```

```
    if [ $(($n % 2)) -eq 0 ]
    then
        return 0
    fi

    if [ $(($n % 3)) -eq 0 ]
    then
        return 0
    fi
    i=5
    while [ $(($i*$i)) -le $n ]
    do
        if [ $(($n % $i)) -eq 0 ]
        then
            return 0
        fi
        if [ $(($n % ($i+2))) -eq 0 ]
        then
            return 0
        fi
        i=$(($i+6))
    done
    return 1
}
```

```
num=2
while [ $num -le 300 ]
do
	checkPrime $num
	isPrime=$?

	if [ $isPrime -eq 1 ]
	then
		echo "$num "
	fi

	num=$((num+1))
done

#Output :

#2
#3
#5
#7
#11
#13
#17
#19
#23
```

#29

#31

#37

#41

#43

#47

#53

#59

#61

#67

#71

#73

#79

#83

#89

#97

#101

#103

#107

#109

#113

#127

#131

#137

#139

#149

#151

#157

#163

#167

#173

#179

#181

#191

#193

#197

#199

#211

#223

#227

#229

#233

#239

#241

#251

#257

#263

#269

#271

#277

#281

#283

#293

****
****
****

#25. Program must display all the combinations of 1, 2, and 3.

```
for i in 1 2 3
do
    for j in 1 2 3
    do
        for k in 1 2 3
        do
            if [ $k -le $j ]
            then
                if [ $j -le $i ]
                then
                   echo $i $j $k
                fi
            fi
        done
    done
```

```
done

#Output:

#1 1 1
#2 1 1
#2 2 1
#2 2 2
#3 1 1
#3 2 1
#3 2 2
#3 3 1
#3 3 2
#3 3 3

**********
**********
**********

#26. Write a script for renaming each file in the directory such that it will have the current
#shell PID as an extension. The shell script should ensure that the directories do not
#get renamed.

for f in *
do
  [ -e $f ] || continue
     mv $f $f.$$
```

```
done

#Output :
#circlePerimiter.sh.3125  dir3.3125  file.3125       l1.3125              t1.3125
#data.dat.3125          done.3125  file4.3125       LargestSmallest.sh.3125  #typescript.3125
#dir2.3125             f1.3125    grosspay.sh.3125  oddeven.sh.3125        xaa.3125
#dir.3125              f3.3125    head.3125        opfile.3125

********************************************************************************
********************************************************************************
******************

#27. A file called wordfile consists of several words. Write a shell script which will receive a
#list of filenames, the first of which would be       wordfile. The shell script should report
#all occurences of each word in wordfile in the rest of the files supplied as arguments.

filesToRead=$(($#-1))
echo $filesToRead

# Reading Line by Line
while read line; do
    # Reading Word by Word
    for word in $line; do
        echo "Searching word: '$word' ..."
        # 2 is slice starting index
        # filesToRead is slice length
```

```
            grep --color=always -n $word $2
            printf "Done.\n\n"
        done
done <"$1" # $1 is the file name we want to search


#Output :
#Searching word: 'find' ...
#1:I need to find a girlfriend
#Done.


#Searching word: 'all' ...
#2:I seached all universes
#Done.


#Searching word: 'these' ...
#Done.


#Searching word: 'words' ...
#Done.
```

****

#28. Write a shell script which deletes all the lines containing the word "unix" in the files

#supplied as arguments to it.

```
for i; do
   sed -i "/\b$word\b/Id" $i
done

#$cat unix.txt
#unix
#hello
#i love unix
#linux is best

#Output :

#$ cat unix.txt
#hello
#linux is best

**********
**********
**********

#29. The word "unix" is present in only some of the files syupplied as arguments to the #shell
#script. You script should search each of these files in turn and stop at the first file
#that it encounters containing the word unix. The filename should be displayed on the #screen.

for i; do
```

```
   echo "Searching file: $i ..."

   if grep -q "unix" "$i"; then
      echo "Found in $i"
      exit
   fi
   echo "Done."
done

#Output:

# sh f1.sh file.txt
#Searching file: file.txt ...
#Found in file.txt

**********************************************************************************
**********************************************************************************
******************

#30. A shell script receives even number of filenames. Suppose four filenames are #supplied then
#the first file should get copied into      second file, the third file should get copied into
#fourth and so on.. If odd number of filenames are supplied display error message.

#Zero arguments
if [ $# -eq 0 ]; then
      echo "No Arguments"
```

```
        exit
fi
prevFile=$1
# If even no of args
if [ $(echo $# % 2 | bc ) -eq 0 ]
then
        # Looping through each Argument
        count=1
        for i; do
                if !(($count % 2)); then
                        cp $prevFile $i
                        echo "'$prevFile' copied to -> $i"
                else
                        prevFile=$i
                fi
                count=$(echo $count+1 | bc )
        done
# if odd no of args
else
        echo "Odd no of Arguments"
        exit
fi


#Output :
#//checking new1.txt which is blank
```

#shubham31@shubham:~$ cat new1.txt

#//checking new.txt

#shubham31@shubham:~$ cat new.txt

#Hello

#My name is Shubham

#Class MCA 3

#Have a good Day

#Nice to Meet You

#//running the scrpit

#shubham31@shubham:~$ sh 30.sh new.txt new1.txt

#'new.txt' copied to -> new1.txt

#//checking new1.txt again

#shubham31@shubham:~$ cat new1.txt

#Hello

#My name is Shubham

#Class MCA 3

#Have a good Day

#Nice to Meet You

****
****
****

#31. The script displays a list of all files in the current directory to which you have read,

#write and execute permissions.

```
ls -l | awk '$1 ~ /rwx/'
```

#Output:

#drwxr-x--x 2 shubham31 shubham31 4096 Nov 20 20:52 dir2.3125

#drwxrwxr-x 2 shubham31 shubham31 4096 Nov 20 20:35 dir.3125

#drwxr-x--x 2 shubham31 shubham31 4096 Nov 20 20:28 dir3.3125

****
****
****

#32. The script receives any number of filenames as arguments. It should check whether #every

#argument supplied is a file or directory. If it is a directory it should be reported.

#If it is a filename then name of the file as well as the number of lines present in it should

#be reported.

```
for i; do
   if [ -d $i ]
   then
       echo "$i -> directory"
   elif [ -f $i ]
   then
       printf "$i -> file with lines: "
```

```
        wc -l $i | awk {'print $1'}
    else
        echo "$i -> Invalid"
    fi
done


#Output:


#$ sh 32.sh hello new.txt
#hello -> directory
#new.txt -> file with lines: 5
**********************************************************************************
**********************************************************************************
******************



#33. A script will receive any number of filenames as arguments. It should check whether #such
#files already exist. If they do, then it       should be reported, if not then check if a
#subdirectory "mydir" exists or not in the current directory, if it doesnt exist then it
#should be created and in it the files supplied as arguments should be created.


if [ $# -eq 0 ]; then
        echo "No Arguments passed"
        exit
fi
```

```
for i; do
    # If file exists
    if [ -f $i ]
    then
        echo "$i exists"
    else
        # if "mkdir" exists
        if [ -d "mydir" ]
        then
            # Directory exists
            printf ""
        else
            mkdir mydir
        fi
        touch mydir/$i
        echo "$i file created in \"mydir\""
    fi
done

#Output:

#$ ls
#1     13.sh  16.sh  19.sh  21.sh  25.sh  29.sh  31.sh  34.sh  38.sh  40.sh  file1.txt  #new.txt   unix.txt
#1.sh   14.sh  17.sh  2.sh   22.sh  27.sh  3.sh   32.sh  35.sh  39.sh  5.sh   hello     #new1.txt  wordFile.txt
#11.sh  15.sh  18.sh  20.sh  24.sh  28.sh  30.sh  33.sh  37.sh  4.sh   6.sh   hello.sh   try
```

#$ sh 33.sh new.txt

#new.txt exists

#$ sh 33.sh new2.txt

#new2.txt file created in "mydir"

#$ ls

#1 13.sh 16.sh 19.sh 21.sh 25.sh 29.sh 31.sh 34.sh 38.sh 40.sh file1.txt mydir try

#1.sh 14.sh 17.sh 2.sh 22.sh 27.sh 3.sh 32.sh 35.sh 39.sh 5.sh hello #new.txt unix.txt

#11.sh 15.sh 18.sh 20.sh 24.sh 28.sh 30.sh 33.sh 37.sh 4.sh 6.sh hello.sh #new1.txt wordFile.txt

#$ cd mydir/

#~/mydir$ ls

#new2.txt

****
****
****

#34. Accept the marks of 5 subjects and calculate the percentage and grade.

read -p "Enter The marks of Subject 1 : " sub1

read -p "Enter The marks of Subject 2 : " sub2

read -p "Enter The marks of Subject 3 : " sub3

read -p "Enter The marks of Subject 4 : " sub4

```
read -p "Enter The marks of Subject 5 : " sub5
total=$((sub1+sub2+sub3+sub4+sub5))
percentage=$(echo $(echo "scale=1; 100/500" | bc)*$total | bc)
echo "Percentage = " $percentage
if [ $(echo "$percentage > 80" | bc -l) -eq 1 ]
then
    echo "Grade : Distiction"
elif [ $(echo "$percentage > 70" | bc -l) -eq 1 ]
then
    echo "Grade : First Class"
elif [ $(echo "$percentage > 60" | bc -l) -eq 1 ]
then
    echo "Grade : Second Class"
elif [ $(echo "$percentage > 50" | bc -l) -eq 1 ]
then
    echo "Grade : Third Class"
elif [ $(echo "$percentage > 35" | bc -l) -eq 1 ]
then
    echo "Grade : Pass"
else
    echo "Grade : Fail"
fi


#Enter The marks of Subject 1 : 50
```

```
#Enter The marks of Subject 2 : 51

#Enter The marks of Subject 3 : 56

#Enter The marks of Subject 4 : 54

#Enter The marks of Subject 5 : 55

#Percentage =  53.2

#Grade : Third Class


********************************************************************************
********************************************************************************
******************
```

#35. Print armstrog nos. from 1 to 500.

```
i=1
while [ $i -le 500 ]
do
    j=$i
    total=0
    while [ $j -gt 0 ]
    do
        temp=$(echo $j%10 | bc)
        sum=$(echo $temp^3 | bc)
        total=$(echo $total+$sum | bc)
        j=$(echo $j/10 | bc)
    done
    if [ $total -eq $i ]
```

```
    then
        echo "Armstrong Number : " $i
    fi
    i=$(echo $i+1 | bc)
done

#Output:
#Armstrong Number :  1
#Armstrong Number :  153
#Armstrong Number :  370
#Armstrong Number :  371
#Armstrong Number :  407
```

****

#36. Accept the measure (angles) of a triangle and displa the type of triangle. (eg. acute, #right,obtuse)

```
read -p "Enter Angle: " angle

if [[ $angle -ge 0 && $angle -lt 90 ]]; then
   echo "Acute Angle"
elif [ $angle -eq 90 ]; then
   echo "Right Angle"
elif [[ $angle -ge 91 && $angle -le 180 ]]; then
   echo "Obtuse Angle"
else
```

```
   echo "Incorrect Input"
fi


#Output:
#Enter Angle: 60
#Acute Angle


#Enter Angle: 160
#Obtuse Angle
```

****

#37. Display all the numbers from 1 to 100 which are divisible by 7.

```
echo "Numbers divisible By 7 inbetwwen 1 - 100 are :"
num=7
start=1
while [ $start -le 100 ]
do
      if [ $(echo $start%$num | bc) -eq 0 ]
      then
            echo $start
      fi

      start=$((start+1))
```

```
done


#Output:
#Numbers divisible By 7 inbetwwen 1 - 100 are :
#7
#14
#21
#28
#35
#42
#49
#56
#63
#70
#77
#84
#91
#98


********************************************************************************
********************************************************************************
******************


#38. Find the largest and smallest of 3 different numbers.


read -p "Enter First Number : " num1
```

```
read -p "Enter Second Number : " num2
read -p "Enter Third Number : " num3
largest=$num1
smallest=$num1
if [ $num2 -gt $largest ]
then
    largest=$num2
    if [ $num3 -gt $largest ]
    then
        largest=$num3
    fi
fi

if [ $num2 -lt $smallest ]
then
    smallest=$num2
    if [ $num3 -lt $smallest ]
    then
        smallest=$num3
    fi
fi
echo "Largest : " $largest "Smallest : " $smallest

#Output:
#Enter First Number : 99
```

#Enter Second Number : 101

#Enter Third Number : 98

#Largest : 101 Smallest : 98

****
****
****

#39. Find HCF and LCM of a given no.

```
read -p "Enter your Numbers : " num1 num2
if [ $num1 -le $num2 ]
then
    temp=$num1
    num1=$num2
    num2=$temp
fi
numerator=$num1
denominator=$num2
rem=1

while [ $rem -gt 0 ]
do
    rem=$(echo $numerator%$denominator | bc)
    numerator=$denominator
    denominator=$rem
done
```

```
echo "HCF : " $numerator
lcm=$(echo $num1*$num2/$numerator | bc)
echo "LCM : " $lcm

#Output:
#Enter your Numbers : 12 30
#HCF :  6
#LCM :  60

**********
**********

#40. Display the dates falling on Sundays of the current month.

startdate=$(date -d "-0 month -$(($(date +%d)-1)) days" +%d-%b-%Y-%a)
enddate=$(date -d "-$(date +%d) days +1 month" +%d-%b-%Y-%a)

d=
n=0
until [ "$d" = "$enddate" ]
do
   ((n++))
   d=$(date -d "$startdate + $n days" +%d-%b-%Y-%a)
   echo $d | grep "Sun"
```

done

#Output:

#13-Dec-2020-Sun

#20-Dec-2020-Sun

#27-Dec-2020-Sun

****
****
****

D E P A R T M E N T O F C O M P U T E R S C I E N C E
R O L L W A L A C O M P U T E R C E N T R E
G U J A R A T U N I V E R S I T Y
M.C.A. – III

R O L L NO : 31
N A M E : SHUBHAM RATHORE H.
S U B J E C T : OPRATING SYSTEM

NO.	TITLE	PAGE NO.	DATE	SIGN
1	Write a program to calculate the gross pay.	1	10/12/2020	
2	Write a program to convert this distance into metres, feet, inches and centimeters	2	10/12/2020	
3	Write a program to calculate the perimeter and area of rectangle and area and circumference of the circle.	3	10/12/2020	
4	Write a program to calculate the sum of its digits.	4	10/12/2020	
5	Write a program which would receive the logname during execution, obtain information about it from the file	5	10/12/2020	
6	Write a program will receive the f filename with its full path, the script should obtain information about this file	6	10/12/2020	
7	write a program to determine whether the seller has made profit or loss.	7	10/12/2020	
8	write a program to determine whether the seller has made profit or loss.	8	10/12/2020	
9	write a program to determine whether the seller has made profit or loss.	9	10/12/2020	
10	Check whether the entered year is a leap year or not.	10	10/12/2020	
11	The script receives two file names as arguments, the script must check whether the files are same or not,	11	10/12/2020	
12	The script receives two file Names as arguments, the script must check whether the files are same or not,	12	10/12/2020	
13	Write a shell script to display the date with the format :- 25th October 2005 is a Tuesday.	13	10/12/2020	
14	Write a shell script to display the appropriate message like : Good Morning / Good Afternoon / Good Evening	14	10/12/2020	
15	Write a shell script to display the menu driven interface current directory,date,user	15	10/12/2020	
16	Create a menu driven calculator which perform basic arithmetic operations.	16	10/12/2020	
17	Find the factorial of any number.	17	10/12/2020	
18	Display the fibonacci series upto some number.	18	10/12/2020	

D E P A R T M E N T O F C O M P U T E R S C I E N C E
R O L L W A L A C O M P U T E R C E N T R E
G U J A R A T U N I V E R S I T Y
M.C.A. – III

R O L L NO : 31
N A M E : SHUBHAM RATHORE H.
S U B J E C T : OPRATING SYSTEM

19	Two numbers are entered through the keyboard, find the power, one number raised to another.	19	10/12/2020	
20	Write a script which has the functionality similar to head and tail commands.	20	10/12/2020	
21	Write a script which reports name and size of all files in a directory. whose sizes exceed 1000.	21	10/12/2020	
22	Print the prime nos. from 1 to 300.	22	10/12/2020	
23	Program must display all the combinations of 1, 2, and 3.	23	10/12/2020	
24	Write a script for renaming each file in the directory such that it will have the current shell PID as an extension.	24	10/12/2020	
25	Write a script for report all occurences of each word in wordfile in the rest of the files supplied as arguments.	25	10/12/2020	
26	Write a shell script which deletes all the lines containing the word "unix" in the files supplied as arguments to it.	26	10/12/2020	
27	script should search each of these files in turn and stop at the first file that it encounters containing the word unix.	27	10/12/2020	
28	A shell script receives even number of filenames then the first file should get copied into second file, the third file should get copied into fourth and so on.. If odd number of filenames are supplied display error message.	28	10/12/2020	
29	The script displays a list of all files in the current directory to which you have read, write and execute permissions.	29	10/12/2020	
30	The script received file is a directory it should be reported. If it is a filename then name of the file as well as the number of lines present in it should be reported.	30	10/12/2020	
31	A script will check whether such files already exist. If they do, then it should be reported, if not then check if a subdirectory "mydir" exists or not in the current directory, if it doesnt exist then its should be created and in it the files supplied as arguments should be created.	31	10/12/2020	
32	Accept the marks of 5 subjects and calculate the percentage and grade.	32	10/12/2020	
33	Print armstrog nos. from 1 to 500.	33	10/12/2020	

D E P A R T M E N T O F C O M P U T E R S C I E N C E
R O L L W A L A C O M P U T E R C E N T R E
G U J A R A T U N I V E R S I T Y
M.C.A. – III

R O L L NO : 31
N A M E : SHUBHAM RATHORE H.
S U B J E C T : OPRATING SYSTEM

34	Accept the measure (angles) of a triangle and displa the type of triangle.	34	10/12/2020	
35	Display all the numbers from 1 to 100 which are divisible by 7.	35	10/12/2020	
36	Find the largest and smallest of 3 different numbers.	36	10/12/2020	
37	Find HCF and LCM of a given no.	37	10/12/2020	
38	Display the dates falling on Sundays of the current month.	38	10/12/2020	