Department of Computer Science

Gujarat University

Certificate

Roll No: **02** *Seat No:* _____

This is to certify that Mr./~~Ms.~~ ***Awasthi Pratik Ajaykumar*** *student of MCA Semester – III has duly completed his/~~her~~ term work for the semester ending in December 2020, in the subject of* ***Operating System(OS)*** *towards partial fulfillment of his/~~her~~ Degree of Masters in Computer Applications.*

Date of Submission
10 - December - 2020

Internal Faculty

Head of Department

Department Of Computer Science
Rollwala Computer Centre
Gujarat University

MCA – III

Subject: - Operating System (OS)

Name: - AWASTHI PRATIK AJAYKUMAR

Roll No.: - 02 **Exam Seat No.: -**

Sr. No.	Contents	Pg. No	Date	Signature
1	Glossary		10-Dec-20	
2	Numericals		10-Dec-20	

Date
Page 1

OS

Assignment

Q1 Define the following keywords

=> Base address

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

=> Batch Processing

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

=> Binary Semaphore

A semaphore that takes on only the values 0 & 1 A binary semaphore always only one process as thread taken access to a shared critical resource at a time

Block

A collection of contiguous records as a unit The units are separated by interblock gaps

Isam

A technique for organising indices in order to keep access time to a minimum it stores the data key in a balanced hierarchy that. Continually balanced hierarchy that itself as items are inserted and deleted thus all nodes always have a similar number of keys

Busy/ waiting

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

Cache memory

A memory that is smaller and faster than main memory and that is inserted between the macro

and main memory. The cache acts as a buffer for frequently used memory locations

Central Processing Unit

That portion of computer that fetches the instructions it consist of an arithmetic and logic unit (ALU) a control unit and registers often simply referred to as a processor

Clusters

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 clusters

Concurrent

Pertaining to process or threads

that take place within a common interval of time during which they have to alternate their common resources

Consumable resources

A new source that can be created and destroyed when a resource is required by a process the resource

Database

is collection of interrelated data often with controlled redundants organized according to schema to serve one or more applications

Deadlock

An event that occurs when multiple process are waiting for the availability of a resource that will not become available because it is held by the another process that is similar to just that

Demand Paging

The transfer of page from secondary memory to main memory is done at the moment of need. Compare paging

Device Driver

An operating system module that deals directly with a device or I/O module

Direct Access

The capability to obtain data from a storage device or to enter data into a storage device in a random manner.

Direct memory access

A form of I/O in which a special module called a DMA module controls the exchange of data between main memory & I/O device. It requires sends a request for the transfer of a block of data to the DMA

Disable Interupt

A condition usually created by the OS during which the processor will ignore interupt request signal of specified class

Disk allocation table

A table that indicates which blocks of secondary storage are free & available allocation to files

Distributed Operating System

A common operating system should be network of computers. The distributed operating system provides support for interprocess communication process migration mutual exclusion and the process as collection of threads

Dynamic Relocation

A process that assigns new absolute address to a computer program during execution so

that the program may be executed from a different area of main storage

Enable Interrupt

A condition usually created by the operating system during which the process will respond to interrupt request signals of specified class

External fragmentation

Occurs when memory is divided into a variable size partition corresponding to the blocks of data assigned to the memory. As jobs are moved into and out of the memory it will be occurs between the occupied partitions of the memory

File

A set of related records treated as a unit

field
- Defined logical data that are part of record
- The elementary unit of a record that may contain a data item a data aggregate or a pointer or a link

file allocation table

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

file management systems

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

file organization

- The physical order of records in a file as determined by the access method used to store and retrieve them

First in First out

e.g. queuing technique is which the next item to be retrieved is the item that has been in the queue for the longest time

first come first served

same as FIFO

Hash file

A file is which records are accessed according to the values of a key field hashing is used to locate a record on the basis of its key value

Hashing

- The selection of a storage location for an item of data by calculating the address as a function of the elements of the contents of the data this technique complicates the storage allocation function

Hit Ratio

In a two level memory the fraction of all memory accesses that are found in the faster memory

Indexed access

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

Indexed file

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

Indexed sequential access

Pertaining to the organization and accessing of the records of a storage structure through an index of the keys

Indexed 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 lookup capability. To quickly locate vicinity of a desired record

Instruction cycle

The time period during which an instruction is fetched from memory and executed when a computer is switched on

Internal fragmentation

occurs when memory is divided into a fixed size partition. If a block of data is assigned to one or more partitions this will occur if the last partition of data is smaller than the last partition

interrupt

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

Interrupt handler

A routine generally part of operating system when an interrupt occurs control is transferred also the corresponding interrupt which takes some action in the response to condition that caused the interrupt

Table

A set of compilation steps packaged to run as a unit

Kernel

A portion of operating system that includes most heavily used functions is maintained permanently in main memory

Kernel mode

A privilege mode of execution reserved for the kernel of the operating system typically kernel mode allows access to regions of main memory that are unavailable to process executing a less privileged mode and also enables execution of certain machine instruction that are restricted to the kernel mode Also referred as system mode or privileged mode

LIFO

A queuing technique in which the next item to be retrieved is the item most recently placed in queue

Deadlock

A condition in which two or more process continuously change their state in response to changes in other process without doing any useful work. This is similar to deadlock in that no progress is made.

Logical Address

A reference to a memory location independent of the current assignment of data to memory A translation must be made to a physical address before the memory access can be done

Logical Record

A record independent of it's physical environment: portion of one logical record may be located in diffent physical record or several logical records or part of logical record may be located in one physical record

Main Memory

Memory that is internal to the comp system is program address and can be loaded into registers for subsequent execution or processing

Date
Page

Malicious Software

Any software designed to cause or a target computer damage to or use up the resources of a target computer

Memory Cycle time

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

Memory Partitioning

The subdivision of storage into independent sections

Microkernel

A small privileged operating system that provide process scheduling memory management and communication services calls or other process to perform some of the func

Multiprocessing

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

Multiprogramming

- 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 diff terminology

Multiprogramming level

The number of process that are partially or fully resident in main memory

Multitasking

A mode of operation that provides for the concurrent performance or interleaved execution of two or more computer tasks. The same as multiprogramming

Mutual exclusion

A condition in which there is a set of process only one of which is able to access a given resource or perform a given function at any time

Operating System

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

Page

In virtual storage a fixed length block that has a virtual address and that is transferred to & from main memory & secondary memory

Page fault

Occurs when the page referred to is not in main memory. This causes the interrupt and requires that proper page be brought into memory

Page frame

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

Paging

The transfer of pages btw main memory & secondary memory

Physical address

The absolute location of a unit of data is memory (eg word as byte is memory block on secondary memory)

Pipe

A circular buffer allowing two process to communicate on the producer consumer model. Thus it is a first in first out queue written by one process and read by another. The pipe is generalized to allow any items in the queue to be exchanged for consumption

Preemption

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

Prepaging

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

Process

A program in execution. A process is created and scheduled by an operating system.

Process control block

The manifestation of a process in an operating system. It is a data structure containing information about the characteristics and state of the process

Process Migration

The transfer of a sufficient amount of the state of a process from one machine to another for the process to execute on the target machine

Process state

All the information that operating system needs to manage a process and that the processor needs to be properly execute the process the process state include the contents of various processor registers such as the program counter and data registers it also includes information of use to operating system such as the priority of the process or weather the process is waiting for the completion of a particular I/O must some as execution context

Program Counter

Instruction address register of program

Processor

In a computer a functional unit that interprets and executes instructions A processor consist of atleast on instruction control unit and on arithmetic unit

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 complete before proceeding

Real time System

An operating System that must Schedule and manage real time tasks

Real time tasks

A task that is executed in connection with some process as function as set of of units Related to the computer system and that must an or more deadline

Registers

high speed memory internal to the CPU some registers are user visible that is available to the programmer via the machine instruction set others registers are used by the CPU for control purpose

Relative address

An address calculated as a displacement from a base address

Remote procedure call (RPC)

A technique by which two programs on different machines interact using procedure call / return syntax & semantics Both the called & calling program behave as if the partner program were running on the same machine

Response time

In a data system the elapsed time between the end of transmission of an enquiry message and the beginning of the receipt of a response time message

Date
Page

Round Robbin

A scheduling algorithm is which process are activated in a fixed cyclic order that is all processes are in circular queue process that cannot proceed because it is waiting for an event (eg termination of child process or an input / output operation) return control to the scheduler

Scheduling

To select jobs or task that are to be dispatched in some operating systems other units of work, such as input / output operations may also be schedule

Secondary memory

Memory located outside the computer system itself that it cannot be processed directly by the processor it first must be copied into main memory example includes disks & tapes

Segment

In virtual memory a block that has virtual address the parts of a program may be unequal size and may even be of dynamically varying sizes

Segmentation

The division of program or application into segments a part of virtual memory scheme

Semaphore

An integer value used for signalling among processes only three operations may be performed on a semaphore all of which are atomic initialize decrement and increment. Depending on the exact definition of the semaphore the decrement operation may be result in the blocking of process and the increment operation may result in the unblocking of a process

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

Session

A collection of one or more processes that represents a single interactive user application or operating system function. All keyboard & mouse input is directed to the foreground session & is directed to the display screen

Shell

The portion of OS that interprets operating system interactive user commands & jobs control language commands it functions as an interface between the user and operating system

Stack

An ordered list in which items are appended to and deleted from the same end of the list known as Top. That is the next item appended to the list is put on the top end the next item to be removed from the list is put in the top and next item to be removed from the list is the item that has been in the list the shortest time. This method is characterized as last in first out

Semaphores

A condition is which a process is indefinitely delayed because the other process are always given preference

Binary Semaphore

A Semaphore is which all process waiting on the same Semaphore are queued and will eventually processed in the same order as they executed the wait

Swapping

A process that interchanges the contents of an area of main storage with the contents of an area in secondary memory

Symmetric multiprocessing

A form of multiprocessing that allows the operating system to execute on any available processor or on several available processors simultaneously

Asynchronous Operations

An operation that occurs regularly or predictably with respect to the occurrence of a specified event in another process for example: the calling of an input/output routine that forms central or required routine in the computer program

Synchronization

Situation in which two or more processes coordinate their activities

System bus

A bus used to interconnect major computer components (CPU, memory, I/O)

Tasks

Same as process

Thrashing

A phenomenon in virtual memory scheme is which the processor spends most of it's time swapping pages rather than executing instructions

Thread

A dispatchable unit of work it includes processor context (which include the program counter & stack and it's own data area for stack (to enable subroutine branching) A thread executes sequentially and is interruptible so that the processor can turn to another thread

Thread Switch

- The act of switching processor control from thread to another within the same process

Time Sharing

- The concurrent use of device by a number of users

Time Slice

- The minimum amount of time that a process can execute before being interrupted

Time Sharing

a model of an operation in which two or more processes are assigned quanta of time on the same processor

Trace

A sequence of instructions that are executed when a process is running

Trap

An unprogrammed conditional jump to a specified address that is automatically activated by the hardware: the location from which the jump was made is recorded

Trap door

Secret undocumented entry point into a program used to grant access without normal methods of access authentication

Trojan horse

Secret undocumented routine embedded within a useful program execution of the program results in execution of secret unit

User mode

The least privileged mode of execution contain regions of main memory and certain machine

Trap

An unprogrammed conditional jump to a specified address that is automatically activated by the hardware: the location from which the jump was made is recorded

Trap door

Secret undocumented entry point into a program used to grant access without normal methods of access authentication

Trojan horse

Secret undocumented routine embedded within a useful program. Execution of the program results in execution of secret unit

User mode

The least privileged mode of execution. Certain regions of memory and certain machine

instructions cannot be used in this mode

Virtual address

The address of storage location is virtual memory

Virtual memory

The storage space that may be regarded as addressable main storage by the user of computer system is virtual address. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of secondary memory available and not by the actual number of main storage locations. Scheme of the computer system and by the amount of secondary memory available and not by the actual number of main storage locations

Virus

Secret documented routine embedded within useful program. Execution of the program results in execution of the secret routine

Weak Semaphore

A semaphore in which all process waiting on the same semaphore, processed in an unspecified order (i.e the order is unknown or undetermined)

Word

An ordered set of bytes or bits that is the normal unit of transmitted or operated on within a given computer. Typically, a processor has a fixed length instruction length is equals word length

Working Set

The working set with parameter Δ for a process at virtual time t, $W(t, \Delta)$ is the set of pages of that process that have been referred in the last Δ time units

Worm

Program that can hand from computer to computer across network connections may contain a worm as backdoor

File allocation Table

A table that indicates the physical location on secondary storage of the space allocated to a file

There is an file allocation table for each file

Tefe management system

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

Encryption

The conversion of plain text or data into unintelligible form by means of a reversible mathematical computation

Multiuser security

A capability that enforces access control ensures multiple levels of classification of data

object request broker

An entity is an object created system acts as an intermediary for request sent from a client to server

File management system

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

Encryption

The conversion of plain text or data into unintelligable form by means of a reversible mathematical computation

Multilevel security

A capability that enforces access control across multiple levels of classification of data

Object request broker

An entity in a object oriented system acts as an intermediary for request sent from a client to servers

Priority inversion

A circumstance in which an operating system forces a higher priority task to wait for lower priority task

Process description

Same as Process control block (PCB)

Process image

All of the contents of a process including program, data, stack and process control block

Forms

A process that responds to request from clients via message
→ is a network a data structure that provides facilities to allow distributed
for eg a file forms a print server, mail fwd etc

Assignment-2

Banker's Algorithm

Process	Allocation			Max			Available			Need		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2	7	4	3
P_1	2	0	0	3	2	2				1	2	2
P_2	3	0	2	9	0	2				6	0	0
P_3	2	1	1	2	2	2				0	1	1
P_4	0	0	2	4	3	3				4	3	1

Need ≤ work → work = work + Allocation

P_0 743 ≤ 332 ← × condition fails

P_1 122 < 332 Condition true

w = work allocation

w = 332 + 200

= 532

P_2 need ≤ work

600 ≤ 532 condition fails

P_3 = need ≤ work

011 ≤ 532

Condition true

w = w + allocation

= 532 + 211

= 743

need ≤ work → work = work + allocated

P_4 : 431 ≤ 743 → = 743 + 002 = 745

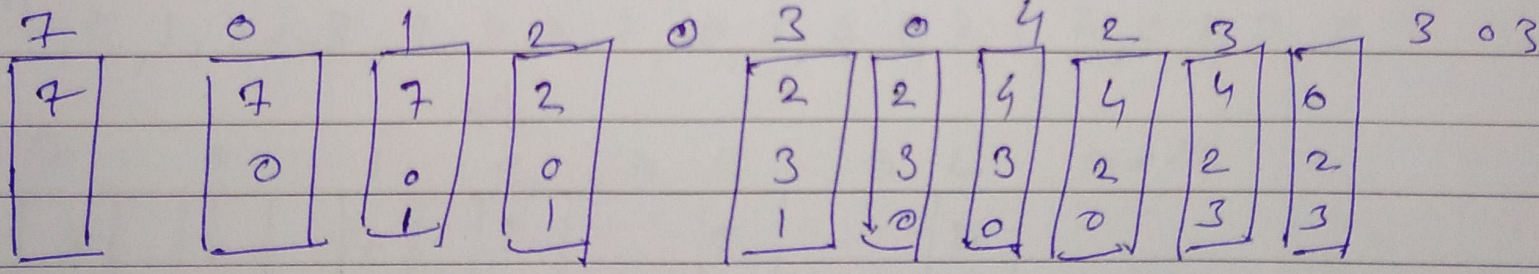
P_3 : need ≤ work → work = work + allocated
743 ≤ 745 = 745 + 010 = 755

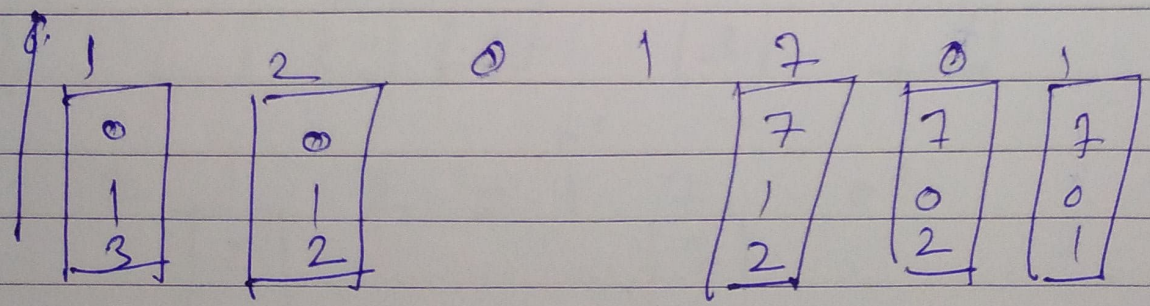
P_2 : need ≤ work → work = work + allocated
600 ≤ 755 = 755 + 342 = 1057

Safe sequence is → <P_1, P_3, P_4, P_0, P_2>

FIFO

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

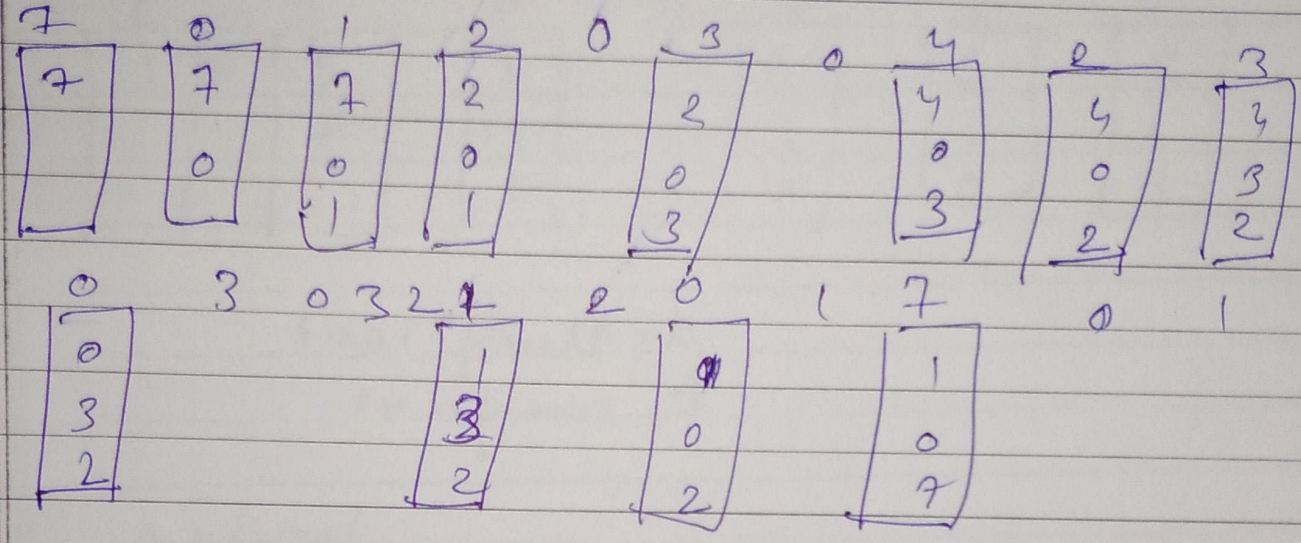




Page fault = 15 No of frames = 3

LRU

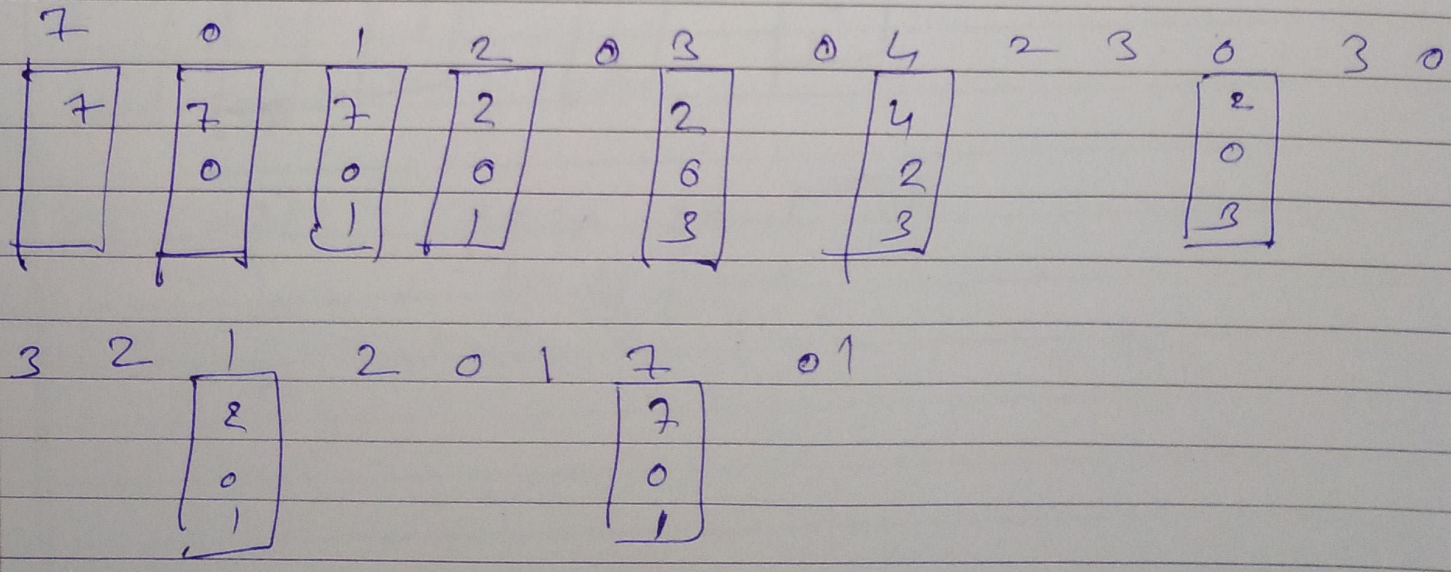
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



no of frames = 3 page fault = 12

optimal

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

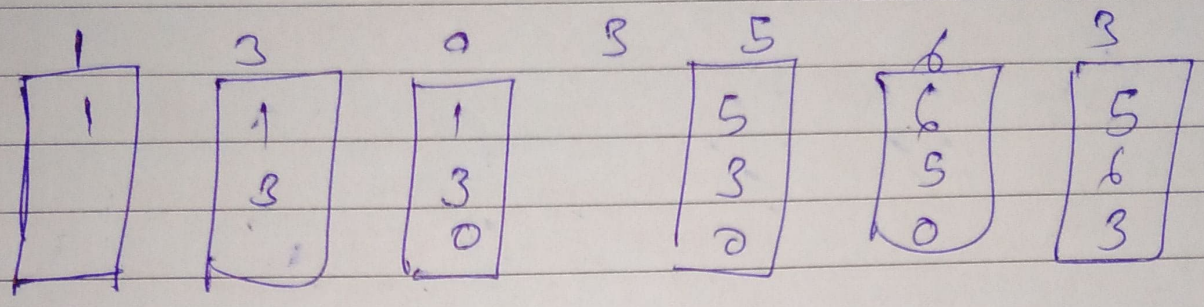


no of frames = 3

page fault = 9

FIFO

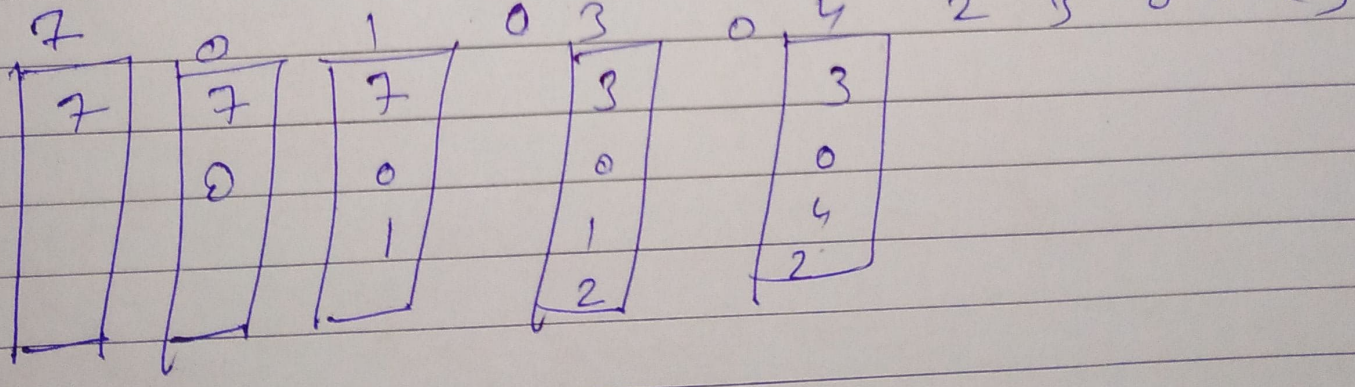
1, 3, 0, 3, 5, 6, 3



Page fault = 6
No of frms = 3

optimal

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



page fault = 6
No of frms = 4

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 : 02
N A M E : AWASTHI PRATIK AJAYKUMAR
S U B J E C T : Operating System (OS)

NO.	TITLE	PAGE NO.	DATE	SIGN
1	Calculate Gross Salar		10-Dec-20	
2	Distance between two cities		10-Dec-20	
3	Area of Rectancle & Circle		10-Dec-20	
4	Sum of digits of a number		10-Dec-20	
5	Lognames		10-Dec-20	
6	File details		10-Dec-20	
7	Profit/ Loss		10-Dec-20	
8	Odd/ Even		10-Dec-20	
9	Prime or No		10-Dec-20	
10	Leap Year		10-Dec-20	
11	Similar Files		10-Dec-20	
12	---Cancelled ---		10-Dec-20	
13	--Cancelled ---		10-Dec-20	
14	Date Display		10-Dec-20	
15	Greeting		10-Dec-20	
16	Menu Driven Interface		10-Dec-20	
17	Arithmetic Calculator		10-Dec-20	
18	Factorial		10-Dec-20	
19	Fibonacci		10-Dec-20	
20	Power of yraised to x		10-Dec-20	
21	Similar to Head/Tail Command		10-Dec-20	
22	Files > 1000		10-Dec-20	
23	---Cancelled ---		10-Dec-20	
24	Prime numbers 1-2=300		10-Dec-20	
25	Combinations of 1, 2, 3		10-Dec-20	
26	Rename each file with extension .PID		10-Dec-20	

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 : 02
N A M E : AWASTHI PRATIK AJAYKUMAR
S U B J E C T : Operating System (OS)

27	**Occurrenceof each word in file**		**10-Dec-20**	
28	**Delete lines with word “unix”**		**10-Dec-20**	
29	**Stop at the first file that encounter word “unix”**		**10-Dec-20**	
30	**Copy even files**		**10-Dec-20**	
31	**All files in current directory with read, write & execute permissions**		**10-Dec-20**	
32	**File or Directory?**		**10-Dec-20**	
33	**File exists or not? If not create in mydi**		**10-Dec-20**	
34	**Calculate Percentage & Grades**		**10-Dec-20**	
35	**Armstrong numbers between 1 to 500**		**10-Dec-20**	
36	**Acute / Right / Obtuse Angle**		**10-Dec-20**	
37	**Numbers divisible by 7 in 1-100**		**10-Dec-20**	
38	**Smallest & Largest of 3 numbers**		**10-Dec-20**	
39	**HCF & LCM**		**10-Dec-20**	
40	**Dates falling on Sunday of current month**		**10-Dec-20**	

****

NAME : PRATIK AWASTHI

ROLLNUM : 02

CLASS : MCA3

SUBJECT : OPERATING SYSTEM

****

#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
dearness=$(echo $salary*40/100 | bc)
houserent=$(echo $salary*20/100 | bc)
gross=$((salary-houserent+dearness))
echo "Dearness Allowance : "$dearness
echo "House Rent : "$houserent
echo "Gross Salaray : "$gross
```

#Output :
#Enter Your Salary : 20000
#Dearness Allowance : 8000
#House Rent : 4000
#Gross Salaray : 24000

#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))

#Output :
#Enter Distance between 2 cities in kms : 50
#Meters : 50000
#Feet :   164042.00
#Inches :  1968504.00
#centimeters :   5000000

#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)

#Output:
#Enter Length And Breadth of Rectangle : 10 20
#Enter Radius of circle : 1
#Perimeter of the Rectangle is :  30
#Area of Rectangle is :  200
#Circumference of Circle :  6.28
#Area of circle :  3.14

#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 : 1234
#Total :  10



#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 :



cut -d ":" -f 1,3,4,5,6,7 /etc/passwd --output-delimiter=' | '

#Output :

#[ec2-user@ip-172-31-85-227 one]$
#root | 0 | 0 | root | /root | /bin/bash
#bin | 1 | 1 | bin | /bin | /sbin/nologin
#daemon | 2 | 2 | daemon | /sbin | /sbin/nologin
```

#adm | 3 | 4 | adm | /var/adm | /sbin/nologin

#lp | 4 | 7 | lp | /var/spool/lpd | /sbin/nologin

#sync | 5 | 0 | sync | /sbin | /bin/sync

#shutdown | 6 | 0 | shutdown | /sbin | /sbin/shutdown

#halt | 7 | 0 | halt | /sbin | /sbin/halt

#mail | 8 | 12 | mail | /var/spool/mail | /sbin/nologin

#operator | 11 | 0 | operator | /root | /sbin/nologin

#games | 12 | 100 | games | /usr/games | /sbin/nologin

#ftp | 14 | 50 | FTP User | /var/ftp | /sbin/nologin

#nobody | 99 | 99 | Nobody | / | /sbin/nologin

#systemd-network | 192 | 192 | systemd Network Management | / | /sbin/nologin

#dbus | 81 | 81 | System message bus | / | /sbin/nologin

#rpc | 32 | 32 | Rpcbind Daemon | /var/lib/rpcbind | /sbin/nologin

#libstoragemgmt | 999 | 997 | daemon account for libstoragemgmt | /var/run/lsm | /sbin/nologin

#sshd | 74 | 74 | Privilege-separated SSH | /var/empty/sshd | /sbin/nologin

#rpcuser | 29 | 29 | RPC Service User | /var/lib/nfs | /sbin/nologin

#nfsnobody | 65534 | 65534 | Anonymous NFS User | /var/lib/nfs | /sbin/nologin

#ec2-instance-connect | 998 | 996 | | /home/ec2-instance-connect | /sbin/nologin

#postfix | 89 | 89 | | /var/spool/postfix | /sbin/nologin

#chrony | 997 | 995 | | /var/lib/chrony | /sbin/nologin

#rngd | 996 | 994 | Random Number Generator Daemon | /var/lib/rngd | /sbin/nologin

#tcpdump | 72 | 72 | | / | /sbin/nologin

#ec2-user | 1000 | 1000 | EC2 Default User | /home/ec2-user | /bin/bash

#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 :

[ec2-user@ip-172-31-85-227 one]$

```
ll|awk '{print $1 "|" $3 "|" $4 "|"  $5 "|" $6,$7 "|" $8}'

# total|||| |
# -rw-rw-r--|ec2-user|ec2-user|0|Dec 8|18:05
# -rw-rw-r--|ec2-user|ec2-user|0|Dec 9|05:26
# -rw-rw-r--|ec2-user|ec2-user|84|Dec 8|18:03
# -rw-rw-r--|ec2-user|ec2-user|50|Dec 8|17:45
# -rw-rw-r--|ec2-user|ec2-user|42|Dec 8|17:46
# -rw-rw-r--|ec2-user|ec2-user|340|Dec 9|05:34


# 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 Swlling 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 : 90
# Enter The Swlling price for Product : 100
# Profit :  10
```

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 : 160
EVEN

Enter The Number : 295
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 : 20
# Not Prime

# Enter the Number : 11
# 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 : 2020
# Its  A 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 :

[ec2-user@ip-172-31-85-227 one]$ ./11.sh students.dat sports.dat

Files are not same

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 :

8 December 2020 is a tuesday

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 Morning!!!!

# 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:

[ec2-user@ip-172-31-85-227 one]$ sh 16.sh

1) List all Files

2) Print Current Directory

3) Print Date

4) Print Users

1

1

16.sh cut ls script.sh seven.sh sports.dat students.dat xyz.txt

[ec2-user@ip-172-31-85-227 one]$ sh 16.sh

1) List all Files

2) Print Current Directory

3) Print Date

4) Print Users

2

/home/ec2-user/one

[ec2-user@ip-172-31-85-227 one]$sh 16.sh

1) List all Files

2) Print Current Directory

3) Print Date

4) Print Users

3

09-December-2020

[ec2-user@ip-172-31-85-227 one]$ sh 16.sh

1) List all Files

2) Print Current Directory

3) Print Date

4) Print Users

4

root

bin

daemon

adm

lp

sync

shutdown

halt

mail

operator

games

ftp

nobody

systemd-network

dbus

rpc

libstoragemgmt

sshd

rpcuser

nfsnobody

ec2-instance-connect

postfix

chrony

rngd

tcpdump

```
# ec2-user

# 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
```

```
# 10
# Enter Number 2
# 10
# 1) Add
# 2) Subtract
# 3) Multiply
# 4) Divide
# 5) Modulo Division
# 1
# 20

# 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: 3
# 6

# 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 ?
# 5
# Fibonacci Series up to 5 terms :
# 0
# 1
# 1
# 2
# 3
```

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: 2 3
# 8

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


 #### THIS NEED TO NE DISCUSSED 21.


# 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 -l|tail -n+2 | sort -nk5 | awk '$5 >= 300 {print $5,$9}'


# Output :


# 340 linux.txt
# 342 25.sh

# 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
   [ -d $f ] || continue
                                    ext=`echo $$`
```

```
    mv $f $f.$ext
done

# Output :

# -rwxrwxrwx 1 ec2-user ec2-user 342 Dec  9 05:54 16.sh.3056
# -rwxrwxrwx 1 ec2-user ec2-user  96 Dec  9 06:18 26.sh.3056
# -rw-rw-r-- 1 ec2-user ec2-user   0 Dec  8 18:05 cut.3056
# -rw-rw-r-- 1 ec2-user ec2-user   0 Dec  9 05:26 ls.3056
# -rw-rw-r-- 1 ec2-user ec2-user  84 Dec  8 18:03 script.sh.3056
# -rwxrwxrwx 1 ec2-user ec2-user 295 Dec  9 05:52 seven.sh.3056
# -rw-rw-r-- 1 ec2-user ec2-user  50 Dec  8 17:45 sports.dat.3056
# -rw-rw-r-- 1 ec2-user ec2-user  42 Dec  8 17:46 students.dat.3056
# -rw-rw-r-- 1 ec2-user ec2-user 340 Dec  9 05:34 xyz.txt.3056

# 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

# [ec2-user@ip-172-31-85-227 one]$ ./27.sh dummy.txt 27.sh
# dummy.txt
# Searching word: 'Hi' ...
# Done.

# Searching word: 'to' ...
# Done.



# 28. Write a shell script which deletes all the lines containing the word "unix" in the files
# supplied as arguments to it.

word="pratik"
for i in *; do
   sed -i "/\b$word\b/Id" $i
done

# dummy.txt:my name is pratik

# 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 in *; do
   echo "Searching file: $i ..."

   if grep -q "JOKER" "$i"; then
      echo "Found in $i"
```

```
        break;
    fi
    echo "Done."
done

# Output:

# Searching file: 20.sh..3056 ...
# Done.
# Searching file: 25.sh.3056 ...
# Done.

# 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 :

# [ec2-user@ip-172-31-85-227 one]$ ./30.sh
# No Arguments



# [ec2-user@ip-172-31-85-227 one]$ ./30.sh 1.txt 2.txt
# '1.txt' copied to -> 2.txt

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

for line in *
do
if [ -r $line -a -w $line -a -x $line ]
then
echo "$line has all permissions"
else
echo "files not having all permissions"
fi
```

```
done

# Output:

# 16.sh..3056 has all permissions
# files not having all permissions
# 26.sh.3056 has all permissions
# 27.sh has all permissions
# files not having all permissions
# 30.sh has all permissions
# files not having all permissions
# files not having all permissions
# files not having all permissions
# files not having all permissions
# seven.sh..3056 has all permissions
# files not having all permissions
# files not having all permissions
# files not having all permissions
# files not having all permissions

# 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 in $*; 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:

# [ec2-user@ip-172-31-85-227 one]$ ./32.sh 30.sh students.dat..3056
# 30.sh -> file with lines: 26
# students.dat..3056 -> file with lines: 4

# 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 in $*; 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:



# [ec2-user@ip-172-31-85-227 one]$ ./33.sh xyz.txt qwe.txt sdf.txt
# xyz.txt file created in "mydir"
# qwe.txt file created in "mydir"
# sdf.txt file created in "mydir"
# [ec2-user@ip-172-31-85-227 one]$ ls mydir/
# qwe.txt  sdf.txt  xyz.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

# Output:

# Enter The marks of Subject 1 : 50
# Enter The marks of Subject 2 : 50
# Enter The marks of Subject 3 : 50
# Enter The marks of Subject 4 : 50
# Enter The marks of Subject 5 : 50
# Percentage =  50
# Grade : Third Class
```

```
# Enter The marks of Subject 1 : 90
# Enter The marks of Subject 2 : 90
# Enter The marks of Subject 3 : 90
# Enter The marks of Subject 4 : 90
# Enter The marks of Subject 5 : 90
# Percentage =  90
# Grade : Distiction

# 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 display 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: 90
# Right Angle

# Enter Angle: 25
# Acute 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 : 15
# Enter Second Number : 14
# Enter Third Number : 13
# Largest :  15 Smallest :  13

# 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 : 24 36
# HCF :  12
# LCM :  72

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

startdate=$(date -d "`date +%Y%m01`" +%d-%b-%Y-%a)
```

```
enddate=$(date -d "`date +%Y%m01` +1 month -1 day" +%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:

# 06-Dec-2020-Sun
# 13-Dec-2020-Sun
# 20-Dec-2020-Sun
# 27-Dec-2020-Sun
```