Department of Computer Science

Gujarat University

Certificate

Roll No: 01 Seat No: _____

This is to certify that Mr./~~Ms.~~ AJMERI JUNED JAKIRBHAI student of MCA Semester – III has duly completed his/~~her~~ term work for the semester ending in December 2020, in the subject of OPERATING SYSTEMS towards partial fulfillment of his/~~her~~ Degree of Masters in Computer Applications.

10TH DEC, 2020
Date of Submission

Internal Faculty

Head of Department

Department Of Computer Science
Rollwala Computer Centre
Gujarat University

MCA – 3

Subject: - OPERATING SYSTEMS

Name: - AJMERI JUNED JAKIRBHAI

Roll No.: - 01 **Exam Seat No.: -**_____

Sr. No.	Contents	Pg. No	Date	Signature
01	Assignment - 1 Definition / Glossary (Introduction to Operating System, Process Management, Process Synchronization, Memory Management, Storage Management, File System)	01	10/12/2020	
02	Assignment - 2 Numericals (Banker's Algorithm, FIFO Algorithm, LRU Algorithm, OPTIMAL Algorithm)	34	10/12/2020	

DEPARTMENT OF COMPUTER SCIENCE
ROLLWALA COMPUTER CENTRE
GUJARAT UNIVERSITY
M.C.A. – 3

ROLL NO : 01
NAME : AJMERI JUNED JAKIRBHAI
SUBJECT : OPERATING SYSTEMS

NO.	TITLE	PAGE NO.	DATE	SIGN
14	Write a shell script to display the date with the format :- 25th October 2005 is a Tuesday.		10/12/2020	
15	Write a shell script to display the appropriate message like : Good Morning / Good Afternoon / Good Evening		10/12/2020	
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".		10/12/2020	
17	Create a menu driven calculator which asks for two integers and perform basic arithmetic operations.		10/12/2020	
18	Find the factorial of any number.		10/12/2020	
19	Display the fibonacci series upto some number		10/12/2020	
20	Two numbers are entered through the keyboard, find the power, one number raised to another.		10/12/2020	
21	Write a script which has the functionality similar to head and tail commands.		10/12/2020	
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		10/12/2020	
23	A friend of yours has promised to log in a particular time. You want to contact him as soon as he logs in, write a script which checks after every minute whether the friend has logged in or not. The logname should be supplied at command prompt.		10/12/2020	
24	Print the prime nos. from 1 to 300.		10/12/2020	
25	Program must display all the combinations of 1, 2, and 3.		10/12/2020	
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.		10/12/2020	
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.		10/12/2020	

DEPARTMENT OF COMPUTER SCIENCE
ROLLWALA COMPUTER CENTRE
GUJARAT UNIVERSITY
M.C.A. – 3

ROLL NO : 01
NAME : AJMERI JUNED JAKIRBHAI
SUBJECT : OPERATING SYSTEMS

NO.	TITLE	PAGE NO.	DATE	SIGN
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.**		**10/12/2020**	
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.**		**10/12/2020**	
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.**		**10/12/2020**	
4	**If a five digit number is entered through the keyboard, calculate the sum of its digits.**		**10/12/2020**	
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.**		**10/12/2020**	
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.**		**10/12/2020**	
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.**		**10/12/2020**	
8	**Check whether the entered no. is odd or even.**		**10/12/2020**	
9	**Check whether the entered no. is prime or not.**		**10/12/2020**	
10	**Check whether the entered year is a leap year or not.**		**10/12/2020**	
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**		**10/12/2020**	
12	**Write a script which will display whether your friend has logged in or not, if he has logged in then send him some message.**		**10/12/2020**	
13	**While executing a shell script, either the logname or uid is supplied at the command prompt, write a shell script to find out at how many terminals has this user logged in.**		**10/12/2020**	

DEPARTMENT OF COMPUTER SCIENCE
ROLLWALA COMPUTER CENTRE
GUJARAT UNIVERSITY
M.C.A. – 3

ROLL NO : 01
NAME : AJMERI JUNED JAKIRBHAI
SUBJECT : OPERATING SYSTEMS

NO.	TITLE	PAGE NO.	DATE	SIGN
28	Write a shell script which deletes all the lines containing the word "unix" in the files supplied as arguments to it.		10/12/2020	
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.		10/12/2020	
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.		10/12/2020	
31	The script displays a list of all files in the current directory to which you have read, write and execute permissions.		10/12/2020	
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.		10/12/2020	
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.		10/12/2020	
34	Accept the marks of 5 subjects and calculate the percentage and grade. 35. Print armstrog nos. from 1 to 500.		10/12/2020	
35	Print armstrog nos. from 1 to 500.		10/12/2020	
36	Accept the measure (angles) of a triangle and display the type of triangle. (eg. acute, right, obtuse)		10/12/2020	
37	Display all the numbers from 1 to 100 which are divisible by 7		10/12/2020	
38	Find the largest and smallest of 3 different numbers.		10/12/2020	
39	Find HCF and LCM of a given no.		10/12/2020	
40	Display the dates falling on Sundays of the current month.		10/12/2020	

* Assignment - 1 *

* Definitions :-

* Base Address :-

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

* Batch processing :-

Pertain 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 value 0 and 1. A binary semaphore allows only one process or thread to have access to shared critical resource at a time.

* Block :-

(1) A collection of contiguous records that are recorded as a unit the unit are separated by interblock gaps.

(2) A group of bits that are transmitted as a unit.

* B-tree :-

A technique for organizing indexes. in order to keep access time to a minimum it stores the data keys in a balanced hierarchy that continally realigns it self as items are insrted and deleted. This, all nods always have a similar number of keys.

* Busy waiting :-

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

* Cache Memory :-

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

* CPU (Central Processing Unit) :-

That portion of a computer that fetches and executes instructions. It consists of an Arithmetic and logic Unit (ALU), a control unit and registers often simply referred to as a processor.

* Cluster :-

A group of interconnected, whole computer 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.

* Concurrent :-

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

* Consumable resource :-

A resource that can be created and destroyed. when a resorce is aquired by a process the resorce ceased to exist.

Example :- interrupts, signals, messages, etc.

* Database :-

A collection of interrelated data often with controlled redundancy, organized according to schema to serve one or more applications, the data are stored so that they can be used by different programs without concern for the data structure or organization.

* **Deadlock :-**

(1) An impasse 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. (2) An impasse that occurs when multiple processes are waiting for an action by or a response from another process that is in a similar wait state.

* **Deadlock avoidance :-**

A Dynamic technique that examines each new resource request for deadlock. if the new request could lead to a deadlock then the request is denied.

* **Demand Paging :-**

The transfer of page from secondary memory to main memory storage at the moment of need. Compare pre-paging.

* Device Drivers :-
An operating system module (usually in the kernal) that deals directly with a device or I/O module.

* DMA (Direct Memory Access) :-
A form of I/O in which a special module, called a DMA module, control the exchange of data between main memory and I/O devices. The process sent a request of for the transfer of a block of data to the DMA module and is interrupted only after the entire block has been transferred.

* Disable interrupt :-
A condition usually created by the operating system, during which the processor will ignore interrupt request signal of a specified class.

* Disk allocation table :-

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

* dispatch :-

To allocate time on a processor to jobs or tasks that are ready for execution.

* 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 prevension or detection of deadlock.

* Dynamic Relocation :-

A process that assigns new absolute addresses 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 processor will respond to interrupt request signals of a specified class.

* External fragmentation :-

occurs when memory divided into variable-size partitions corresponding to the blocks of data assigned to the memory. As segments are moved into and out of the memory, gaps will occurs between the occupied portions of memory.

* File :-

A set of related records treated

as a unit.

* Field :-
= (1) Defined logical data that are part of record. (2) The elementary unit of a record that may contain a data item, a data aggregate, a pointer or a link.

* FAT (File allocation Table) :-
= = = = A table that indicates the physical location on secondary storage of the space allocationed to a file. There is one file allocation table for each file.

* File management system :-
= = = = A set of system software that provides services to users and applications in the use of files, including file 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.

* FCFS (First come first served) :-
Same as FIFO.

* FIFO (First in First out) :-
A queueing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

* 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 its key value.

* Hashing :-
= The selection of a storage location for an item of data by calculating the address as a function of the contents of the data. This technique complicates the storage allocation function but results in rapid random variable.

* Hit Ratio :-
= = in a two level memory, the fraction of all memory access that are found in the faster memory.

* Index access :-
= = Pertaining to the organization and accessing of the records of a storage structure through a separated index to the locations of the stored records.

* Index file :-
= = A file in which records are accessed according to the

value of key fields. An index is required that indicates the location of each record on the basis of each key value.

* index sequential access :-
Pertaining to the organization and accessing of a records of a storage structure through an index of the keys that are stored in arbitrarily partitioned sequenced files.

* index sequence File :-
A File in which records are ordered according to the value 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 look-up capability to quickly reach the vicinity of desired record.

* instruction cycle :-
The time period during which one instruction is fetched from memory and executed when a computer is given an instruction in machine language.

* internal fragementation :-
Occurs when memory is divided into fixed-size partitions, if a block of data is assigned to one or more partitions, then there may be wasted space in the last partition. Then this will occurs if the last portion of data is smaller than the last-partition.

* 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.

* interrupt handler :-
= A routine generally part of the operating system. When an interrupt occurs, control is transferred to the corresponding interrupt handler, which take some action in response to condition that caused by interrupt.

* Job :-
= A set of computation steps packaged to run as a unit.

* Kernel :-
= A portion of the OS that includes the most heavily used portion of software. Generally, the kernel is maintained permanently in main memory. The kernel runs in privileged mode and responds to calls from processes and interrupts from devices.

* **Kernel mode :-**
A privileged mode of execution reserved for the kernel mode allows access to regions of main memory that are unavailable to processes executing in a less-priviledged mode and also enables execution of certain machine instructions that are restricted to the kernel mode. Also referred to as system mode or previledged mode.

* **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.

* **livelock :-**
A condition in which two or more processes continously change their state in response to changes in other processes without doing any useful work.

* logical address :-
A reference to a memory location independent of the current assignment of data to memory.

* logical record :-
A record independent of its physical environment, portions 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.

* Main Memory :-
Memory that is internal to the computer system in, is program addressable and can be loaded into registers for subsequent execution or processing.

* Malicous software :-
Any software designed to cause damage to or use up the resources of a target computer. Malicous software is frequently concealed within or masquerades as legitimate software.

* Memory Cycle Time :-
The time it takes to read one word from or write one word to memory. This is the inverse of the rate at which words can be read from or written to memory.

* Memory Partitioning :-
The subdividing of storage into independent sections.

* Micro kernel :-
A small privileged OS core that provides process scheduling,

memory management, and communication services and relies on other processing to perform some of the functions traditionally associated with the OS kernel.

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

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

* Multiprogramming level :- The number of processes that are partially or fully resident in main memory.

* **Multitasking :-**
A mode of operation that provides for concurrent performance or interleaved execution of two or more computer tasks.

* **Mutual Exclusion :-**
A condition in which there is set of processes 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, and data management.

* Page :-
= In virtual storage, a fixed length block that has a virtual address and that is transferred as a unit between main memory and secondary memory.

* 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.

* Page frame :-
= A fixed-size contiguous block of main memory used to hold a page.

* Paging :-
= The transfer of pages between main memory and secondary memory.

* Physical address :-
The absolute location of a byte unit of data in memory. Ex.:- word or byte in main memory, etc.

* Pipe :-
A circular buffer allowing two processes to communicate on the producer - consumer model. Thus, it is a first in first out queue, written by one process and read by another. In some systems, the pipe is generalized to allow any item in the queue to be selected 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 pages will be needed in the near future, conserving disk I/O Compare demand paging.

* Process :-
A program in execution. A process is controlled and scheduled by the operating system. Same as task.

* Process control block :-
The manifestation of a process in an Os. it is a data structure containing information about the characteristics and state of the process.

* process state :-
All of the information that the os needs to manage a

process and that the process. The process state includes the contents of the various processor registers, such as the program counter and data registers, it also includes information of use to the OS, such as the priority of the process and whether the process is waiting for completion of a particular i/o event. same as execution context.

* **processor :-** in a computer, a functional unit that interprets and executes instructions. A processor consists of at least an instruction control unit and an arithmetic unit.

* **program counter :-** instruction address register.

* **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 task :-** A task that is executed in connection with some process or function with some process or set of events external to the computer system and that must meet one or more deadlines to interact effectively and correctly with the external environment.

* **Registers :-** High speed memory internal to the CPU. some registers are

user visible that is, available to the programmer via the machine instruction set. Other registers are used only by the CPU, for control purposes.

* Relative Address :-
= = An address calculated as a displacement from a base address.

* 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 message, measured at the enquiry terminal.

* Round Robin :-
= = A scheduling algorithm in which processes are activated in a fixed cyclic order, that is, all processes are in a circular queue. A process that can not processed

because it is waiting for some event returns control to the scheduler.

* Scheduling :-
Scheduling To select jobs or tasks that are to be dispatched. In some OS, other units of work, such. i/o operations, may also be scheduled.

* Secondary Memory :-
Memory located out side the computer system itself, that is, it cannot be processed directly by processor. it must first be copied into main memory.

* Segment :-
in virtual memory, a block that has a virtual address. The blocks of a program may be of unequal length and may even be of dynamically varying lengths.

* Segmentation :-

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

* Semaphoze :-

An integer value used for signaling among processes. Only three operations may be performed on a semaphoze, all of which are atomic, initialize, decrement and increment. Depending on the exact definition of the semaphoze, the decrement operation may result in the blocking of a process, and the increment operation may result in the unblocking of a process. Also known as a counting semaphoze or a general semaphoze.

* Sequential File :-

A file in which records are ordered according to the values of one or more key field and processed in the same sequence from the beginning of the file.

* **Shell :-**

The portion of the OS that interprets interactive user commands and job control language commands. it functions as an interface between the user and OS.

* **stack :-**

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

* **starvation :-**

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

* Strong semaphore :-
A semaphore in which all processes waiting on the same semaphore are queued and will eventually proceed in the same order as they executed the wait (P) operations (FIFO order).

* Swapping :-
A process that interchanges the contents of an area of main storage with the contents of an area in secondary memory.

* SMP (symmetric multiprocessing) :-
A form of multiprocessing that allows the OS to execute on any available processor or on several available processors simultaneously.

* Synchronization :-
Situation in which two or more processes coordinate their activities based on a condition.

* Synchronous operation :-
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 i/o routine that received control at a recorded location in a computer program.

* System bus :-
A bus used to interconnect major computer components. (CPU, memory, i/o)

* thread :-
A dispatchable unit of work. it includes a processor context and its own data area for a stack. A thread executes sequentially and is interruptible so that the processor can turn to another thread. A process may consist of multiple threads.

* thread switch :-
= = The act of switching processor control from one thread to another with in the same process.

* Time sharing :-
= = The concurrent use of a device by a number of users.

* time slice :-
= = The maximum amount of time that a process can executed before being interrupted.

* trap :-
= An unprogrammed conditional jump to a specified address that is automatically activated by hardware, the location from which the jump was made is recorded.

* trojan horse :-
= = Secret undocumented routine embedded within a useful program. Execution of the program results in execution of the secret routine.

* User mode :-
  = = The least privileged mode of execution. Certain regions of main memory and certain machine instructions cannot be used in this mode.

* Virtual address :-
  = = The address of a storage location in virtual memory.

* Virtual memory :-
  = = The storage space that may be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. 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.

* Virus :-
= Secret undocumented routine embedded within a useful program. Execution of the program results in execution of the secret routine.

* Weak semaphore :-
= = = A semaphore in which all processes waiting on the same semaphore proceed in an unspecified order.

* Word :-
= An ordered set of bytes or bits that is the normal unit in which information may be stored, transmitted or operated on within a given computer.
instruction length = word length.

* worm :-
= Program that can travel from computer to computer across network connections. May contain a virus or bacteria.

—°*°—

* Assignment - 2 *

* Banker's Algorithm :-

(1)

Process	Allocation			Max.			(Work) Available			(Max.-Allocation) Need		
	A	B	C	A	B	C	A	B	C			
P_0	0	1	0	7	5	3	3	3	2			
P_1	2	0	0	3	2	2						
P_2	3	0	2	9	0	2						
P_3	2	1	1	2	2	2						
P_4	0	0	2	4	3	3						

⇒

Process	Allocation			Max.			(Work) Available			(Max.-Allocation) 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	3
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

$P_0 \rightarrow 7\ 5\ 3 \leq 3\ 3\ 2$ ← (X)

condition fails.

Need ≤ work

$P_1 \rightarrow 122 \leq 332 \rightarrow$ (T) condition true.

New work = work + alloc.

= 332 + 200

= 532

Need ≤ work

$P_2 \rightarrow 600 \leq 532 \rightarrow$ (F) condition fails.

Need ≤ work

$P_3 \rightarrow 011 \leq 532 \rightarrow$ (T) condition true

New work = work + alloc.

= 532 + 211

= 743

Need ≤ work

$P_4 \rightarrow 431 \leq 743 \rightarrow$ (T) condition true.

New work = work + alloc.

= 743 + 002

= 745

Need ≤ work

$P_0 \rightarrow 743 \leq 745 \rightarrow$ (T) condition true.

New work = work + alloc.

= 745 + 010

= 755

Need ≤ work

$P_2 \rightarrow 600 \leq 755 \rightarrow$ (T) condition true.

New work = work + alloc.

= 755 + 342

= 1057

* Safe sequence is =>
$< P_1, P_3, P_4, P_0, P_2 >$.

(2)
=

* FIFO :-
=

=> 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	4	4	4	0
	0	0	0	0	3	3	3	2	2	2
		1	1	1	1	0	0	0	3	3
✓	✓	✓	✓	X	✓	✓	✓	✓	✓	✓

3	0	3	2	1	2	0	1	7	0	1
0	0	0	0	0	0	0	0	7	7	7
2	2	2	2	1	1	1	1	1	0	0
3	3	3	3	3	2	2	2	2	2	1
X	X	X	X	✓	✓	X	X	✓	✓	✓

=> No. of Page fault = 15
=> No. of Page Hit = 7
=> 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	2	2	4	4	4	0
	0	0	0	0	0	0	0	0	3	3
		1	1	1	3	3	3	2	2	2
✓	✓	✓	✓	X	✓	X	✓	✓	*✓	*✓

3	0	3	2	1	2	0	1	7	0	1
0	0	0	0	1	1	1	1	1	1	1
3	3	3	3	3	3	0	0	0	0	0
2	2	2	2	2	2	2	2	7	7	7
X	X	X	X	✓	X	✓	X	✓	X	X

=> No. of page fault = 12
=> No. of page Hit = 10
=> No. of frames = 3

Optimal :-

⇒ 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	3
7	7	7	2	2	2	2	2	2	2	2	2
	0	0	0	0	0	0	4	4	4	0	0
		1	1	1	3	3	3	3	3	3	3
✓	✓	✓	✓	X	✓	X	✓	X	X	✓	X

0	3	2	1	2	0	1	7	0	1
2	2	2	2	2	2	2	7	7	7
0	0	0	0	0	0	0	0	0	0
3	3	3	1	1	1	1	1	1	1
X	X	X	✓	X	X	X	✓	X	X

⇒ No. of Page fault = 9

⇒ No. of Page Hit = 13

⇒ No. of frames = 3.

(3)

* FIFO :-

⇒ 1, 3, 0, 3, 5, 6, 3.

1	3	0	3	5	6	3
1	1	1	1	5	5	5
	3	3	3	3	6	6
		0	0	0	0	3
✓	✓	✓	X	✓	✓	✓

⇒ No. of page fault = 6

⇒ No. of Page Hit = 1

⇒ No. of frames = 3

(4)

* Optimal :-

⇒ 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	7	3	3	3	3	3	3	3	3	3
	0	0	0	0	0	0	0	0	0	0	0	0	0
		1	1	1	1	1	4	4	4	4	4	4	4
			2	2	2	2	2	2	2	2	2	2	2
✓	✓	✓	✓	X	✓	X	✓	X	X	X	X	X	X

=) No. of page fault = 6
=) No. of page Hit = 8
=) No. of frames = 4

(5)
= * LRU :-
=

=) 1, 3, 0, 3, 5, 6, 3.

1	3	0	3	5	6	3
1	1	1	1	5	5	5
	3	3	3	3	3	3
		0	0	0	6	6
✓	✓	✓	x	✓	✓	✓

=) No. of page fault = 6
=) No. of page Hit = 1
=) No. of page frames = 3

----------SHELL PROGRAMMING --> ASSIGNMENT ----------

Write the shell scripts for the following :

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 : 12000
Dearness Allowance : 4800
House Rent : 2400
Gross Salaray : 14400
```

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 : 23

Meters : 23000

Feet :  75459.32

Inches :  905511.84

centimeters :  2300000
```

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 : 3 5

Enter Radius of circle : 4

Perimeter of the Rectangle is :  16

Area of Rectangle is :  15

Circumference of Circle :  25.13272

Area of circle :  50.26544
```

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 : 123456
Total :  21
```

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/nologin
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 | 112 | | /run/uuidd | /usr/sbin/nologin

tcpdump | 108 | 113 | | /nonexistent | /usr/sbin/nologin

sshd | 109 | 65534 | | /run/sshd | /usr/sbin/nologin

landscape | 110 | 115 | | /var/lib/landscape | /usr/sbin/nologin

pollinate | 111 | 1 | | /var/cache/pollinate | /bin/false

p_x | 1000 | 1000 | ,,, | /home/p_x | /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 :

```
echo "---------- Using stat----------"
stat -c "%A %h %U %G %s %.19y %n" *



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


ls -l | tr -s ' ' | cut -d ' ' -f 1-9 -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"}'
```

Output:

```
---------- Using stat----------
-rw-r--r-- 1 p_x p_x 431 2020-11-24 23:46:22 1
-rw-r--r-- 1 p_x p_x 254 2020-11-26 11:37:58 1.sh
-rw-r--r-- 1 p_x p_x 244 2020-11-28 13:07:05 11.sh
-rw-r--r-- 1 p_x p_x 37 2020-11-28 13:11:54 13.sh
-rw-r--r-- 1 p_x p_x 40 2020-11-28 13:13:34 14.sh
-rw-r--r-- 1 p_x p_x 178 2020-11-28 13:23:32 15.sh
```

```
-rw-r--r-- 1 p_x p_x 279 2020-11-28 13:26:41 16.sh
-rw-r--r-- 1 p_x p_x 342 2020-11-28 13:30:34 17.sh
-rw-r--r-- 1 p_x p_x 118 2020-11-28 13:32:56 18.sh
-rw-r--r-- 1 p_x p_x 219 2020-11-28 13:37:19 19.sh
-rw-r--r-- 1 p_x p_x 241 2020-11-26 11:50:26 2.sh
-rw-r--r-- 1 p_x p_x 98 2020-11-28 14:06:02 20.sh
-rw-r--r-- 1 p_x p_x 196 2020-11-28 14:16:05 21.sh
-rw-r--r-- 1 p_x p_x 51 2020-11-28 14:24:06 22.sh
-rw-r--r-- 1 p_x p_x 480 2020-11-28 14:42:22 24.sh
-rw-r--r-- 1 p_x p_x 352 2020-11-26 12:15:07 3.sh
-rw-r--r-- 1 p_x p_x 890 2020-11-28 12:27:00 34.sh
-rw-r--r-- 1 p_x p_x 322 2020-11-28 13:32:18 35.sh
-rw-r--r-- 1 p_x p_x 195 2020-11-28 14:33:11 37.sh
-rw-r--r-- 1 p_x p_x 472 2020-11-28 14:59:43 38.sh
-rw-r--r-- 1 p_x p_x 395 2020-11-28 15:39:36 39.sh
-rw-r--r-- 1 p_x p_x 246 2020-11-26 13:51:18 4.sh
-rw-r--r-- 1 p_x p_x 195 2020-11-28 17:13:16 40.sh
-rw-r--r-- 1 p_x p_x 85 2020-11-28 12:36:42 5.sh
-rw-r--r-- 1 p_x p_x 363 2020-11-28 12:43:54 6.sh
-rw-r--r-- 1 p_x p_x 115 2020-11-28 13:06:36 file1.txt
drwxr-xr-x 2 p_x p_x 4096 2020-10-05 20:14:40 hello
-rwxr-xr-x 1 p_x p_x 798 2020-11-25 00:13:39 hello.sh

---------- Using ls -l ----------
total 112
```

```
-rw-r--r-- 1 p_x p_x 431 Nov 24 23:46 1
-rw-r--r-- 1 p_x p_x 254 Nov 26 11:37 1.sh
-rw-r--r-- 1 p_x p_x 244 Nov 28 13:07 11.sh
-rw-r--r-- 1 p_x p_x 37 Nov 28 13:11 13.sh
-rw-r--r-- 1 p_x p_x 40 Nov 28 13:13 14.sh
-rw-r--r-- 1 p_x p_x 178 Nov 28 13:23 15.sh
-rw-r--r-- 1 p_x p_x 279 Nov 28 13:26 16.sh
-rw-r--r-- 1 p_x p_x 342 Nov 28 13:30 17.sh
-rw-r--r-- 1 p_x p_x 118 Nov 28 13:32 18.sh
-rw-r--r-- 1 p_x p_x 219 Nov 28 13:37 19.sh
-rw-r--r-- 1 p_x p_x 241 Nov 26 11:50 2.sh
-rw-r--r-- 1 p_x p_x 98 Nov 28 14:06 20.sh
-rw-r--r-- 1 p_x p_x 196 Nov 28 14:16 21.sh
-rw-r--r-- 1 p_x p_x 51 Nov 28 14:24 22.sh
-rw-r--r-- 1 p_x p_x 480 Nov 28 14:42 24.sh
-rw-r--r-- 1 p_x p_x 352 Nov 26 12:15 3.sh
-rw-r--r-- 1 p_x p_x 890 Nov 28 12:27 34.sh
-rw-r--r-- 1 p_x p_x 322 Nov 28 13:32 35.sh
-rw-r--r-- 1 p_x p_x 195 Nov 28 14:33 37.sh
-rw-r--r-- 1 p_x p_x 472 Nov 28 14:59 38.sh
-rw-r--r-- 1 p_x p_x 395 Nov 28 15:39 39.sh
-rw-r--r-- 1 p_x p_x 246 Nov 26 13:51 4.sh
-rw-r--r-- 1 p_x p_x 195 Nov 28 17:13 40.sh
-rw-r--r-- 1 p_x p_x 85 Nov 28 12:36 5.sh
-rw-r--r-- 1 p_x p_x 363 Nov 28 12:43 6.sh
```

```
-rw-r--r-- 1 p_x p_x 115 Nov 28 13:06 file1.txt
drwxr-xr-x 2 p_x p_x 4096 Oct 5 20:14 hello
-rwxr-xr-x 1 p_x p_x 798 Nov 25 00:13 hello.sh

total |  |  |  |   |  |
-rw-r--r-- | p_x | p_x | 431 | Nov 24 | 23:46 | 1
-rw-r--r-- | p_x | p_x | 254 | Nov 26 | 11:37 | 1.sh
-rw-r--r-- | p_x | p_x | 244 | Nov 28 | 13:07 | 11.sh
-rw-r--r-- | p_x | p_x | 37 | Nov 28 | 13:11 | 13.sh
-rw-r--r-- | p_x | p_x | 40 | Nov 28 | 13:13 | 14.sh
-rw-r--r-- | p_x | p_x | 178 | Nov 28 | 13:23 | 15.sh
-rw-r--r-- | p_x | p_x | 279 | Nov 28 | 13:26 | 16.sh
-rw-r--r-- | p_x | p_x | 342 | Nov 28 | 13:30 | 17.sh
-rw-r--r-- | p_x | p_x | 118 | Nov 28 | 13:32 | 18.sh
-rw-r--r-- | p_x | p_x | 219 | Nov 28 | 13:37 | 19.sh
-rw-r--r-- | p_x | p_x | 241 | Nov 26 | 11:50 | 2.sh
-rw-r--r-- | p_x | p_x | 98 | Nov 28 | 14:06 | 20.sh
-rw-r--r-- | p_x | p_x | 196 | Nov 28 | 14:16 | 21.sh
-rw-r--r-- | p_x | p_x | 51 | Nov 28 | 14:24 | 22.sh
-rw-r--r-- | p_x | p_x | 480 | Nov 28 | 14:42 | 24.sh
-rw-r--r-- | p_x | p_x | 352 | Nov 26 | 12:15 | 3.sh
-rw-r--r-- | p_x | p_x | 890 | Nov 28 | 12:27 | 34.sh
-rw-r--r-- | p_x | p_x | 322 | Nov 28 | 13:32 | 35.sh
-rw-r--r-- | p_x | p_x | 195 | Nov 28 | 14:33 | 37.sh
-rw-r--r-- | p_x | p_x | 472 | Nov 28 | 14:59 | 38.sh
```

```
-rw-r--r-- | p_x | p_x | 395 | Nov 28 | 15:39 | 39.sh
-rw-r--r-- | p_x | p_x | 246 | Nov 26 | 13:51 | 4.sh
-rw-r--r-- | p_x | p_x | 195 | Nov 28 | 17:13 | 40.sh
-rw-r--r-- | p_x | p_x | 85 | Nov 28 | 12:36 | 5.sh
-rw-r--r-- | p_x | p_x | 363 | Nov 28 | 12:43 | 6.sh
-rw-r--r-- | p_x | p_x | 115 | Nov 28 | 13:06 | file1.txt
drwxr-xr-x | p_x | p_x | 4096 | Oct 5 | 20:14 | hello
-rwxr-xr-x | p_x | p_x | 798 | Nov 25 | 00:13 | hello.sh
```

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 : 30
Enter The Swlling price for Product : 23
Loss :  -7

Enter The Cost Price For Product : 45
Enter The Swlling price for Product : 56
Profit :  11

Enter The Cost Price For Product : 3
Enter The Swlling price for Product : 3
No Profit No Loss
```

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

Enter The Number : 24
EVEN
```

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

Enter the Number : 8
Not Prime

Enter the Number : 9
Not 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

Enter The Year : 2019
Its Not 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
```

```
p_x@DESKTOP-8NGKNR6:~$ ls
1     11.sh  14.sh  16.sh  18.sh  2.sh   21.sh  24.sh  34.sh
37.sh  39.sh  40.sh  6.sh       file2.txt  hello.sh
1.sh  13.sh  15.sh  17.sh  19.sh  20.sh  22.sh  3.sh   35.sh
38.sh  4.sh   5.sh   file1.txt  hello
```

Output :

```
p_x@DESKTOP-8NGKNR6:~$ sh 11.sh file1.txt file2.txt
Files are same
file2.txt Deleted

p_x@DESKTOP-8NGKNR6:~$ ls
1     11.sh  14.sh  16.sh  18.sh  2.sh   21.sh  24.sh  34.sh
37.sh  39.sh  40.sh  6.sh       hello
1.sh  13.sh  15.sh  17.sh  19.sh  20.sh  22.sh  3.sh   35.sh
38.sh  4.sh   5.sh   file1.txt  hello.sh
```

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 :

```
30 November 2020 is a Monday
```

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 Afternoon
```

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:

```
p_x@DESKTOP-8NGKNR6:~$ sh 16.sh
```

```
1) List all Files
2) Print Current Directory
3) Print Date
4) Print Users
1
1     11.sh  14.sh  16.sh  18.sh  2.sh   21.sh  24.sh  34.sh
37.sh  39.sh  40.sh  6.sh       hello
1.sh  13.sh  15.sh  17.sh  19.sh  20.sh  22.sh  3.sh   35.sh
38.sh  4.sh   5.sh   file1.txt  hello.sh
p_x@DESKTOP-8NGKNR6:~$ sh 16.sh
1) List all Files
2) Print Current Directory
3) Print Date
4) Print Users
2
/home/p_x
p_x@DESKTOP-8NGKNR6:~$ sh 16.sh
1) List all Files
2) Print Current Directory
3) Print Date
4) Print Users
3
30-November-2020
p_x@DESKTOP-8NGKNR6:~$ sh 16.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
sshd
landscape
pollinate
p_x
```

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
30
Enter Number 2
20
1) Add
2) Subtract
3) Multiply
4) Divide
5) Modulo Division
1
```

```
50
```

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: 6
720
```

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 ?
7
Fibonacci Series up to 7 terms :
0
1
1
2
3
```

5

8

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 P_x
 ...
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 :

```
p_x@DESKTOP-8NGKNR6:~/try$ ls
26.sh  new1.txt  new2.txt  new3.txt

p_x@DESKTOP-8NGKNR6:~/try$ sh 26.sh

p_x@DESKTOP-8NGKNR6:~/try$ ls
26.sh.587  new1.txt.587  new2.txt.587  new3.txt.587
```

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.

```
p_x@DESKTOP-8NGKNR6:~$ cat unix.txt
unix
hello
i love unix
linux is best

p_x@DESKTOP-8NGKNR6:~$ sh 28.sh unix.txt

Output :

p_x@DESKTOP-8NGKNR6:~$ 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:

```
p_x@DESKTOP-8NGKNR6:~$ sh 29.sh unix.txt
Searching file: unix.txt ...
Found in unix.txt

p_x@DESKTOP-8NGKNR6:~$ sh 29.sh file1.txt
Searching file: file1.txt ...
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 :
//checking new1.txt which is blank
p_x@DESKTOP-8NGKNR6:~$ cat new1.txt

//checking new.txt
p_x@DESKTOP-8NGKNR6:~$ cat new.txt
Hello
My name is P_x
Class MCA 3
Have a good Day
Nice to Meet You

//running the scrpit
p_x@DESKTOP-8NGKNR6:~$ sh 30.sh new.txt new1.txt
'new.txt' copied to -> new1.txt

//checking new1.txt again
p_x@DESKTOP-8NGKNR6:~$ cat new1.txt
Hello
My name is P_x
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:

```
p_x@DESKTOP-8NGKNR6:~$ sh 31.sh
drwxr-xr-x 2 p_x p_x 4096 Oct  5 20:14 hello
-rwxr-xr-x 1 p_x p_x  798 Nov 25 00:13 hello.sh
drwxr-xr-x 2 p_x p_x 4096 Nov 30 15:25 try
```

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:

```
p_x@DESKTOP-8NGKNR6:~$ 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:

```
p_x@DESKTOP-8NGKNR6:~$ 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

p_x@DESKTOP-8NGKNR6:~$ sh 33.sh new.txt
new.txt exists

p_x@DESKTOP-8NGKNR6:~$ sh 33.sh new2.txt
new2.txt file created in "mydir"

p_x@DESKTOP-8NGKNR6:~$ 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

p_x@DESKTOP-8NGKNR6:~$ cd mydir/

p_x@DESKTOP-8NGKNR6:~/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
```

```
Output:
Enter The marks of Subject 1 : 45
Enter The marks of Subject 2 : 45
Enter The marks of Subject 3 : 35
Enter The marks of Subject 4 : 66
Enter The marks of Subject 5 : 23
Percentage =  42.8
Grade : Pass

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
```