Department of Computer Science

Gujarat University

Certificate

Roll No: 16 Seat No: _____

This is to certify that ~~Mr.~~/Ms. MEHTA ADITI GAURANG 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.

10TH DECEMBER, 2020
Date of Submission

Internal Faculty

Head of Department

Department Of Computer Science
Rollwala Computer Centre
Gujarat University

MCA – 3

Subject: - OPERATING SYSTEM

Name: - MEHTA ADITI GAURANG

Roll No.: - 16 **Exam Seat No.: -**

Sr. No.	Contents	Pg. No	Date	Signature
1	Assignment -1	1-28	10-Dec-20	
	Definitions			
	(Introduction of operating system			
	,process management,process			
	Synchronization, memory			
	Management , Storage			
	Management , File System)			
2	Assignment-2	29-40	10-Dec-20	
	Banker's algorithm			
	FIFO algorithm			
	LRU algorithm			
	OPTIMAL algorithm			

24/11/2020 DEFINITION - ASSIGNMENT: 1

1. Application programming Interface (API)

— A standardized library of programming tools used by software developers to write applications that are compatible with a specific operating system or graphic user interface.

2. Base address

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

3. 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.

4. Binary semaphore

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

5. Block

– A collection of contiguous records that are recorded as a unit; the units are separated by interblock gaps. A group of bits that are transmitted as a unit.

6. 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 continually realigns itself as items are inserted and deleted. Thus, all nodes always have a similar no. of keys.

7. Busy waiting

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

8. 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.

9. Central Processing Unit (CPU)

— A portion of a computer that fetches and executes instructions. It consists of an Arithmetic and Logic Unit (ALU), a control unit and registers. Referred to as a processor.

10. Cluster

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

11. Concurrent

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

12. Consumable Resource

— A resource that can be created (produced) and destroyed (consumed). When a resource is acquired by a process, the resource ceases

to exist. Examples of consumable resources are interrupts, signals, messages and information in I/O buffers.

13. Database

- A collection of interrelated data, often with controlled redundancy, organized according to a scheme 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. A common approach is used to add new data and to modify and retrieve existing data.

14. Deadlock

- 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 way as wait state.

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.

15. Demand paging

— The transfer of a page from secondary memory to main memory storage at the moment of need. Compare prepaging.

16. Device Driver

— An operating system module (usually in kernel) that deals directly with device or I/O module.

17. Direct access

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

18. Direct memory access (DMA)

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

19. Disabled interrupt

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

20. Disk allocation table

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

21. Dispatch

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

22. Distributed operating system

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

23. 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.

24. Enabled Interrupt

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

25. External fragmentation

— Occurs when memory is 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 occur between the occupied portion of memory.

26. Field

— Defined logical data that are part of a record. The elementary unit of a record that may contain a data item, a data aggregate, a pointer or a link.

27. File

— A set of related records treated as a unit.

28. File allocation table (FAT)

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

29. 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.

30. File organization

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

31. First come first served (FCFS)

— Same as FIFO

32. First in First out

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

33. 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.

34. 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 result in rapid random retrival.

35. Hit ratio

— In a 2-level memory, the fraction of all memory accesses that are found in the faster memory (eg. the cache).

36. Indexed access

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

37. Indexed 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.

38. Indexed sequential access

— Pertaining to the organisation and accessing of the records of a storage structure through an index of the keys that are sorted in arbitrarily partitioned sequential files.

39. Indexed sequential file

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

40. 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.

41. 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.

42. Internal fragmentation

— Occurs when memory is divided into fixed-size partitions (eg page frame in main memory). If a block of data assigned to one or more partitions, then there may be wasted space in the last partition. This will occur if the last portion of data is smaller than the last partitioned.

43. Interrupt handler

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

44. Job

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

45. Kernel

— A portion of the operating system that includes the most heavily used portion of software. Generally, the kernel is maintained permanently in main memory. The kernel runs in a privileged mode and responds to call from processes and interrupts from devices.

46 Kernel mode

— Also referred to as privileged mode as system mode. A privileged 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 processes executing in a less-privileged mode, and also enables execution of certain machine instructions that are restricted to the kernel mode.

47. Last in first out (LIFO)

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

48. Lightweighted process

— A thread.

49. Livelock

— A condition in which 2 or more processes continuously change their state in response to changes in the other processes without doing any useful work. This is similar to deadlock in that no progress is made, but it different in that neither process is blocked nor waiting for anything.

50. 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 achieved.

51. Logical record

— A record independent of its physical environment, portions of one logical record may be located in different physical record or several logical records or parts of logical records may be located in one physical record.

52. Main memory

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

153. Malicious software

— Any software designed to cause damage to or use up the resources of a target computer. Malicious software (malware) is frequently concealed within a masquerades as legitimate software. In some cases, it spreads itself to other computers via emails or infected disks. Types of malware includes viruses, trojan horses worms, and hidden software for launching denial-of-service attack.

154. 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

55. Memory partitioning

— The subdividing of storage into independent sections.

56. Microkernel

— A small privileged operating system core that provides process scheduling, memory management, and communication services and acts as a trusted process to perform some of the functions traditionally associated with the operating system.

57. Multiprocessing

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

58. Multiprocessor

— A computer that has 2 or more processors that have common access to a main storage.

59. 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 different terminology.

60. Multiprogramming level

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

61. Multitasking

— A mode of operation that provides for the concurrent performance or interleaved execution of 2 or more computer tasks. The same as multi-programming, using different terminologies.

62. 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.

63. Critical section

— In an asynchronous procedure of a computer program, a part that can't be executed simultaneously with an associated critical section of another asynchronous procedure.

64. Operating system

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

65. 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.

66. 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.

67. Page frame

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

68. Paging

— The transfer of pages between main memory and secondary memory.

69. Physical address.

— The absolute location of unit of data in memory. (eg. word or byte in main memory, block on secondary memory).

70. Pipe

— A circular buffer allowing 2 processes to communicate on the producer-consumer model. Thus, it is a FIFO queue, written by one process, read by another. In some system, the pipe is generalized to allow any item in the queue to be selected for consumption.

71. Preemption

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

72. Prepaging

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

73. Process

— A program in execution. A process is controlled and scheduled by the operating system.

74. Process Control Block

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

75. Process State

- All the information that the operating system needs to manage a process and that the processor need to properly execute 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 operating system, such as the priority of the process and whether the process is waiting for the completion of a particular I/O event.

76. Processor

- In a computer, a functional unit that interprets and executes instruction. A processor consist of at least an instruction control unit and an arithmetic unit.

77. Program Counter

- Instruction address register.

78. 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

79. Race Condition

— Situation in which multiple processes access and manipulates shared data with the outcome dependent on relative timing of the processes.

80. Real-time system

— An operating system that must schedule and manage real-time tasks.

81. Real address

— A physical address in main memory

82. Real-time task

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

83. 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 register are used only by the CPU, for control purpose.

184. Relative address

— An address calculated as a displacement from a base address.

85. Remote procedure call (RPC)

— A technique by which 2 programs on different machines interact using procedure call / return syntax and semantics. Both the called and calling program behave as if the partner program were running on the same machine.

86. 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.

87. Round Robin

— A scheduling algorithm in which processes are activated in a fixed cyclic order, that is, all processes are in circular queue. A process that can not proceed because it is waiting for some event (eg. termination of a child process). returns control to the scheduler.

88. Scheduling

— To select jobs as tasks that are to be dispatch. In some operating system, other units of work, such as I/O operations, may also be scheduled.

89. Secondary Memory

— Memory located outside the computer system itself, that is, it can't be processed directly by the processor. It must first be copied into main memory.

90. Segment

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

91. Segmentation

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

92. Semaphore

— An integer value used for signaling among processes. Only three operations may be performed on a semaphore, all of which are atomic: initilize, decrement and increment. Depending on the exact definition of the semaphore, 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 semaphore or a general semaphore.

93) 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.

94. Shell

— The portion of the operating system that

Interpreb Interactive use Commands and job control language commands. It functions as an interface between the use and the Os.

95. Stack

— An ordered list in which items are appended to and delete from the same end of the list, known as top. That is, the next item appended to the list is put on the top, and the 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 LIFO - last in first out.

96. Starvation

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

97. 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.

98. Swapping

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

99. Symmetric multiprocessing

— A form of multiprocessing that allows the OS to execute on any available processor or on several available processors simultaneously.

100. Synchronous operation

— An operation that occurs regularly or predictably with respect to the occurance of a specified event in another process, for example, the calling of an I/O routine that receives control at a precoded location in a computer program.

101. Synchronization

— Situation in which 2 or more processes coordinate their activities based on a condition.

102. System Bus

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

103. Thread

— A dispatchable unit of work. It includes a processor context (which includes the program counter and stack pointer) and its own data area for a stack (to enable subroutine branching). A thread is a lightweighted process that executes sequentially and is interruptible so that the processor can turn to another thread. A process may consist of multiple threads.

104. Thread switch

— The act of switching processor control from one thread to another within the same process.

105. Time sharing

— The concurrent use of a device by a number of users.

106. Time slice

— The maximum amount of time that a process can execute before being interrupted.

107. 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.

108 Trojan Horse

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

109 User mode

— The least-privileged mode of execution. Certain regions of main memory and certain machine instructions can't be used in this mode.

110 Virtual address

— The address of a storage location in virtual memory.

111. Virtual memory

— The storage space that may be regarded as addressable main storage by the user of a computer whose 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 no of main storage locations.

112 Virus

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

113 Weak semaphore

— A semaphore in which all processes waiting on the same semaphore proceed in an unspecified order (i.e. the order is unknown or indeterminate).

114 Word

— An ordered set of bytes or bits that is normal unit in which information may be stored, transmitted or operated on within a given computer.

115. Worm

— Program that can travel from computer to computer across network connections. May contain a virus or a bacteria.

30/11/2020 NUMERICALS : ASSIGNMENT - 2

* PAGE REPLACEMENT ALGORITHMS

1. A system uses 3 page frames for storing process pages in main memory. It uses the First in First out (FIFO) page replacement policy. Assume that all the pages frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below:
4, 7, 6, 1, 7, 6, 1, 2, 7, 2.
Calculate the hit ratio and miss ratio.

- Total number of references = 10

4	7	6	1	7	6	1
		6	6	6	6	6
	7	7	7	7	7	7
4	4	4	1	1	1	1
✓	✓	✓	✓	X	X	X

2	7	2
6	7	7
2	2	2
1	1	1
✓	✓	X

From here,

Total number of page fault occured = 6.

Total number of page hits
= Total number of references - Total number of page misses or page fault

= 10 - 6

= 4

∴ Hit ratio
= Total number of page hits / Total number of references.

= 4/10

= 0.4 or 40%.

Now,
Miss ratio
= Total number of page misses / Total number of references

= 6/10

= 0.6 or 60%.

→ If we use (LRU) i.e. Least recently used page replacement policy, then the hit ratio and miss ratio are same as FIFO page replacement policy.

12. A system uses 3 page frames for storing process pages in main memory. It uses Optimal page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below:
4, 7, 6, 1, 7, 6, 1, 2, 7, 2.
Also calculate hit ratio and miss ratio.

- Total number of references = 10

4	7	6	1	7	6	1
		6	6	6	6	6
	7	7	7	7	7	7
4	4	4	1	1	1	1
✓	✓	✓	✓	X	X	X

2	7	2
2	2	2
7	7	7
1	1	1
✓	X	X

From here,

Total number of page fault occured = 5

= Total number of page hits
= Total number of references − Total number of page miss or page faults

= 10 − 5

= 5

∴ Hit ratio
= Total number of page hits / Total number of references.

= 5/10

= 0.5 or 50%

Now,
Miss ratio
= Total number of page misses / Total number of references

= 5/10

= 0.5 or 50%

* BANKER'S ALGORITHM

1. A single processor system has three resources types X, Y, Z which are shared by 3 processes. There are 5 units of each resource type. Consider the following scenario, where the column Alloc denotes the number of units of each resources type allocated to each process and the column request denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish LAST (P0, P1, P2, none of the above since the system is in a deadlock)

	Alloc			Request		
	X	Y	Z	X	Y	Z
P_0	1	2	1	1	0	3
P_1	2	0	1	0	1	2
P_2	2	2	1	1	2	0

- Total = [X Y Z] = [5 5 5]
  Total_Allocation = [X Y Z] = [5 4 3]

Now
Available = Total - Total_Alloc
= [5 5 5] - [5 4 3]
= [0 1 2]

(a) With the instances available currently, only the requirement of the process P1 can be satisfied.

So, process P1 is allocated the requested resources.

It completes its execution and then free up the instances of resources held by it.

Available = [0 1 2] + [2 0 1]
= [2 1 3]

(b) With the instances available currently, only the requirement of the process P0 can be satisfied.

So, process P0 is allocated the requested resource.

It completes its execution and then free up the instances of resources held by it..

Available = [2 1 3] + [1 2 1]
= [3 3 4]

(c) With the instances available currently, the requirement of the process P_2 can be satisfied.

So, process P_2 is allocated the requested resources.

It completes its execution and then free up the instances of resources held by it.

Available = [3 3 4] + [2 2 1]
= [5 5 5]

Thus,

There exist a safe sequence P1, P0, P_2 in which all the processes can be executed.

So, the system is in a safe state.

Process P_2 will be executed last.

2. A system has 4 processes and 5 allocatable resource. The current allocation and maximum needs are as follows -

	Allocated					Maximum				
A	1	0	2	1	1	1	1	2	1	3
B	2	0	1	1	0	2	2	2	1	0
C	1	1	0	1	1	2	1	3	1	1
D	1	1	1	1	0	1	1	2	2	0

If Available = [0 0 x 1 1], what is the smallest value of x for which this is a safe state?

— Need = Maximum − Allocation

	Need				
A	0	1	0	0	2
B	0	2	1	0	0
C	1	0	3	0	0
D	0	0	1	1	0

(a) — For x = 0
If x = 0, then
Available = [0 0 0 1 1]

With the instances available currently, the requirement of any process can not be satisfied.

So, for X = 0, system remains in a deadlock which is an unsafe state.

(b) — For X = 1

If X = 1, then

Available = [0 0 1 1 1]

With the instances available currently, only the requirement of the process D can be satisfied.

So, process D is allocated the requested resources.

It completes its execution and then free up the instances of resources held by it.

Available = [0 0 1 1 1] + [1 1 1 1 0]

= [1 1 2 2 1]

With the instances available currently, the requirement of any process can't be satisfied.

So, for X = 1, system remains in a deadlock which is an unsafe state.

(C) For X = 2

– If X = 2, then

Available = [0 0 2 1 1]

With the instances available currently, only the requirement of all the processes A, B, C and D can be satisfied.

So, processes A, B, C, D are allocated the requested resources one by one.

They complete their execution and then free up the instances of resources held by it.

Available = [0 0 2 1 1] + [1 1 1 1 0] + [1 1 0 1 1]
+ [1 0 2 1 1] + [2 0 1 1 0]
= [5 2 6 5 3]

There exits a safe sequence in which all the processes can be executed.

So, the system in a safe state.

Thus, the minimum value of X that ensures system is in safe state = 2.

* BELADY'S ANOMALY PAGE FAULT

→ Consider the reference string is :
0, 1, 2, 3, 0, 1, 4, 0, 1, 2, 3, 4.

(i) Perform belady's anomaly page fault for Optimal Page Replacement Algorithm.

– When frame size = 3

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

0	1	2	3	4	
4	4	4	4	4	Number of page fault = 7
1	1	1	1	1	
0	0	2	3	3	

– When frame size = 4

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

0	1	2	3	4	
4	4	4	4	4	Number of page fault = 6
2	2	2	2	2	
1	1	1	1	1	
0	0	0	3	3	

(ii) LRU page replacement algorithm

- When page / frame size = 3
  Total number of page fault = 10

- When frame size = 4
  Total number of page fault = 8

So from both optimal and LRU page replacement algorithm, we can conclude that :

- At all steege ℞ Case - 02 (frame size = 4) main memory compulsorily contains the set of pages that are present in the corresponding steiteges ℞ Case - 01 (frame size = 3).

- Thus, LRU and optimal page replacement algorithm follows the stack property.

- Hence it does not suffer from Belady's Anomaly.

- As a proof, number of page fault decreases when the number of frames is increased from 3 to 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. – 3

R O L L NO : 16
N A M E : MEHTA ADITI GAURANG
S U B J E C T : OPERATING SYSTEM

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.	1	10-Dec-20	
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.	1	10-Dec-20	
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.	1	10-Dec-20	
4	If a five digit number is entered through the keyboard, calculate the sum of its digits.	2	10-Dec-20	
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.	2	10-Dec-20	
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.	3	10-Dec-20	
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.	5	10-Dec-20	
8	Check whether the entered no. is odd or even.	5	10-Dec-20	
9	Check whether the entered no. is prime or not.	5	10-Dec-20	
10	Check whether the entered year is a leap year or not.	6	10-Dec-20	
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	6	10-Dec-20	
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.	7	10-Dec-20	
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.	7	10-Dec-20	
14	Write a shell script to display the date with the format :-	8	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. – 3

R O L L NO : 16
N A M E : MEHTA ADITI GAURANG
S U B J E C T : OPERATING SYSTEM

	25th October 2005 is a Tuesday.			
15	**Write a shell script to display the appropriate message like : Good Morning / Good Afternoon / Good Evening**	**8**	**10-Dec-20**	
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".**	**8**	**10-Dec-20**	
17	**Create a menu driven calculator which asks for two integers and perform basic arithmetic operations.**	**10**	**10-Dec-20**	
18	**Find the factorial of any number.**	**10**	**10-Dec-20**	
19	**Display the fibonacci series upto some number**	**11**	**10-Dec-20**	
20	**Two numbers are entered through the keyboard, find the power, one number raised to another.**	**11**	**10-Dec-20**	
21	**Write a script which has the functionality similar to head and tail commands.**	**11**	**10-Dec-20**	
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**	**12**	**10-Dec-20**	
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.**	**12**	**10-Dec-20**	
24	**Print the prime nos. from 1 to 300.**	**12**	**10-Dec-20**	
25	**Program must display all the combinations of 1, 2, and 3.**	**14**	**10-Dec-20**	
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.**	**15**	**10-Dec-20**	
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.**	**15**	**10-Dec-20**	
28	**Write a shell script which deletes all the lines containing the word "unix" in the files supplied as arguments to it.**	**16**	**10-Dec-20**	
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**	**16**	**10-Dec-20**	

R O L L NO : 16
N A M E : MEHTA ADITI GAURANG
S U B J E C T : OPERATING SYSTEM

	be displayed on the screen.			
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.	16	10-Dec-20	
31	The script displays a list of all files in the current directory to which you have read, write and execute permissions.	17	10-Dec-20	
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.	17	10-Dec-20	
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.	18	10-Dec-20	
34	Accept the marks of 5 subjects and calculate the percentage and grade. 35. Print armstrog nos. from 1 to 500.	19	10-Dec-20	
35	Print armstrog nos. from 1 to 500.	20	10-Dec-20	
36	Accept the measure (angles) of a triangle and display the type of triangle. (eg. acute, right, obtuse)	20	10-Dec-20	
37	Display all the numbers from 1 to 100 which are divisible by 7	20	10-Dec-20	
38	Find the largest and smallest of 3 different numbers.	21	10-Dec-20	
39	Find HCF and LCM of a given no.	22	10-Dec-20	
40	Display the dates falling on Sundays of the current month.	22	10-Dec-20	
41	In a college, students are allowed to select any one sporting event during his studies. Create two files as mentioned below : File : sports.dat Code Game --- 101Cricket 102Football 103Tennis 104Badminton	23	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. – 3

R O L L NO : 16
N A M E : MEHTA ADITI GAURANG
S U B J E C T : OPERATING SYSTEM

	File : students.dat **Name Code** **----------** **Aamir 101** **Sharukh 103** **Salman 103** **Ajay 102** **Write a shell script to list the students according to their choice of games …** **Eg. Cricket : Aamir** **Football : Ajay** **Tennis : Sharukh, Salman**			
42	**Write a shell script to generate summary from the sales.dat file.** **Sales made by 3 salesman by selling 3 products are entered in a file. Add atleast 10 records. The format is as shown below:** **Salesman:Product1:Product2:Product3** **Sample data:** **Mr. Abhishek Sharma:21:29:12** **Mr. Akash Srivastava:11:15:28** **Mr. Abhilash Dwivedi:31:04:13** **Calculate the followings :** **Total sales of the company** **Highest sold product** **Best salesman (who sold the most)** **Worst salesman (who sold the least)**	**24**	**10-Dec-20**	
43	**Create a file "medals.dat" which contains the details of medals won by each country in Olympics. The data in the file may be as given below :** **(Country name is Primary key.)** **Country Gold Silver Bronze** **----------** **India 21 12 15** **Pakistan 12 10 08** **USA 10 14 19** **Srilanka 00 09 07** **…….and so on……..**	**25**	**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. – 3

R O L L NO : 16
N A M E : MEHTA ADITI GAURANG
S U B J E C T : OPERATING SYSTEM

	Write a shell script which will ask the user to enter the Country name, further modify the no. of medals for that country. **Delete all the countries who get zero Gold medals.** **Calculate the total no. of medals won by each country.** **Find the country with highest Gold medals.**			
44	**Write a shell script to generate summary from a file : "student.dat" with following format :** **Student_no : student_name : gender : marks1 : marks2 : marks3** **Each field must be separated by a delimeter '-'** **Process the following queries:** **Calculate the total marks of each student** **Calculate the percentage of marks for each student** **Count the total number of male and female students** **Count the total number of students who pass and those who fail.**	**27**	**10-Dec-20**	
45	**A reputed MCA institute of Gujarat has students from various states. A sample file "students.dat" is shown as under :** **State M F** **----------** **Gujarat 18 12** **Maharashtra 12 04** **M.P. 08 04** **UP 05 00** **Rajasthan 07 00** **Total Male candidates are 50 and Female are 20. Write a shell script to generate a Statewise Candidate Distribution Report as under :** **STATEWISE CANDIDATES LISTING** **----------** **STATE %MALE %FEMALE TOTAL** **----------**	**28**	**10-Dec-20**	

R O L L NO : 16
N A M E : MEHTA ADITI GAURANG
S U B J E C T : OPERATING SYSTEM

	---------- GUJARAT 36 60 30 MAHARASHTRA 24 20 16 …….. ……. And so on ……..			
46	Write a Shell script to generate summary from a file "books.dat" which contains the following details : Field Description ---------- No Numeric (4) – uniquely identifies each book. Title Alphanumeric(30) – title of the book Author Character(20) – Author of the book Publisher Character(20) – Publisher (PHI , TMH, BPB…) Edition Numeric (2) Sample Data: b1001Programming in Java Balagurusway TMH Second b1002Computer Networks Tanenbaum Pearson Fifth b1003Operating Systems Chaudhari Jaico First After creating the file do the followings : Your script must replace all the BPB publisher with TMH. List the titles with the name 'Java'. List the books written 'Balaguruswamy List the books which are not the first edition	29	10-Dec-20	
47	Create a file "election.dat" which contains the Election details for a specific city. Field Description ---------- Idno Numeric - Unique Name Character – Voter's Name Sex Character – M / F Age Numeric Ward Numeric – Ward no. / Division no. of the city. Sample data:	30	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. – 3

R O L L NO : 16
N A M E : MEHTA ADITI GAURANG
S U B J E C T : OPERATING SYSTEM

	e101-abhishek-M-35-44 **e102-ashutosh-M-97-14** **e103-anamika-F-21-50** **Suppose the same file is to be modified after 4 years. Write a shell script to simulate this process.** **Your program must update the age of all People (Add 4 years to age). In case if the age exceeds 99 then delete the record from the file, assuming that the person is dead.** **Display the election.dat and final output of your program.**			
48	**In a college, students are allowed to select any one elective subject during his studies. Create two files by entering the data as mentioned below (you may skip the heading line if required) :** **File : elective.dat** **Code Game** **---** **101 AI** **102 Computer Graphics** **103 Parallel Processing** **104 Data Mining** **File : students.dat** **RollNo. Name Code** **---** **1 Sonal 101** **2 Madhu 101** **3 Mahim 103** **4 Esha 104** **Write a shell script to list the students according to their choice of electives …** **Eg. AI :- Sonal, Madhu** **Computer Graphics: -** **Parallel Processing :- Mahim** **Data Mining :- Esha**	**31**	**10-Dec-20**	
49	**Create two files: subjects.dat and students.dat containing the subject details and the student details. Sample data is as shown below:** **subjects.dat** **Course_id-Semester_id-Subject_id-Subject_name** **CS-1-1-FCO** **CS-1-2-FOP** **CS-1-3-SL**	**32**	**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. – 3

R O L L NO : 16
N A M E : MEHTA ADITI GAURANG
S U B J E C T : OPERATING SYSTEM

	CS-2-1-DS **CS-2-2-DBMS** **CS-3-1-OS** **CS-3-2-JAVA** **faculty.dat** **Faculty_id:Semester_id:Subject_id** **F1-2-1** **F2-3-2** **F3-1-3** **F1-1-1** **Write a shell script to list the faculties and their respective subjects. Sample Output will be :** **F1 : FCO, DS** **F2 : JAVA** **F3 : SL**			
50	**Create two files employee.dat and departments.dat and add atleast 10 records in the following format :** **employee.dat** **emp_id:department_id:birthdate** **e101:M1:11-01-1960** **e102:C1:21-03-1973** **e103:M2:21-03-1973** **e104:C1:21-03-1973** **e105:B1:08-10-1965** **e101:M1:11-11-1964** **departments.dat** **departmend_id:department_name** **B1:Botany** **C1:Chemistry** **M1:Mathematics** **M2:Management** **Write a shell script to do the followings:** **1)List all the employee_ids department-wise** **2)List the employee_ids born after 1970** **3)List the employee_ids according to birthdate in sorted order**	**33**	**10-Dec-20**	

==========
ROLLNO : 16
NAME : ADITI MEHTA
CLASS : MCA-3
SUBJECT : OPERATING SYSTEM
==========

----------SHELL PROGRAMMING ASSIGNMENT 1----------

==========
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
aditi | 1000 | 1000 | ,,, | /home/aditi | /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 aditi aditi 431 2020-11-24 23:46:22 1
-rw-r--r-- 1 aditi aditi 254 2020-11-26 11:37:58 1.sh
-rw-r--r-- 1 aditi aditi 244 2020-11-28 13:07:05 11.sh
-rw-r--r-- 1 aditi aditi 37 2020-11-28 13:11:54 13.sh
-rw-r--r-- 1 aditi aditi 40 2020-11-28 13:13:34 14.sh
-rw-r--r-- 1 aditi aditi 178 2020-11-28 13:23:32 15.sh
-rw-r--r-- 1 aditi aditi 279 2020-11-28 13:26:41 16.sh
-rw-r--r-- 1 aditi aditi 342 2020-11-28 13:30:34 17.sh
-rw-r--r-- 1 aditi aditi 118 2020-11-28 13:32:56 18.sh
-rw-r--r-- 1 aditi aditi 219 2020-11-28 13:37:19 19.sh
-rw-r--r-- 1 aditi aditi 241 2020-11-26 11:50:26 2.sh
-rw-r--r-- 1 aditi aditi 98 2020-11-28 14:06:02 20.sh
-rw-r--r-- 1 aditi aditi 196 2020-11-28 14:16:05 21.sh
-rw-r--r-- 1 aditi aditi 51 2020-11-28 14:24:06 22.sh
-rw-r--r-- 1 aditi aditi 480 2020-11-28 14:42:22 24.sh
-rw-r--r-- 1 aditi aditi 352 2020-11-26 12:15:07 3.sh
-rw-r--r-- 1 aditi aditi 890 2020-11-28 12:27:00 34.sh
-rw-r--r-- 1 aditi aditi 322 2020-11-28 13:32:18 35.sh
-rw-r--r-- 1 aditi aditi 195 2020-11-28 14:33:11 37.sh
-rw-r--r-- 1 aditi aditi 472 2020-11-28 14:59:43 38.sh
-rw-r--r-- 1 aditi aditi 395 2020-11-28 15:39:36 39.sh
-rw-r--r-- 1 aditi aditi 246 2020-11-26 13:51:18 4.sh
-rw-r--r-- 1 aditi aditi 195 2020-11-28 17:13:16 40.sh
-rw-r--r-- 1 aditi aditi 85 2020-11-28 12:36:42 5.sh
-rw-r--r-- 1 aditi aditi 363 2020-11-28 12:43:54 6.sh
-rw-r--r-- 1 aditi aditi 115 2020-11-28 13:06:36 file1.txt
drwxr-xr-x 2 aditi aditi 4096 2020-10-05 20:14:40 hello
-rwxr-xr-x 1 aditi aditi 798 2020-11-25 00:13:39 hello.sh

---------- Using ls -l ----------
```

```
total 112
-rw-r--r-- 1 aditi aditi 431 Nov 24 23:46 1
-rw-r--r-- 1 aditi aditi 254 Nov 26 11:37 1.sh
-rw-r--r-- 1 aditi aditi 244 Nov 28 13:07 11.sh
-rw-r--r-- 1 aditi aditi 37 Nov 28 13:11 13.sh
-rw-r--r-- 1 aditi aditi 40 Nov 28 13:13 14.sh
-rw-r--r-- 1 aditi aditi 178 Nov 28 13:23 15.sh
-rw-r--r-- 1 aditi aditi 279 Nov 28 13:26 16.sh
-rw-r--r-- 1 aditi aditi 342 Nov 28 13:30 17.sh
-rw-r--r-- 1 aditi aditi 118 Nov 28 13:32 18.sh
-rw-r--r-- 1 aditi aditi 219 Nov 28 13:37 19.sh
-rw-r--r-- 1 aditi aditi 241 Nov 26 11:50 2.sh
-rw-r--r-- 1 aditi aditi 98 Nov 28 14:06 20.sh
-rw-r--r-- 1 aditi aditi 196 Nov 28 14:16 21.sh
-rw-r--r-- 1 aditi aditi 51 Nov 28 14:24 22.sh
-rw-r--r-- 1 aditi aditi 480 Nov 28 14:42 24.sh
-rw-r--r-- 1 aditi aditi 352 Nov 26 12:15 3.sh
-rw-r--r-- 1 aditi aditi 890 Nov 28 12:27 34.sh
-rw-r--r-- 1 aditi aditi 322 Nov 28 13:32 35.sh
-rw-r--r-- 1 aditi aditi 195 Nov 28 14:33 37.sh
-rw-r--r-- 1 aditi aditi 472 Nov 28 14:59 38.sh
-rw-r--r-- 1 aditi aditi 395 Nov 28 15:39 39.sh
-rw-r--r-- 1 aditi aditi 246 Nov 26 13:51 4.sh
-rw-r--r-- 1 aditi aditi 195 Nov 28 17:13 40.sh
-rw-r--r-- 1 aditi aditi 85 Nov 28 12:36 5.sh
-rw-r--r-- 1 aditi aditi 363 Nov 28 12:43 6.sh
-rw-r--r-- 1 aditi aditi 115 Nov 28 13:06 file1.txt
drwxr-xr-x 2 aditi aditi 4096 Oct 5 20:14 hello
-rwxr-xr-x 1 aditi aditi 798 Nov 25 00:13 hello.sh

total |  |  |  |  |  |
-rw-r--r-- | aditi | aditi | 431 | Nov 24 | 23:46 | 1
-rw-r--r-- | aditi | aditi | 254 | Nov 26 | 11:37 | 1.sh
-rw-r--r-- | aditi | aditi | 244 | Nov 28 | 13:07 | 11.sh
-rw-r--r-- | aditi | aditi | 37 | Nov 28 | 13:11 | 13.sh
-rw-r--r-- | aditi | aditi | 40 | Nov 28 | 13:13 | 14.sh
-rw-r--r-- | aditi | aditi | 178 | Nov 28 | 13:23 | 15.sh
-rw-r--r-- | aditi | aditi | 279 | Nov 28 | 13:26 | 16.sh
-rw-r--r-- | aditi | aditi | 342 | Nov 28 | 13:30 | 17.sh
-rw-r--r-- | aditi | aditi | 118 | Nov 28 | 13:32 | 18.sh
-rw-r--r-- | aditi | aditi | 219 | Nov 28 | 13:37 | 19.sh
-rw-r--r-- | aditi | aditi | 241 | Nov 26 | 11:50 | 2.sh
-rw-r--r-- | aditi | aditi | 98 | Nov 28 | 14:06 | 20.sh
-rw-r--r-- | aditi | aditi | 196 | Nov 28 | 14:16 | 21.sh
-rw-r--r-- | aditi | aditi | 51 | Nov 28 | 14:24 | 22.sh
-rw-r--r-- | aditi | aditi | 480 | Nov 28 | 14:42 | 24.sh
-rw-r--r-- | aditi | aditi | 352 | Nov 26 | 12:15 | 3.sh
-rw-r--r-- | aditi | aditi | 890 | Nov 28 | 12:27 | 34.sh
-rw-r--r-- | aditi | aditi | 322 | Nov 28 | 13:32 | 35.sh
-rw-r--r-- | aditi | aditi | 195 | Nov 28 | 14:33 | 37.sh
-rw-r--r-- | aditi | aditi | 472 | Nov 28 | 14:59 | 38.sh
-rw-r--r-- | aditi | aditi | 395 | Nov 28 | 15:39 | 39.sh
-rw-r--r-- | aditi | aditi | 246 | Nov 26 | 13:51 | 4.sh
-rw-r--r-- | aditi | aditi | 195 | Nov 28 | 17:13 | 40.sh
-rw-r--r-- | aditi | aditi | 85 | Nov 28 | 12:36 | 5.sh
-rw-r--r-- | aditi | aditi | 363 | Nov 28 | 12:43 | 6.sh
-rw-r--r-- | aditi | aditi | 115 | Nov 28 | 13:06 | file1.txt
```

drwxr-xr-x | aditi | aditi | 4096 | Oct 5 | 20:14 | hello
-rwxr-xr-x | aditi | aditi | 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 : 7
ODD

Enter The Number : 22
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 : 22
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
```

```
aditi@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 : :

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

aditi@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
```

==========

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.

==========

```
echo "Enter user name "
read name
who | grep $name > /dev/null
if [ $? -eq 0 ]
then
echo "User is logged in "
echo "Please enter a message to send him "
read msg
echo $msg
else
echo "User is not logged in "
fi
```

Output :

```
aditi@aditi-VirtualBox:~/aditi$ sh s_s_12.sh
Enter user name
aditi
User is logged in
Please enter a message to send him
Hii !! How are you ?
```

```
Hii !! How are you ?
aditi@aditi-VirtualBox:~/aditi$ sh s_S_12.sh
Enter user name
jaladhi
User is not logged in
aditi@aditi-VirtualBox:~/aditi$
```

==========

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 :

```
aditi@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
aditi@DESKTOP-8NGKNR6:~$ sh 16.sh
1) List all Files
2) Print Current Directory
3) Print Date
4) Print Users
2
/home/aditi
aditi@DESKTOP-8NGKNR6:~$ sh 16.sh
1) List all Files
2) Print Current Directory
3) Print Date
4) Print Users
3
30-November-2020
aditi@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
aditi

==========

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 aditi
  ...
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 : :

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

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

aditi@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.

==========

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

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

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

aditi@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

```
aditi@DESKTOP-8NGKNR6:~$ cat new1.txt
```

//checking new.txt

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

//running the scrpit

```
aditi@DESKTOP-8NGKNR6:~$ sh 30.sh new.txt new1.txt
'new.txt' copied to -> new1.txt
```

//checking new1.txt again

```
aditi@DESKTOP-8NGKNR6:~$ cat new1.txt
Hello
My name is aditi
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 :

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

```
aditi@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 :

```
aditi@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

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

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

aditi@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

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

aditi@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
```

==========
ROLLNO : 16
NAME : ADITI MEHTA
CLASS : MCA-3
SUBJECT : OPERATING SYSTEM
==========
----------SHELL PROGRAMMING ASSIGNMENT 2----------
==========
Write the shell scripts for the following :
==========
41.In a college, students are allowed to select any one sporting event during his studies. Create two files as mentioned below :

File : sports.dat
Code Game
---
101 Cricket
102 Football
103Tennis
104Badminton

File : students.dat
Name Code

Aamir 101
Sharukh 103
Salman 103
Ajay 102

Write a shell script to list the students according to their choice of games …
Eg. Cricket : Aamir
Football : Ajay
Tennis : Sharukh, Salman
==========

```
while read line
do
        echo $line

done < sports.dat

echo -e "Choose sport name : "
read sport

code=`grep $sport sports.dat | cut -d ":" -f1`

echo "Following students play $sport"
`grep $code students.dat | cut -d " " -f1 > display.txt`
cat display.txt
rm display.txt
```

Output :

```
101:Cricket
102:Football
103:Tennis
104:Badminton
```

Choose sport name : Cricket
Following students play Cricket
Raviraj
Aditi
==========

42.Write a shell script to generate summary from the sales.dat file.
Sales made by 3 salesman by selling 3 products are entered in a file. Add atleast 10 records. The format is as shown below:
Salesman:Product1:Product2:Product3

Sample data:
Mr. Abhishek Sharma:21:29:12
Mr. Akash Srivastava:11:15:28
Mr. Abhilash Dwivedi:31:04:13

Calculate the followings :
Total sales of the company
Highest sold product
Best salesman (who sold the most)
Worst salesman (who sold the least)
==========

```
sp1=0
sp2=0
sp3=0
tsum=0

while read line
do
        p1=`echo $line | cut -d ":" -f2`
        p2=`echo $line | cut -d ":" -f3`
        p3=`echo $line | cut -d ":" -f4`
        sp1=`expr $sp1 + $p1`
        sp2=`expr $sp2 + $p2`
        sp3=`expr $sp3 + $p3`

        t=`expr $p1 + $p2 + $p3`
        sman=`echo $line | cut -d ":" -f1`
        echo "$sman $t" >> highsaleman.txt

done < Sales.dat

tsum=`expr $sp1 + $sp2 + $sp3`
echo "Total sale of company : $tsum"

echo "Product1 $sp1" > high.txt
echo "Product2 $sp2" >> high.txt
echo "Product3 $sp3" >> high.txt

echo "`sort -k2 -r -n high.txt | head -1 | cut -d " " -f1` is highest selling product"

echo "`sort -k4 -r -n highsaleman.txt | head -1 | cut -d " " -f 1-3` is best salesman"

rm high.txt
rm highsaleman.txt
```

Output :

Total sale of company : 367
Product1 is highest selling product
Ms. Aditi is best salesman

==========

43.Create a file “medals.dat” which contains the details of medals won by each country in Olympics. The data in the file may be as given below :
(Country name is Primary key.)

Country	Gold	Silver	Bronze
India	21	12	15
Pakistan	12	10	08
USA	10	14	19
Srilanka	00	09	07

……..and so on……..

Write a shell script which will ask the user to enter the Country name, further modify the no. of medals for that country.
Delete all the countries who get zero Gold medals.
Calculate the total no. of medals won by each country.
Find the country with highest Gold medals.

==========

```
cat "medals.dat"

echo -n "Enter country name to modify medals : "
read country
echo -n "Enter gold medals :"
read gold
echo -n "Entrer silver medals : "
read silver
echo -n "Enter bronze medals : "
read bronze

while read line
do
        set $line
        if [ $country == $1 ]
        then
                echo "$1 $gold $silver $bronze" >> "temp.dat"
        else
                echo $line >> "temp.dat"
        fi
done < "medals.dat"

cp "temp.dat" "medals.dat"

echo "Medal list after update"
cat "medals.dat"

rm temp.dat

echo "----------------------------------------"
while read line
do
        set $line
        if [ $2 -eq 00 ]
```

```
        then
                echo "$1 country deleted"
        else
                echo $line >> "temp.dat"
        fi
done < "medals.dat"

cp "temp.dat" "medals.dat"
cat "medals.dat"

rm temp.dat
echo "----------------------------------------"

while read line
do
        set $line
        total=`expr $2 + $3 + $4`
        echo "Total medals won by $1 = $total"
        echo "$1 $2" >> highmedals.txt

done < "medals.dat"

echo "----------------------------------------"
echo "`sort -k2 -n -r highmedals.txt | head -1 | cut -d " " -f1` won the highest
gold medals"

rm highmedals.txt
```

Output :

```
India 22 15 16
Pakistan 12 10 08
USA 10 14 19
Srilanka 00 09 07

Enter country name to modify medals : India
Enter gold medals :23
Entrer silver medals : 15
Enter bronze medals : 17

Medal list after update
India 23 15 17
Pakistan 12 10 08
USA 10 14 19
Srilanka 00 09 07
----------------------------------------
Srilanka country deleted
India 23 15 17
Pakistan 12 10 08
USA 10 14 19
----------------------------------------
Total medals won by India = 55
Total medals won by Pakistan = 30
Total medals won by USA = 43
----------------------------------------
India won the highest gold medals
```

==========
44.Write a shell script to generate summary from a file : “student.dat” with following format :

Student_no : student_name : gender : marks1 : marks2 : marks3

Each field must be separated by a delimeter ‘-‘

Process the following queries:
Calculate the total marks of each student
Calculate the percentage of marks for each student
Count the total number of male and female students
Count the total number of students who pass and those who fail.
==========
cat student.dat

```
male=0
female=0
pass=0
fail=0

while read line
do
        set $line
        total=`expr $4 + $5 + $6`
        echo "$2 obtained $total marks out of 300"
        per=`expr $total \* 100 / 300`
        echo "$2 obtained $per percentage"

        if [ $3 == 'M' ]
        then
                male=`expr $male + 1`
        else
                female=`expr $female + 1`
        fi

        if [ $4 -ge 35 -a $5 -ge 35 -a $6 -ge 35 ]
        then
                pass=$(($pass + 1))
        else
                fail=$(($fail + 1))
        fi
done < student.dat

echo "Toatal male students = $male"
echo "Total female students = $female"
echo "Total pass students = $pass"
echo "Total fail students = $fail"
```

Output :

```
22 Raviraj M 85 89 78
07 Aditi F 80 75 76
03 Aditya M 81 70 75
05 AchyutM 79 69 70
02 Pratik M 58 30 58
26 Jaladhi F 45 56 58
```

Raviraj obtained 252 marks out of 300
Raviraj obtained 84 percentage
Aditi obtained 231 marks out of 300
Aditi obtained 77 percentage
Aditya obtained 226 marks out of 300
Aditya obtained 75 percentage
Pardip obtained 218 marks out of 300
Achyut obtained 72 percentage
Pratik obtained 146 marks out of 300
Pratik obtained 48 percentage
Jaladhi obtained 159 marks out of 300
Jaladhi obtained 53 percentage

Total male students = 4
Total female students = 2
Total pass students = 5
Total fail students = 1

==========

45.A reputed MCA institute of Gujarat has students from various states. A sample file "students.dat" is shown as under :

```
State                   M          F
-------------------------------------
Gujarat                 18         12
Maharashtra     12          04
M.P.                    08         04
UP                      05         00
Rajasthan          07          00
```

Total Male candidates are 50 and Female are 20. Write a shell script to generate a Statewise Candidate Distribution Report as under :

```
                                    STATEWISE CANDIDATES LISTING
-------------------------------------------------------------------------------------------
                STATE                   %MALE                   %FEMALE                 TOTAL
-------------------------------------------------------------------------------------------
                GUJARAT                 36                              60
30
                MAHARASHTRA     24                              20
16
…….. ……. And so on ……..
```

==========

```
echo "                    STATEWISE CANDIDATES LISTING                    "
echo "----------------------------------------------------------"
echo "    State          %Male                   %Female                 Total"
echo "----------------------------------------------------------"

while read line
do
        set $line
        mper=`expr $2 \* 100 / 50`
        fper=`expr $3 \* 100 / 20`

        echo "$1          $mper          $fper          `expr $2 + $3` "

done < "candidate.dat"
```

Output :

```
              StateWise Candidate Listing
-----------------------------------------------------
State          %Male                  %Female            Total
-----------------------------------------------------
Gujarat              36                    60                30
Maharastra         24              20                16
M.P.                 16              20                12
UP                    10                        0                 5
Rajasthan          14                0                  7
```

==========

46.Write a Shell script to generate summary from a file “books.dat” which contains the following details :

Field	Description
No	Numeric (4) – uniquely identifies each book.
Title	Alphanumeric(30) – title of the book
Author	Character(20) – Author of the book
Publisher	Character(20) – Publisher (PHI , TMH, BPB…)
Edition	Numeric (2)

Sample Data:

```
b1001Programming in Java          Balagurusway          TMH                    Second
b1002Computer Networks             Tanenbaum              Pearson              Fifth
b1003Operating Systems              Chaudhari               Jaico
First
```

After creating the file do the followings :

- Your script must replace all the BPB publisher with TMH.
- List the titles with the name ‘Java’.
- List the books written ‘Balaguruswamy
- List the books which are not the first edition

==========

```
echo "File before modify"
cat "books.dat"

echo "File after modify"
cat "books.dat" | tr "TMH" "BPB"

echo "Books named as java"
grep "java" "books.dat" | cut -d " " -f2-4

echo "Books written by balagurusway are as follow:"
grep "Balagurusway"    books.dat | cut -d " " -f2-4

echo "Books which are not first edition:"
grep -v "First" "books.dat" | cut -d " " -f2-4
```

Output :

```
File before modify
b1001 Programming in java Balagurusway TMH Second
b1002 Networks Tanenbaum Pearson Fifth
b1003 Operating system Chaudhri Jaico First
```

File after modify
b1001 Programming in java Balagurusway BPB Second
b1002 Networks Banenbaum Pearson Fifth
b1003 Operating system Chaudhri Jaico First
Books named as java
Programming in java

Books written by balagurusway are as follow:
Programming in java

Books which are not first edition:
Programming in java
Networks Tanenbaum Pearson

==========

47.Create a file “election.dat” which contains the Election details for a specific city.

Field	Description
Idno	Numeric - Unique
Name	Character – Voter’s Name
Sex	Character – M / F
Age	Numeric
Ward	Numeric – Ward no. / Division no. of the city.

Sample data:
e101-abhishek-M-35-44
e102-ashutosh-M-97-14
e103-anamika-F-21-50
Suppose the same file is to be modified after 4 years. Write a shell script to simulate this process. Your program must update the age of all People (Add 4 years to age). In case if the age exceeds 99 then delete the record from the file, assuming that the person is dead.

Display the election.dat and final Output : of your program.

==========

```
echo "File before update"
cat "election.dat"

while read line
do
        set $line
        age=$(($4 + 4))

        if [ $age -gt 99 ]
        then
                echo "$1 record deleted"
        else
                echo "$1 $2 $3 $age $5" >> "modify.dat"
        fi
done < "election.dat"

cp "modify.dat" "election.dat"
rm "modify.dat"

echo "File after update"
cat election.dat
```

Output :

File before update
e101 Raviraj M 35 44
e102 Aditya M 97 14
e103 Aditi F 21 50

e101 record deleted

File after update
e102 Aditya M 39 44
e103 Aditi F 25 50

==========

48.In a college, students are allowed to select any one elective subject during his studies. Create two files by entering the data as mentioned below (you may skip the heading line if required) :

File : elective.dat

Code	Game
101	AI
102	Computer Graphics
103	Parallel Processing
104	Data Mining

File : students.dat

RollNo.	Name	Code
1	Aditi	101
2	Raviraj	101
3	Aditya	103
4	Jaladhi	104

Write a shell script to list the students according to their choice of electives …

Eg. AI :- Aditi, Raviraj
Computer Graphics: -
Parallel Processing :- Aditya
Data Mining :- Jaladhi

==========

```
echo "Elective subjects"
cat "elective.dat"

echo "Students details:"
cat "students1.dat"

while read line
do
        set $line
        echo "$2 $3"
        c=`grep -c $1 "students1.dat"`

        if [ $c -eq 0 ]
        then
                echo "No one choosen $2 $3 as elective subject"
        else
                grep $1 "students1.dat" | cut -d " " -f2
        fi

done < "elective.dat"
```

Output :

```
Elective subjects
101 AI
102 Computer graphics
103 Parallel Processing
104 Data Mining
Students details:
1 Aditi 101
2 Raviraj 101
3 Aditya 103
4 Jaladhi 104
AI
Aditi
Raviraj
Computer graphics
No one choosen Computer graphics as elective subject
Parallel Processing
Aditya
Data Mining
Jaladhi
```

==========

49.Create two files: subjects.dat and students.dat containing the subject details and the student details. Sample data is as shown below:

subjects.dat

```
Course_id-Semester_id-Subject_id-Subject_name
CS-1-1-FCO
CS-1-2-FOP
CS-1-3-SL
CS-2-1-DS
CS-2-2-DBMS
CS-3-1-OS
CS-3-2-JAVA
```

faculty.dat

```
Faculty_id:Semester_id:Subject_id
F1-2-1
F2-3-2
F3-1-3
F1-1-1
```

Write a shell script to list the faculties and their respective subjects. Sample Output : will be :

```
F1 : FCO, DS
F2 : JAVA
F3 : SL
```

==========

```
while read line
do
        sem=`echo $line | cut -d " " -f2`
        sub=`echo $line | cut -d " " -f3`
        fac=`echo $line | cut -d " " -f1`

        while read line2
        do
                set $line2

                if [ $sem == $2 -a $sub == $3 ]
                then
                        echo "$fac teaches $4 subject"
```

```
                fi
        done < "subjects.dat"

done < "faculty.dat"
```

Output :
F1 teaches DS subject
F2 teaches JAVA subject
F3 teaches SL subject
F1 teaches FCO subject

==========

50.Create two files employee.dat and departments.dat and add atleast 10 records in the following format :

employee.dat
emp_id:department_id:birthdate
e101:M1:11-01-1960
e102:C1:21-03-1973
e103:M2:21-03-1973
e104:C1:21-03-1973
e105:B1:08-10-1965
e101:M1:11-11-1964

departments.dat
departmend_id:department_name
B1:Botany
C1:Chemistry
M1:Mathematics
M2:Management

Write a shell script to do the followings:
1)List all the employee_ids department-wise
2)List the employee_ids born after 1970
3)List the employee_ids according to birthdate in sorted order

==========

```
echo "Department wise employees"

while read line
do
        did=`echo $line | cut -d " " -f1`
        dname=`echo $line |    cut -d " " -f2`

        echo "Department $dname"
        while read line2
        do
                edid=`echo $line2 | cut -d " " -f2`
                eid=`echo $line2 | cut -d " " -f1`

                if [ $edid == $did ]
                then
                        echo $eid
                fi

        done < "employee.dat"
done < "department.dat"

echo "Employees born after year 1970 are as follow:"
```

```
while read line
do
        age=`echo $line | cut -d " " -f3 | cut -d "-" -f3`

        if [ $age -gt 1970 ]
        then
                echo $line
        fi

done < "employee.dat"

echo "Employees sorted according to their DOB"

sort -n -k 3.9 -k 3.5 -k 3 "employee.dat"
```

Output :

```
Department wise employees
Department Botany
e105
Department Chemistry
e102
e104
Department Mathematics
e101
e106
Department Management
e103
Employees born after year 1970 are as follow:
e102 C1 21-03-1973
e103 M2 21-03-1973
e104 C1 21-03-1973
Employees sorted according to their DOB
e101 M1 11-01-1960
e106 M1 11-11-1964
e105 B1 08-10-1965
e102 C1 21-03-1973
e103 M2 21-03-1973
e104 C1 21-03-1973
```

==========