Department of Computer Science

Gujarat University

Certificate

Roll No: 13 Seat No: _____

This is to certify that Mr./~~Ms.~~ Makhija Rakesh Jeramdas 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 towards partial fulfillment of his/her Degree of Masters in Computer Applications.

10/12/2020
Date of Submission

Internal Faculty

Head of Department

Department Of Computer Science
Rollwala Computer Centre
Gujarat University

MCA – 3

Subject: - Operating system

Name: - **Makhija Rakesh Jeramdas**

Roll No.: - **13** **Exam Seat No.: -**

Sr. No.	Contents	Pg. No	Date	Signature
1	Assignment-1		10/12/2020	
2	Assignment-2		10/12/2020	

Assignment :- 1

1) Base Address.
- An address that is used as the origin in the calculation of addresses in the execution of a computer program.

2) 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.

3) 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.

4) Block.
- 1) A collection of contiguous records that are recorded as a unit; the units are separated by interblock gaps.
  2) A group of bits that are transmitted as a unit.

5) 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 number of keys.

6) Busy waiting.

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

7) 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.

8) CPU. (Central Processing Unit).
- That portion of a computer that fetches and executes instructions. It consists of an Arithmetic and Logical Unit (ALU), a control unit, and registers. Often simply referred to as a processor.

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

10) concurrent.
- Pertaining to processes or threads that take place within a common interval of time during which they may have to alternately share common resources.

11) 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.

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

13) 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 impasses that occurs when multiple processes are waiting for an action by or a response from another process that is in a similar wait state.

14) 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.

15) Deadlock detection.

- A technique in which requested resources are always granted when available. Periodically, the operating system tests for deadlock.

16) Deadlock prevention.
- A technique that guarantees that a deadlock will not occur. Prevention is achieved by assuring that one of the necessary conditions for deadlock is not met.

17) Demand paging.
- The transfer of a page from secondary memory to main memory storage at the moment of need. Compare prepaging.

18) Device drivers.
- An operating system module (usually in the kernel) that deals directly with a device or I/O module.

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

20) DMA (Direct memory Access)
- A form of I/O in which a special module, called a DMA module, controls the exchange of data between main memory and an I/O device. The processor sends a request

for the transfer of a block of data to the DMA module and is interrupted only after the entire block has been transferred.

21) Disabled Interrupt.
- A condition, usually created by the operating system, during which the processor will ignore interrupt request signals of a specified class.

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

23) 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.

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

25) 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.

26) Enabled Interrupt.

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

27) External fragmentation.
- Occurs when memory is divided into variable-size partitions corresponding to the blocks of data assigned to the memory (e.g., segments in main memory). As segments are moved into and out of the memory, gaps will occur between the occupied portions of memory.

28) File.
- A set of related records treated as a unit.

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

30) FAT (File allocation Table).
- 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.

31) 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.

32) File organization.
- The physical order of records in a file, as determined by the access method used to store and retrieve them.

33) FCFS (First come First served).
- A queueing technique in which the next item to be retrieved is the item that has been in the queue for the longest time.

34) 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.

35) 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.

36) 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 retrieval.

37) Hit ratio.
- In a two-level memory, the fraction of all memory accesses that are found in the faster memory (e.g., the cache).

38) 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.

39) 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.

40) Indexed sequential Access.
- Pertaining to the organization and accessing of the records of a storage structure through an index of the keys that are stored in arbitrarily partitioned sequential files.

41) Indexed sequential File.
- A file in which records are ordered according to the values of a key field. The main file is supplemented with an index file that contains a partial list of key values; the index provides a look up capability to quickly reach the vicinity of a desired record.

42) 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.

43) Internal fragmentation.
- Occurs when memory is divided into fixed-size partitions.

(e.g., page frames in main memory, physical blocks on disk). If a block of data is 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 partition.

44) 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.

45) 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.

46) Job.
- A set of computational steps packaged to run as a unit.

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

48) Kernel 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. Also referred to as system mode or privileged mode.

49) 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.

50) Livelock.
- A condition in which two or more processes continuously change their state in response to changes in the other process(es) without doing any useful work. This is similar to deadlock in that no progress is made, but it differs in that neither process is blocked or waiting for anything.

51) 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.

52) Logical record.
- A record independent of its physical environment; portion

of one logical record may be located in different physical records or several logical records or parts of logical records may be located in one physical record.

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

54) 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 or masquerades as legitimate software. In some cases, it spreads itself to other computers via e-mail or infected disks. Types of malicious software include viruses, Trojan horses, worms, and hidden software for launching denial-of-service attacks.

55) 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.

56) Memory partitioning.
- The subdividing of storage into independent sections.

57) Microkernel.
- A small privileged operating system core that provides.

process scheduling, memory management, and communication services and relies on other processes to perform some of the functions traditionally associated with the operating system kernel.

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

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

62) Mutual exclusion.
- A condition in which there is a set of processes, only one of which is able to access a given resource or perform a given function at any time.

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

64) 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.

65) 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.

66) Page frame.
- A fixed-size contiguous block of main memory used to hold a page.

67) Paging
- The transfer of pages between main memory and secondary memory.

68) Physical Address.
- The absolute location of a unit of data in memory. (e.g. word or byte in main memory, block on secondary memory)

69) Pipe.
- A circular buffer allowing two processes to communicate

on the producer-consumers 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.

70) preemption.
- Reclaiming a resource from a process before the process has finished using it.

71) 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.

72) Process.
- A program in execution. A process is controlled and scheduled by the operating system.

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

74) Process state.
- All of the information that the operating system needs to manage a process and that the processor needs 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.

75) 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.

76) Program counter.
- Instruction address register.

77) 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.

78) Real time system.
- An operating system that must schedule and manage real-time tasks.

79) Real-time task.
- A task that is executed in connection with some process or function 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.

80) Registers
- High-speed memory internal to the CPU. some registers user visible that is, available to the programmers via the machine instruction set, other registers are used only by the CPU, for control purposes.

81) Relative address.
- An address calculated as a displacement from a base address.

82) 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.

83) 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 cannot proceed because it is waiting for some event (e.g., termination of a child process or an input/output operation) returns control to the scheduler.

84) Scheduling.
- To select jobs or tasks that are to be dispatched. In some operating systems, others units of work, such as input/output operations, may also be scheduled.

85) Secondary memory.
- Memory located outside the computers system itself; that is, it cannot be processed directly by the processor, It must first be copied into main memory. Examples include disk and tape.

86) 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.

87) Segmentation.
- The division of a program or application into segments as part of a virtual memory scheme.

88) Semaphore.
- An integer value used for signaling among processes. only three operations may be performed on a semaphore, all of which are atomic; initialize, decrement, and increment. Depending on the exact definition of the semaphore, the decrement operation may 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.

89) 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.

90) Shell.
- The portion of the operating system that interprets interactive user commands and job control language commands. It functions as an interface between the user and the operating system.

91) 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 item to be removed from the list is the item that has been in the list the shortest time. This method is characterized as last in first out.

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

93) 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).

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

95) SMP (Symmetric multiprocessing).
- A form of multiprocessing that allows the operating system to execute on any available processor or on several available processors simultaneously.

96) 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 input/output routine that receives control at a precoded location in a computer program.

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

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

99) 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 executes sequentially and is interruptible so that the processor can turn to another thread. A process may consist of multiple threads.

100) Thread switch.
- The act of switching processor control from one thread to another within the same process.

101) Time sharing.
- The concurrent use of a device by a number of users.

102) Time slice.
- The maximum amount of time that a process can execute before being interrupted.

103) 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.

104) Trojan horse.
- Secret undocumented routine embedded within a useful program. Execution of the program results in execution of the secret routine.

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

106) Virtual Address.
- The address of a storage location in virtual memory.

107) 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.

108) Virus.
- Secret undocumented routine embedded within a useful program. Execution of the program results in execution of the secret routine.

109) 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).

110) 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. Typically, if a processor has a fixed-length instruction set, then the instruction length equals the word length.

111) Worm
- Program that can travel from computer to computer across network connections. May contain a virus or bacteria.

Assignment : 2 Bankers's Algorithm.

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	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 2
P_2	3 0 2	9 0 2		6 0 0
P_3	2 1 1	2 2 2		0 1 1
P_4	0 0 2	4 3 3		4 3 1

⇒ Need ≤ work = work = work + Allocation.

P_0 343 ≤ 332 ✗ condition fails.

P_1 122 ≤ 332 · condition true.

we work + allocation.

= 3 3 2 + 2 0 0

= 5 3 2

P_2 Need ≤ work

6 0 0 ≤ 5 3 2 · condition false.

P_3 need ≤ work
011 ≤ 532 condition true
w = w + allocation
= 532 + 211
= 743

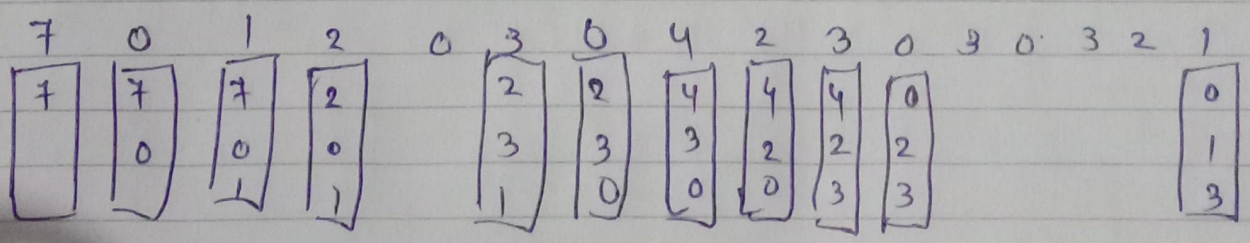
P_4 need ≤ work
431 ≤ 343
=> w = w + allocation.
= 343 + 002
= 745

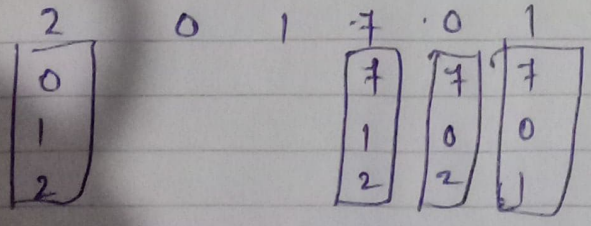
P_5 need ≤ work
343 ≤ 345
=> w = 745 + 010
= 755

P_6 = 600 ≤ 755
=> 755 + 342
= 1097

* FIFO

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

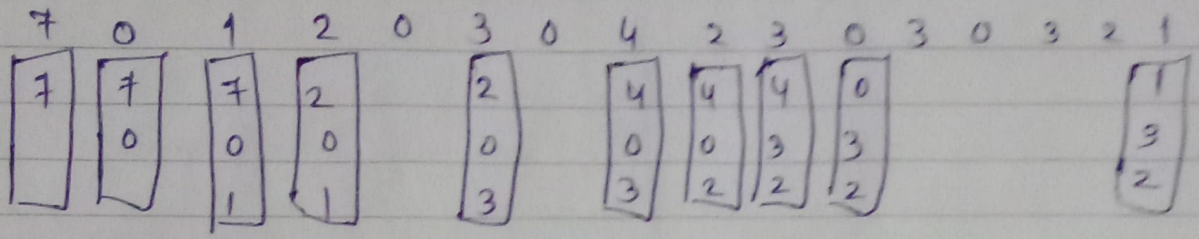


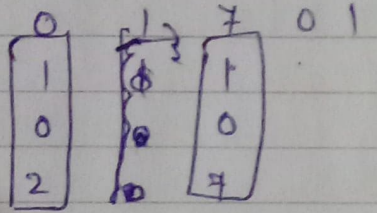


page fault = 15, no of frames = 3.

- LRU

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

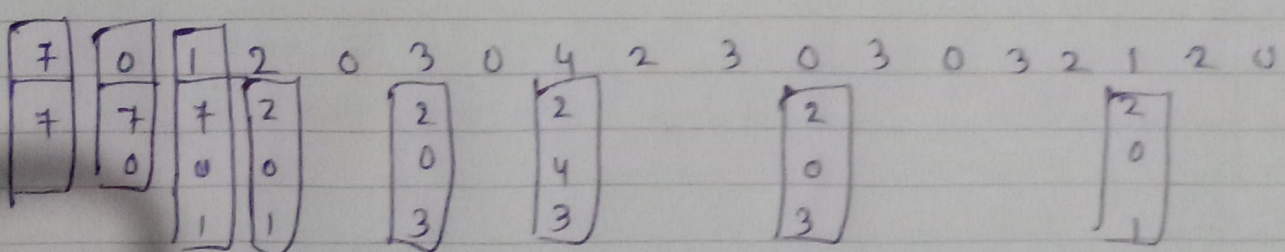


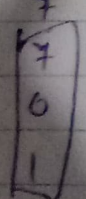


no. of frames = 3
page fault = 12.

* Optimal :-

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

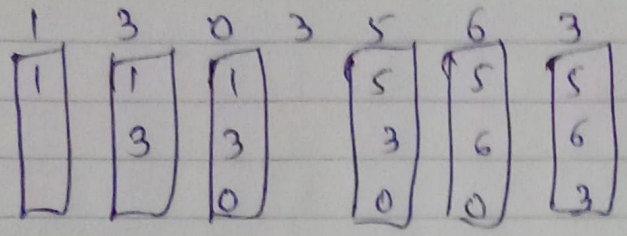




No. of frames = 3
Page fault = 9.

→ FIFO

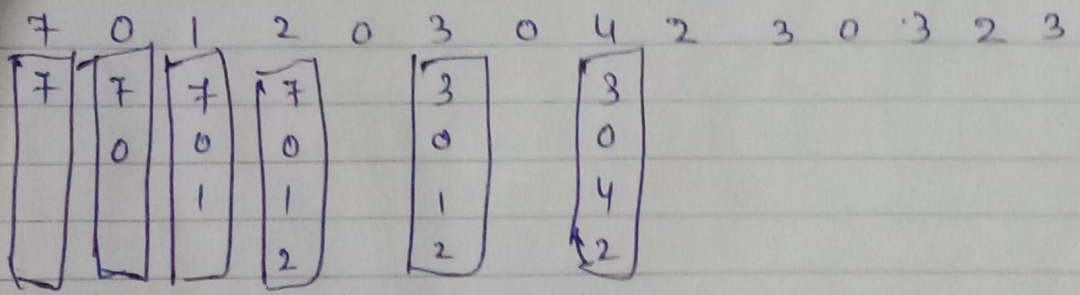
1, 3, 0, 3, 5, 6, 3



page fault = 6,
no. of frames = 3.

→ Optimal :-

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



page fault = 6,
No. of frames = 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 : 13
N A M E : Makhija Rakesh Jeramdas
S U B J E C T : Operating Sytem (OS)

NO.	TITLE	PAGE NO.	DATE	SIGN
	Assignment-1			
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. (Hint : use cut)** **eg. Logname : , UID : , GID : , Default working directory : , Default working shell :**		**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.** **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 :**		**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**		**10/12/2020**	

D E P A R T M E N T O F C O M P U T E R S C I E N C E
R O L L W A L A C O M P U T E R C E N T R E
G U J A R A T U N I V E R S I T Y
M.C.A. – 3

R O L L NO : 13
N A M E : Makhija Rakesh Jeramdas
S U B J E C T : Operating Sytem (OS)

	similar then delete the second file.			
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**	
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. (hint : use sleep)**		**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**		**10/12/2020**	

D E P A R T M E N T O F C O M P U T E R S C I E N C E
R O L L W A L A C O M P U T E R C E N T R E
G U J A R A T U N I V E R S I T Y
M.C.A. – 3

R O L L NO : 13
N A M E : Makhija Rakesh Jeramdas
S U B J E C T : Operating Sytem (OS)

	would be wordfile. The shell script should report all occurences of each word in wordfile in the rest of the files supplied as arguments			
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.**		**10/12/2020**	
35	**Print armstrog nos. from 1 to 500.**		**10/12/2020**	
36	**Accept the measure (angles) of a triangle and displa 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**	

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 : 13
N A M E : Makhija Rakesh Jeramdas
S U B J E C T : Operating Sytem (OS)

	Assignment – 2			
1	**In a college, students are allowed to select any one sporting event during his studies. Create two files as mentioned below :** **<u>File : sports.dat</u>** **Code Game** **---** **101 Cricket** **102 Football** **103 Tennis** **104 Badminton** **<u>File : students.dat</u>** **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**		**10/12/2020**	
2	**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** **<u>Sample data:</u>** **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**		**10/12/2020**	

R O L L NO : 13
N A M E : Makhija Rakesh Jeramdas
S U B J E C T : Operating Sytem (OS)

	• **Highest sold product** • **Best salesman (who sold the most)** • **Worst salesman (who sold the least)**			
3	**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.**		**10/12/2020**	
4	**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.**		**10/12/2020**	
5	**A reputed MCA institute of Gujarat has students from various states. A sample file "students.dat" is shown as**		**10/12/2020**	

	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 ……..**			
6	**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)** **<u>Sample Data:</u>** **b1001Programming in Java Balagurusway TMH Second** **b1002Computer Networks Tanenbaum Pearson Fifth**		**10/12/2020**	

D E P A R T M E N T O F C O M P U T E R S C I E N C E
R O L L W A L A C O M P U T E R C E N T R E
G U J A R A T U N I V E R S I T Y
M.C.A. – 3

R O L L NO : 13
N A M E : Makhija Rakesh Jeramdas
S U B J E C T : Operating Sytem (OS)

	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**			
7	**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.** **<u>Sample data:</u>** **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.**		**10/12/2020**	
8	**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) :** **<u>File : elective.dat</u>** **Code Game** **----------**		**10/12/2020**	

R O L L NO : 13
N A M E : Makhija Rakesh Jeramdas
S U B J E C T : Operating Sytem (OS)

	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**			
9	**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**		**10/12/2020**	

D E P A R T M E N T O F C O M P U T E R S C I E N C E
R O L L W A L A C O M P U T E R C E N T R E
G U J A R A T U N I V E R S I T Y
M.C.A. – 3

R O L L NO : 13
N A M E : Makhija Rakesh Jeramdas
S U B J E C T : Operating Sytem (OS)

	subjects. Sample Output will be : **F1 : FCO, DS** **F2 : JAVA** **F3 : SL**			
10	**Create two files employee.dat and departments.dat and add atleast 10 records in the following format :** **<u>employee.dat</u>** **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** **<u>departments.dat</u>** **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**		**10/12/2020**	

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. 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.


salary=${1}


dearnessAllowance=$(( $salary*40/100  ))


houseRent=$(( $salary*20/100  ))


total=$(( $salary+$dearnessAllowance+$houseRent ))


echo "salary: $salary"

echo "dearnessAllowance: $dearnessAllowance"

echo "houseRent: $houseRent"

echo "total: $total"
```

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.

```
# 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.
```

```
#!/bin/bash

echo "Enter Distance Between 2 cities (in km)"
read kms

kms=$(echo $kms*1.0 | bc -l);
metres=$(echo $kms*1000 | bc -l)
feet=$(echo $kms*3280.84 | bc -l);
inches=$(echo $kms*39370.1 | bc -l);
cms=$(echo $kms*100000 | bc -l);

echo "kms:" $kms
echo "metres:" $metres
echo "feet:" $feet
echo "inches:" $inches
echo "cms:" $cms
```

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.

```
# 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

#! /bin/bash

# ----- Rectangle -----
```

```
echo "Enter Length of Rectangle"

read length


echo "Enter Breadth of Rectangle"

read breadth


perimeter=$(printf '2*(%s + %s)\n' "$length" "$breadth" | bc)

echo "Perimeter of Rectangle:" $perimeter


area=$(printf '%s * %s\n' "$length" "$breadth" | bc)

echo "Area of Rectangle:" $area



# ----- Circle -----

printf "\nEnter Radius Of circle\n"

read radius


cirArea=$(printf '3.14 * %s * %s\n' "$radius" "$radius" | bc)

echo "Area of Circle:" $cirArea


circumference=$(printf '2 * 3.14 * %s\n' "$radius" | bc)

echo "Area of Circle:" $circumference
```

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

```
# 4. If a five digit number is entered through the keyboard,
```

```
# calculate the sum of its digits.


total=0


echo "Enter a number"
read num


while [ $num -gt 0 ]
do
        total=$((total+num%10))
        ((num/=10))
done


echo "Total:" $total
```

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 :

```
# 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=' | '

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 :

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.

#! /bin/bash

--

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

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

--

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

in 'awk', string Without quotes are interpreted are variables

Without Concatenation

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"}'

`|` is a string and thats why it is written in double quotes

With Concatenation

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"}'

EXPLANATION:

---------- tr ----------

`tr` stands for translate

we use `tr` command along with squeeze option (-s flag)

to convert all multiple consecutive spaces to a single space

---------- apply style on your string ----------

https://stackoverflow.com/questions/2924697/how-does-one-output-bold-text-in-bash

1. echo -e: The `-e` option means that escaped (backslashed) strings will be interpreted

2. \033: escaped sequence represents beginning/ending of the style

3. lowercase m: indicates the end of the sequence

4. 1: Bold attribute (see below for more)

5. [0m: resets all attributes, colors, formatting, etc.

0 - Normal Style

1 - Bold

2 - Dim

3 - Italic

4 - Underlined

5 - Blinking

7 - Reverse

8 – Invisible

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.

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.

```
#! /bin/bash

printf "Enter Cost price: "

read costPrice

printf "Enter Selling price: "

read sellPrice

if [ $costPrice -gt $sellPrice ]

then

        echo "Loss of -"$(($costPrice-$sellPrice))
```

```
elif [ $costPrice -lt $sellPrice ]
        then
                echo "Profit of +"$(($sellPrice-$costPrice))
        else
                echo "No profit No loss"
        fi
```

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

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

printf "Enter num: "
read num

if [ $(($num % 2)) -eq 0 ]
then
        echo "Even"
else
        echo "Odd"
fi
```

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

```
#! /bin/bash

checkPrime () {
   n=$1
```

```
if [ $n -le 1 ]
then
   return 0
fi

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

if [[ $(($n % 2)) -eq 0 || $(($n % 3)) -eq 0 ]]
then
   return 0
fi

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

```
    return 1
}



echo "Enter num"
read num

checkPrime $num
isPrime=$?

if [ $isPrime -eq 1 ]
then
    echo "Prime"
else
    echo "Composite"
fi
```

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

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



printf "Enter num: "
read num

if [ $(($num % 400)) -eq 0 ]; then
```

```
        echo "Leap"
        exit
elif [ $(($num % 100)) -eq 0 ]; then
        echo "Not Leap Year"
        exit
elif [ $(($num % 4)) -eq 0 ]; then
        echo "Leap"
        exit
else
        echo "Not Leap Year"
fi
```

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.

```
# 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.

#! /bin/bash

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
```

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.

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.

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

```
# 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
```

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

```
# 15. Write a shell script to display the appropriate message like :

# Good Morning / Good Afternoon / Good Evening


#! /bin/bash


hour=`date +%H`

echo $hour



if [[ $hour -ge 5 && $hour  -lt 12 ]]; then

        echo "Good Morning"

elif [[ $hour -ge 12 && $hour  -lt 18 ]]; then

        echo "Good Afternoon"

else

        echo "Good Evening"

fi
```

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".

```
# 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".
```

```
#! /bin/bash


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
```

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

```
# 17. Create a menu driven calculator which asks for

# two integers and perform basic arithmetic operations.
```

```
#! /bin/bash

echo "Enter Number 1"
read a

echo "Enter Number 2"
read b

#! /bin/bash

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
```

18. Find the factorial of any number.

```
# 18. Find the factorial of any number.

#!/bin/bash

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

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

echo $fact
```

19. Display the fibonacci series upto some number.

```
# 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
```

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

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

#! /bin/bash

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

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

```
# 21. Write a script which has the functionality similar
```

```
# to head and tail commands.


# This script takes

# First argument as number of lines to print

# Second argument as file name


#! /bin/bash

RED='\033[0;31m'

NC='\033[0m' # No Color


# Using Head Tail ==================================

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

printf " ---------- Using Head and Tail ----------\n"

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

head -$1 $2

echo " ..."

tail -$1 $2


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


# Count Lines

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



# Using Perl ==================================

# Incrementing by +1, because perl start to print from that lines

# So 1 line need to be reduced

let "startFromLine=startFromLine+1"

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

printf "\n ---------- Using perl ----------\n"

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

perl -ne"1..$1 and print" $2

echo " ..."

perl -ne"$startFromLine..$1 and print" $2
```

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.

```
# 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.

```
#! /bin/bash

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

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. (hint : use sleep)

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

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

```
#! /bin/bash


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


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


   if [[ $(($n % 2)) -eq 0 || $(($n % 3)) -eq 0 ]]
```

```
    then
        return 0
    fi

    i=5
    while [ $(($i*$i)) -le $n ]
    do
        if [[ $(($n % $i)) -eq 0 || $(($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 [ $isPrime -eq 1 ]
    then
        printf "$num "
    fi

    num=$((num+1))
done
```

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

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

```
#! /bin/bash

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 && $j -le $i ]]; then
                                echo $i $j $k
                        fi
                done
        done
```

done

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.

```
# 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.


#! /bin/bash


# Why you shouldn't parse the output of ls

# http://mywiki.wooledge.org/ParsingLs


echo "This script is purposely exited to accidently avoid renaming files"

echo "To rename files comment the below line"

exit


for f in *; do

        [[ -e $f ]] || continue

        mv $f $f.$$

done
```

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.

```
# 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

#! /bin/bash

if [ $# -eq 0 ]; then
   printf "Usage:\n"
   echo "./27-findWordFromFile.sh <wordFile> <findFile ...>"
   exit
fi

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:filesToRead}
```

```
    printf "Done.\n\n"
  done

done <"$1" # $1 is the file name we want to search
```

```
# GREP ----------

# -n is for line number

# Use -o, and we can only print the fetched word, instead of line
```

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

```
# 28. Write a shell script which deletes all the lines containing

# the word "unix" in the files supplied as arguments to it.

#! /bin/bash

# ========================================================================

# sed '/foo/d' deleteFromFile.txt

#  the substring foo inside the foobar string is also replaced.

# If this is not the wanted behavior, use the word-boundary expression (\b)

# at both ends of the search string.

# This ensures the partial words are not matched.

# ========================================================================
```

```
word="UNIX"


# Read all args

for i; do

   # I is for Insensitive

   # d is for delete

   # I must be written first

   sed -i "/\b$word\b/Id" $i

done
```

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.

```
# 29. The word "unix" is present in only some of the files

# supplied 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.


#! /bin/bash


for i; do

        echo "Searching file: $i ..."


   # Add -q option when you don't need the string displayed when it was found.

   if grep -q "unix" "$i"; then
```

```
        echo "Found in $i"
        exit
    fi
    echo "Done."
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.

```
# 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.

#! /bin/bash

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

# If even no of args
if [[ $(( $# % 2 )) == 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++))
    done

# if odd no of args
else
    echo "Odd no of Arguments"
    exit
fi
```

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

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

#! /bin/bash
ls -l | awk '$1 ~ /rwx/'
```

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.

```
# 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.


#! /bin/bash


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
```

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.

```
# 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.



#! /bin/bash

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
```

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

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

#! /bin/bash

echo "Enter Marks of Subject 1"
read sub1

echo "Enter Marks of Subject 2"
read sub2

echo "Enter Marks of Subject 3"
read sub3

echo "Enter Marks of Subject 4"
read sub4
```

```
echo "Enter Marks of Subject 5"
read sub5

total=$(echo $sub1 + $sub2 + $sub3 + $sub4 + $sub5 | bc)
percentage=$(echo "scale=2; $total  / 5" | bc)
echo ""
echo "Percentage:" $percentage


if (( $(echo "$percentage > 90" | bc -l) ));then
        echo "Grade A"
elif (( $(echo "$percentage > 70" | bc -l) ));then
        echo "Grade B"
elif (( $(echo "$percentage > 50" | bc -l) ));then
        echo "Grade C"
elif (( $(echo "$percentage > 30" | bc -l) ));then
        echo "Grade D"
else
        echo "Fail"
fi
```

35. Print armstrog nos. from 1 to 500.

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

checkArmstrong(){
```

```
    digits=0
    n1=$1

    n2=$n1
    n3=$n1

    while [ $n2 -gt 0 ]; do
        ((digits++))
        ((n2/=10))
    done

    total=0
    while [ $n1 -gt 0 ]; do
        rem=$((n1%10))
        power=$(echo "$rem^$digits" | bc)
        total=$(echo "$total+$power" | bc)
        ((n1/=10))
    done

    if [ $total -eq $n3 ]; then
        return 1
    else
        return 0
    fi
}
```

```
num=1
while [ $num -le 300 ]; do
   checkArmstrong $num
   ifArmstrong=$?

   if [ $ifArmstrong -eq 1 ]; then
      printf "$num "
   fi

   num=$((num+1))
done
```

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

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

read -p "Enter Angle: " angle

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

```
        echo "Obtuse Angle"
else
        echo "Incorrect Input"
fi
```

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

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

#! /bin/bash

checkDivisible(){
   n=$1
   if [ $((n % 7)) -eq 0 ]; then
      return 1
   fi
   return 0

}

num=1

while [ $num -le 100 ]
do
   checkDivisible $num
   isDivisible=$?
```

```
    if [ $isDivisible -eq 1 ]
    then
        printf "$num "
    fi

    num=$((num+1))
done
```

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

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

#! /bin/bash

read -p "Enter 3 numbers seperated by space: " n1 n2 n3

if [[ $n1 > $n2 && $n1 > $n3 ]]; then
        echo "$n1"
elif [[ $n2 > $n1 && $n2 > $n3 ]]; then
        echo "$n2"
elif [[ $n3 > $n2 && $n3 > $n1 ]]; then
        echo "$n3"
elif [[ $n3 == $n2 || $n1 == $n2 || $n1 == $n3 ]]; then
        echo "Two or more numbers are same"
fi
```

39. Find HCF and LCM of a given no.

39. Find HCF and LCM of a given numbers

```
#! /bin/bash

read -p "Enter 2 number seperated by space: " n1 n2

#finding the highest value
if [[ $n1 -eq $n2 ]];then
 echo -e "H.C.F = \e[1;34m$n1\e[0m"
 echo -e "L.C.M = \e[1;34m$n1\e[0m"
 exit 0
elif [[ $n1 -gt $n2 ]];then
 greater=$n1
 lower=$n2
else
 greater=$n2
 lower=$n1
fi

# Copying Lower and Higher for finding LCM
lowerNum=$lower
greaterNum=$greater
```

```
#finding hcf
while [ $lower -ne 0 ];do
  hcf=$lower
  lower=$((greater%lower))
  greater=$hcf
done

echo -e "H.C.F = \e[1;34m$hcf\e[0m"

# # Finding LCM
lcm=$greaterNum
while [ $((lcm % lowerNum)) -ne 0 ]; do
  ((lcm+=greaterNum))
done
echo -e "L.C.M = \e[1;34m$lcm\e[0m"
```

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

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

#! /bin/bash

# Using awk and cal
echo "Sundays in current month are:"

echo " ----- Using AWK ----- "
```

```
cal | awk 'FNR > 2{print $1}'



# Iterative

echo ""

echo " ----- Iterative -----"

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
```

SHELL PROGRAMMING ASSIGNMENT

1.

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
103	Tennis
104	Badminton

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
```

2

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
```

3.

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
```

4

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"
```

5

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"
```

6.

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
```

7

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
```

8.

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

```
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"
```

9.

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"
```

10

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"
```