Department of Computer Science

Gujarat University

Certificate

Roll No: **36** Seat No: _____

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

Date of Submission **10th-Dec-2020** Internal Faculty

Head of Department

Department Of Computer Science
Rollwala Computer Centre
Gujarat University

MCA – III

Subject: - Operating System

Name: - Preksha Sheth

Roll No.: - 36 **Exam Seat No.: -_____**

Sr. No.	Contents	Pg. No	Date	Signature
1.	Assignment -1		10/12/ 20	
	Definitions:			
	(Introduction of operating system			
	,process management,process			
	Synchronization, memory			
	Management , Storage			
	Management , File System)			
2.	Assignment-2		10/12/ 20	
	Banker's algorithm			
	FIFO algorithm			
	LRU algorithm			
	FIFO algorithm			
	OPTIMAL algorithm			

Assignment

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 Central Processing Unit (CPU) :

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

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 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.

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 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.

14 Demand paging:

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

15 Device driver:

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

16 Direct Access:

The capability to obtain data from a storage device or to enter data

into a storage device in a sequence independent of their relative position, by means of addresses that indicate the physical location of the data.

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

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

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

20 Distributed operating system:
A common operating system shared by a network of computers. The distributed operating system provides support for interprocess communication, process

migration, mutual exclusion, and the prevention or detection of deadlock.

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

22 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.

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

24, external fragmentation:
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.

25 File:
A set of related records treated as a unit.

26 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.

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

28 File Management system:
A set of system software that provides services to users and applications in the use of files, including file access, directory maintenance, and access control.

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

30 First in First out:

A queueing technique in which the next item to be retrived is the item that has been in the queue for the longest time.

31 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.

32 hasing:

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.

33 hit ratio:

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

34 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.

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

36 Index 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.

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

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

39 Internal fragmentation:

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

40 interrupt:

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

41 interrupt handler:

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

42 job:

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

43 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.

44 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.

45 Last in first Out:

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

46 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

mode, but it differs in that neither process is blocked or waiting for anything.

47 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.

48 Logical record:

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

49 Main memory malicious software:

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

50 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.

51 Memory partitioning:
The subdividing of storage into independent sections.

52 Micro kernel:
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.

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

54 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.

55 multiprogramming level:
The number of processes that are partially or fully resident in main memory.

56 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 ~~terms~~ terminology.

57 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. See critical section.

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

59 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.

60 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.

61 Page frame :

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

62 Paging :

The transfer of pages between main memory and secondary memory.

63 Physical address :

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

64 pipe :

A circular buffer allowing two processes to communicate on the producer-

consumer model. Thus, it is a first-in-first out queue, written by one process and read by another. In some systems, the pipe is generalized to allow any item in the queue to be selected for consumption.

65 Preemption:

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

66 Prepaging:

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

67 Process:

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

68 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.

69 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. Same as execution context.

70 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.

71 Program Counter:

Instruction address register.

72 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.

73 Real-time system:

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

74 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.

75 Registers:

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

76 Relative address:

An address calculated as a displacement from a base address.

77 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.

78 Round robin :

A scheduling algorithm in which processes are activated in a fixed cyclic order; that is, all processes are in a circular queue. A process that can not 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.

79 Scheduling :

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

80 Secondary memory :

Memory located outside the computer 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.

81 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.

82 Segmentation :

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

83 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.

84 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.

85 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.

86 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.

87 Starvation:

A condition in which a process is indefinitely delayed

because other processes are always given preference.

88 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)

89 Swapping:

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

90 Symmetric multiprocessing (SMP):

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

91 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.

92 Synchronization :

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

93 System bus :

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

94 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.

95 Thread switch :

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

96 Time sharing:

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

97 Time slice:

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

98 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.

99 Trojan horse:

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

100 User mode:

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

101 Virtual address:

The address of a storage location in virtual memory.

102 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.

103 Virus:

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

104 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).

105 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.

106 Thrashing :

A phenomenon in virtual memory schemes, in which the processor spends most of its time swapping pieces rather than executing instructions.

107 Time slicing :

A mode of operation in which two or more processes are assigned quanta of time on the same processor.

108 Trace :

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

109 Translation lookaside buffer (TLB):

A high-speed cache used to hold recently referenced page table entries as part of a paged virtual memory scheme. the TLB reduces the frequency of access to main memory to retrieve page table entries.

110 Trap door:

Secret undocumented entry point into a program, used to grant access without normal methods of access authentications.

111 Trusted System:

A computer and operating system that can be verified to implement a given security policy.

112 Session:

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

113 Spin lock :

Mutual exclusion mechanism in which a process executes in an infinite loop waiting for the value of a lock variable to indicate availability.

114 Spooling :

The use of secondary memory as buffer storage to reduce processing delays when transferring data between peripheral equipment and the processors of a computer.

115 Working set :

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

116 Worm :

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

Assignment 2

Banker'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_0 743 ≤ 332 ← × condition fails

P_1 122 ≤ 332 . condition trues.

W = Work + Allocation
= 332 + 200
= 532

P_2 Need $\leq$ work

$600 \leq 532$ condition false

P_3 Need $\leq$ work

$011 \leq 532$ condition true

$w = w + \text{allocation}$

$= 532 + 211$

$= 743$

$\Rightarrow P_4$ Need $\leq$ work

$431 \leq 743$

$\Rightarrow w = w + \text{allocation}$

$= 743 + 002$

$= 745$

$\Rightarrow P_0$ Need $\leq$ work

$743 \leq 745$

$\Rightarrow \text{work} = \text{work} + \text{allocation}$

$= 745 + 010$

$= 755$

$\Rightarrow P_2$ Need $\leq$ work

$600 \leq 755$

$\Rightarrow \text{work} = \text{work} + \text{alloc}$

$= 755 + 302$

$= 1057$

Safe sequence is $\Rightarrow \langle P_1, P_3, P_4, P_0, P_2 \rangle$

* FIFO

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

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

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

page fault = 15 . no of frames = 3.

* LRU

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

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

2 2 2 0 1 7 0 1

		1		1			
		0		0			
		2		7			

no of frames = 3
page fault = 12

* Optimal

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

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

2 0 1 7 0 1

7					
0					
1					

no of frames = 3
page fault = 9

=> FIFO :

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

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

page fault := 6 no of frames = 3

=> optimal

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

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

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. – III

R O L L NO : 36
N A M E : Preksha Sheth
S U B J E C T : Operating System(OS)

NO.	TITLE	PAGE NO.	DATE	SIGN
1	**Basic salary of a person is input through the keyboard. His dearness allowance is 40% of basic salary and house rent is 20% of basic salary. Write a program to calculate the gross pay.**		**10/12/2020**	
2	**The distance between two cities is input through the keyboard. (in km). Write a program to convert this distance into metres, feet, inches and centimeters and display the results.**		**10/12/2020**	
3	**The length and breadth of a rectangle and radius of a circle are entered through the keyboard, calculate the perimeter and area of rectangle and area and circumference of the circle.**		**10/12/2020**	
4	**If a five digit number is entered through the keyboard, calculate the sum of its digits.**		**10/12/2020**	
5	**The file /etc/passwd contains info about all users. Write a program which would receive the logname during execution, obtain information about it from the file and display the information on screen in some appropriate format.**		**10/12/2020**	
6	**The script will receive the filename or filename with its full path, the script should obtain information about this file as given by "ls -l" and display it in proper format.**		**10/12/2020**	
7	**If cost price and selling price of an item are entered through the keyboard, write a program to determine whether the seller has made profit or loss. Also determine how much profit/loss is made.**		**10/12/2020**	
8	**Check whether the entered no. is odd or even.**		**10/12/2020**	
9	**Check whether the entered no. is prime or not.**		**10/12/2020**	
10	**Check whether the entered year is a leap year or not.**		**10/12/2020**	
11	**The script receives two file names as arguments, the script must check whether the files are same or not, if they are similar then delete the second file**		**10/12/2020**	
12	**Write a script which will display whether your friend has logged in or not, if he has logged in then send him some message.**		**10/12/2020**	
13	**While executing a shell script, either the logname or uid is supplied at the command prompt, write a shell script to find out at how many terminals has this user logged in.**		**10/12/2020**	

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

M.C.A. – III

R O L L NO : 36
N A M E : Preksha Sheth
S U B J E C T : Operating System(OS)

NO.	TITLE	PAGE NO.	DATE	SIGN
14	Write a shell script to display the date with the format :- 25th October 2005 is a Tuesday.		10/12/2020	
15	Write a shell script to display the appropriate message like : Good Morning / Good Afternoon / Good Evening		10/12/2020	
16	Write a shell script to display the menu driven interface :- 1) list all files of the current directory 2) print the current directory 3) print the date 4) print the users otherwise display "Invalid Option".		10/12/2020	
17	Create a menu driven calculator which asks for two integers and perform basic arithmetic operations.		10/12/2020	
18	Find the factorial of any number.		10/12/2020	
19	Display the fibonacci series upto some number		10/12/2020	
20	Two numbers are entered through the keyboard, find the power, one number raised to another.		10/12/2020	
21	Write a script which has the functionality similar to head and tail commands.		10/12/2020	
22	Write a script which reports name and size of all files in a directory. whose sizes exceed 1000. The filenames should be printed in the descending order of their sizes. The total no. of files must be reported		10/12/2020	
23	A friend of yours has promised to log in a particular time. You want to contact him as soon as he logs in, write a script which checks after every minute whether the friend has logged in or not. The logname should be supplied at command prompt.		10/12/2020	
24	Print the prime nos. from 1 to 300.		10/12/2020	
25	Program must display all the combinations of 1, 2, and 3.		10/12/2020	
26	Write a script for renaming each file in the directory such that it will have the current shell PID as an extension. The shell script should ensure that the directories do not get renamed.		10/12/2020	
27	A file called wordfile consists of several words. Write a shell script which will receive a list of filenames, the first of which would be wordfile. The shell script should report all occurences of each word in wordfile in the rest of the files supplied as arguments.		10/12/2020	

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

M.C.A. – III

R O L L NO : 36
N A M E : Preksha Sheth
S U B J E C T : Operating System(OS)

NO.	TITLE	PAGE NO.	DATE	SIGN
28	**Write a shell script which deletes all the lines containing the word "unix" in the files supplied as arguments to it.**		**10/12/2020**	
29	**The word "unix" is present in only some of the files syupplied as arguments to the shell script. You script should search each of these files in turn and stop at the first file that it encounters containing the word unix. The filename should be displayed on the screen.**		**10/12/2020**	
30	**A shell script receives even number of filenames. Suppose four filenames are supplied then the first file should get copied into second file, the third file should get copied into fourth and so on.. If odd number of filenames are supplied display error message.**		**10/12/2020**	
31	**The script displays a list of all files in the current directory to which you have read, write and execute permissions.**		**10/12/2020**	
32	**The script receives any number of filenames as arguments. It should check whether every argument supplied is a file or directory. If it is a directory it should be reported. If it is a filename then name of the file as well as the number of lines present in it should be reported.**		**10/12/2020**	
33	**A script will receive any number of filenames as arguments. It should check whether such files already exist. If they do, then it should be reported, if not then check if a subdirectory "mydir" exists or not in the current directory, if it doesnt exist then it should be created and in it the files supplied as arguments should be created.**		**10/12/2020**	
34	**Accept the marks of 5 subjects and calculate the percentage and grade. 35. Print armstrog nos. from 1 to 500.**		**10/12/2020**	
35	**Print armstrog nos. from 1 to 500.**		**10/12/2020**	
36	**Accept the measure (angles) of a triangle and display the type of triangle. (eg. acute, right, obtuse)**		**10/12/2020**	
37	**Display all the numbers from 1 to 100 which are divisible by 7**		**10/12/2020**	
38	**Find the largest and smallest of 3 different numbers.**		**10/12/2020**	
39	**Find HCF and LCM of a given no.**		**10/12/2020**	
40	**Display the dates falling on Sundays of the current month.**		**10/12/2020**	

```
**********************************************************************
**********************************************************************
Name: Sheth Preksha
Class: MCA III
Rollno: 36
Subject: Operating System - OS

**********************************************************************
**********************************************************************
Q1)

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.
**********************************************************************
**********************************************************************
echo "Enter Salary : "
read salary

da=$(($salary * 40  / 100))
h_r=$(($salary * 20 / 100))
gross_pay=`expr $salary + $da + $h_r`
echo "Gross Pay Salary  = " $gross_pay

**********************************************************************
**********************************************************************
Output:
preksha@DESKTOP-A0UDC7F:~$ sh a1.sh
Enter Salary :
10000
Gross Pay Salary  =  16000
**********************************************************************
**********************************************************************
Q2)

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.

**********************************************************************
**********************************************************************
echo "Enter Distance between 2 cities :"
read dist
meter=`expr $dist \* 1000`
echo "Distance in meter = "$meter
```

```
feet=`echo "scale = 2;$dist * 3280.84"| bc`
echo "Distance in Feet = "$feet

inch=`echo "scale = 2;$dist *39370.08"| bc`
echo "Distance in Inches = "$inch

cm=`expr $dist \* 100`
echo "Distance in Centimeter = "$cm
```

****
****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a2.sh
Enter Distance between 2 cities :
125
Distance in meter = 125000
Distance in Feet = 410105.00
Distance in Inches = 4921260.00
Distance in Centimeter = 12500
```

****
****

Q3)

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.

****
****

```
echo "Enter length for rectangle :"
read length
echo "Enter breadth for rectangle :"
read breadth

area=`echo "scale = 2;$length * $breadth" | bc`
echo "Area of Rectangle = "$area
peri=`echo "scale = 2;$area * 2" | bc`
echo "Perimeter of rectangle = "$peri
echo "Enter radius for circle :"
read radius

area_c=`echo "scale = 2;3.14 * $radius * $radius" | bc`
echo "Area of circle = "$area_c

cirm=`echo "scale = 2; 2 * 3.14 * $radius" | bc`
```

```
echo "Circumference  of the circle = "$cirm
```

****
****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a3.sh
Enter length for rectangle :
2
Enter breadth for rectangle :
3
Area of Rectangle = 6
Perimeter of rectangle = 12
Enter radius for circle :
3
Area of circle = 28.26
Circumference  of the circle = 18.84
```

****
****

Q4)

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

****
****

```
echo "Enter Five digit Number : "
read no
num=$no
sum=0
while [ "$no" -gt 0 ]
do
        rem=$(($no % 10))
        sum=$(($sum + $rem))
        no=$(($no / 10))
done
echo "Sum of "$num" of digits = "$sum
```

****
****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a4.sh
Enter Five digit Number :
12345
Sum of 12345 of digits = 15
```

```
**********************************************************************
**********************************************************************
Q5)

The file /etc/passwd contains info about all users. Write a
program which would receive the logname during execution, obtain
   information about it from the file and display the
information on screen in some appropriate format. (Hint : use
cut)
   eg. Logname : UID : GID : Default working directory : Default
working shell

**********************************************************************
**********************************************************************
cut -f 1,3,4,6,7 -d":" /etc/passwd | tail -n 1

**********************************************************************
**********************************************************************
Output:

preksha@DESKTOP-A0UDC7F:~$ sh a5.sh
preksha:1000:1000:/home/preksha:/bin/bash
**********************************************************************
**********************************************************************
Q6)

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 "Enter File Name : "
read fname

ls -l $fname | cut -d ' ' -f 1,2,3,4,5,6,7,8,9

or

echo "Enter File Name : "
read fname

ls -l $fname | cut -d ' ' -f 1,2,3,4,5,6,7,8,9 |awk '{print $9
":" $1 ":" $2 ":" $3 ":" $4 ":" $5 ":" $6 ":" $7 ":" $8}'
```

```
**********************************************************************
**********************************************************************
Output:
preksha@DESKTOP-A0UDC7F:~$ sh a6.sh
Enter File Name :
a1.sh
-rw-r--r-- 1 preksha preksha 169 Nov 20 22:46 a1.sh


or


preksha@DESKTOP-A0UDC7F:~$ sh a6.sh
Enter File Name :
a1.sh
a1.sh:-rw-r--r--:1:preksha:preksha:169:Nov:20:22:46
**********************************************************************
**********************************************************************
Q7)

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.

**********************************************************************
**********************************************************************
echo "Enter Cost price : "
read cp
echo "Enter Selling price : "
read sp

if [ $sp -gt $cp ]
then
        echo "The Seller Made profit of "`expr $sp - $cp`
else
        echo "The seller made loss of "`expr $cp - $sp`
fi

**********************************************************************
**********************************************************************
Output:

preksha@DESKTOP-A0UDC7F:~$ sh a7.sh
```

```
Enter Cost price :
100
Enter Selling price :
120
The Seller Made profit of 20


preksha@DESKTOP-A0UDC7F:~$ sh a7.sh
Enter Cost price :
120
Enter Selling price :
100
The seller made loss of 20
```

****

****

Q8)

Check whether the entered no. is odd or even.

****

****

```
echo "Enter Number : "
read no

rem=`expr $no % 2`
if [ $rem -eq 0 ]
then
        echo "$no is even number.."
else
        echo "$no is odd number.."
fi
```

****

****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a8.sh
Enter Number :
12
12 is even number..

preksha@DESKTOP-A0UDC7F:~$ sh a8.sh
Enter Number :
11
11 is odd number..
```

****

****

Q9)

Check whether the entered no. is prime or not.

```
**************************************************************
**************************************************************
echo "Enter Number : "
read no

flag=1
i=2

while [ $i -lt $no ]
do
        rem=`expr $no % $i`
        #echo "$rem is..."
        if [ $rem -eq  0 ]
        then
                flag=0
                break
        fi
        i=`expr $i + 1`
done

if [ $flag -eq 1 ]
then
        echo "$no is prime Number."
else
        echo "$no is not prime Number."
fi
**************************************************************
**************************************************************
```

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a9.sh
Enter Number :
6
6 is not prime Number.


preksha@DESKTOP-A0UDC7F:~$ sh a9.sh
Enter Number :
5
5 is prime Number.
**************************************************************
**************************************************************
```

Q10)

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

```
**************************************************************
**************************************************************
echo "Enter Year : "
read yr

if [ `expr $yr % 4` -eq 0 -a `expr $yr % 100` -ne 0 -o `expr $yr
% 400` -eq 0 ]
then
        echo "$yr is leap year."
else
        echo "$yr is not leap year."
fi

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

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a10.sh
Enter Year :
2019
2019 is not leap year.


preksha@DESKTOP-A0UDC7F:~$ sh a10.sh
Enter Year :
2020
2020 is leap year.
**************************************************************
**************************************************************
```

Q11)

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.

```
**************************************************************
**************************************************************
cmp -s $1 $2`
if [ $? -eq 0 ]
then
        echo "$1 and $2 are same files"
        rm $2
        echo "$2 file deleted"
```

```
else
        echo "$1 and $2 are not same files"
fi
```

****
****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a11.sh demo2.txt demotest.txt
demo2.txt and demotest.txt are not same files
```

****
****

Q12)

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 uname
who | grep $uname > /dev/null

if [ $? -eq 0 ]
then
        echo "User is Logged in.."
        echo "Please enter message :"
        read msg
        echo $msg
else
        echo "User is not logged in .."
fi
```

****
****

Output:

```
[ec2-user@ip-172-31-93-145 ~]$ sh a12.sh
Enter User Name :
preksha
User is Logged in..
Please enter message :
hii
hii
```

```
**************************************************************
**************************************************************
```
Q13)

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.

```
**************************************************************
**************************************************************
if [ $# -eq 1 ]
then
          total=`who | grep -c $1`
          echo "$1 logged in on total $total terminals"
else
          echo "please enter user name..."
fi
**************************************************************
**************************************************************
```
Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a13.sh
please enter user name...
preksha@DESKTOP-A0UDC7F:~$ sh a13.sh preksha
preksha logged in on total 1 terminals
**************************************************************
**************************************************************
```
Q14)

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

```
**************************************************************
**************************************************************
date +"%dth %B %Y is a %A"
**************************************************************
**************************************************************
```
Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a14.sh
08th December 2020 is a Tuesday

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

Q15)

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

```
**************************************************************
**************************************************************
time=`date '+%H'`
echo "$time"

if [ $time -ge 6 -a $time -lt 12 ]
then
        echo "Good Morning.."
elif [ $time -ge 12 -a $time -lt 16 ]
then
        echo "Good Afternoon.."
elif [ $time -ge 16 -a $time -lt 20 ]
then
        echo "Good Evening.."
else
        echo "Good Night.."
fi
**************************************************************
**************************************************************
```

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a15.sh
16
Good Evening..
**************************************************************
**************************************************************
```

Q16)

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".

```
**************************************************************
**************************************************************
while [ 1 ]
do
        echo -e "\n\n1.List of all the current
directory\n2.Print the current directory\n3.print the date
\n4.print the users\n0.exit"

        echo "Enter your  choice :"
```

```
        read ch

        case $ch in
                "1") ls ;;
                "2") pwd ;;
                "3") date ;;
                "4") who ;; # awk -F: '{print $1}' /etc/passwd
                "0") echo "Exit"
                     break ;;
                 *) echo "Invalid choice.."
        esac
done
```

****
****

Output:

```
[ec2-user@ip-172-31-93-145 ~]$ sh a16.sh


1.List of all the current directory
2.Print the current directory
3.print the date
4.print the users
0.exit
Enter your  choice :1
a16.sh  emp.dat  first.sh  mca06  sample.dat  sample.sh
second.sh  stud.dat  third.sh


1.List of all the current directory
2.Print the current directory
3.print the date
4.print the users
0.exit
Enter your  choice :2
/home/ec2-user


1.List of all the current directory
2.Print the current directory
3.print the date
4.print the users
0.exit
Enter your  choice :3
Sun Nov 22 08:12:42 UTC 2020
```

```
1.List of all the current directory
2.Print the current directory
3.print the date
4.print the users
0.exit
Enter your  choice :4
ec2-user pts/0        2020-11-22 07:55 (43.241.144.141)


1.List of all the current directory
2.Print the current directory
3.print the date
4.print the users
0.exit
Enter your  choice :5
Invalid choice..


1.List of all the current directory
2.Print the current directory
3.print the date
4.print the users
0.exit
Enter your  choice :0
Exit
```

****
****

Q17)

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

****
****

```
echo "Enter !st Integer :"
read no1
echo "Enter 2nd Integer :"
read no2

while [ 1 ]
do
        echo
"\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Division\n0.Ex
it\n"
        echo "Enter Your Choice :"
```

```
        read ch

        case $ch in
                "1") ans=`expr $no1 + $no2`
                        echo "Addition = " $ans ;;
                "2") ans=`expr $no1 - $no2`
                        echo "Subtraction = " $ans ;;
                "3") ans=`expr $no1 \* $no2`
                        echo "Multiplication = " $ans ;;
                "4") ans=`expr $no1 \/ $no2`
                        echo "Division = " $ans ;;
                "0") echo "Exit.."
                        break ;;
                *) echo "Invalid Choice.."
        esac
done
```

****

****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a17.sh
Enter !st Integer :
40
Enter 2nd Integer :
20

1.Addition
2.Subtraction
3.Multiplication
4.Division
0.Exit

Enter Your Choice :
1
Addition =  60

1.Addition
2.Subtraction
3.Multiplication
4.Division
0.Exit

Enter Your Choice :
3
Multiplication =  800

1.Addition
```

```
2.Subtraction
3.Multiplication
4.Division
0.Exit

Enter Your Choice :
4
Division =  2

1.Addition
2.Subtraction
3.Multiplication
4.Division
0.Exit

Enter Your Choice :
2
Subtraction =  20

1.Addition
2.Subtraction
3.Multiplication
4.Division
0.Exit

Enter Your Choice :
u
Invalid Choice..

1.Addition
2.Subtraction
3.Multiplication
4.Division
0.Exit

Enter Your Choice :
5
Invalid Choice..

1.Addition
2.Subtraction
3.Multiplication
4.Division
0.Exit

Enter Your Choice :
0
```

```
Exit..
**********************************************************************
**********************************************************************
```
Q18)

Find the factorial of any number.

```
**********************************************************************
**********************************************************************
echo "Enter Number : "
read no

fact=1

while [ $no -gt 1 ]
do
        fact=$(($fact * $no))
        no=$(($no - 1))
done
echo "Factorial = $fact"


or


echo "Enter Number : "
read no
num=$no
fact=1

while [ $no -gt 1 ]
do
        fact=`expr $fact \* $no`
        no=$(($no - 1))
        #echo "$no+1! is $fact.."
done
echo "Factorial = $fact"
~

**********************************************************************
**********************************************************************
Output:

preksha@DESKTOP-A0UDC7F:~$ sh a18.sh
Enter Number :
5
Factorial = 120
```

```
**********************************************************************
**********************************************************************
```
Q19)

Display the fibonacci series upto some number.

```
**********************************************************************
**********************************************************************
echo "Enter number :"
read num

n1=0
n2=1
i=2

echo "Fibonacci Series : "
echo "$n1\n$n2"

while [ $i -lt $num ]
do
        i=`expr $i + 1`
        n3=`expr $n1 + $n2`
        echo $n3
        n1=$n2
        n2=$n3
done

**********************************************************************
**********************************************************************
```
Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a19.sh
Enter number :
5
Fibonacci Series :
0
1
1
2
3
**********************************************************************
**********************************************************************
```
Q20)

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

```
**********************************************************************
**********************************************************************
echo "Enter Power : "
read pow
echo "Enter Exponent : "
read exp

i=0
ans=1

while [ $i -lt $pow ]
do
        ans=`expr $ans \* $exp`
        i=`expr $i + 1`
done


echo "$exp ^ $pow = $ans"
~
**********************************************************************
**********************************************************************

Output:

preksha@DESKTOP-A0UDC7F:~$ sh a20.sh
Enter Power :
2
Enter Exponent :
5
5 ^ 2 = 25
**********************************************************************
**********************************************************************

Q22)
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:

```
preksha@DESKTOP-A0UDC7F:~$ sh a22.sh
4096 mydir
1496 a16.sh
1392 a17.sh
1059 MCA3_36_Preksha.sh
```

****

****

Q24)

Print the prime nos. from 1 to 300.

****

****

```
        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)) -eq 0 ]
        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:

```
preksha@DESKTOP-A0UDC7F:~$ sh a24.sh
2
3
5
7
11
13
17
19
23
25
29
31
35
37
41
43
47
49
53
55
59
61
65
67
71
73
77
```

79
83
85
89
91
95
97
101
103
107
109
113
115
119
121
125
127
131
133
137
139
143
145
149
151
155
157
161
163
167
169
173
175
179
181
185
187
191
193
197
199
203
205
209
211
215
217

```
221
223
227
229
233
235
239
241
245
247
251
253
257
259
263
265
269
271
275
277
281
283
287
289
293
295
299
```

****
****

Q25)

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
                        echo "$i $j $k"
                done
        done
done
```

****
****
Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a25.sh
1 1 1
1 1 2
1 1 3
1 2 1
1 2 2
1 2 3
1 3 1
1 3 2
1 3 3
2 1 1
2 1 2
2 1 3
2 2 1
2 2 2
2 2 3
2 3 1
2 3 2
2 3 3
3 1 1
3 1 2
3 1 3
3 2 1
3 2 2
3 2 3
3 3 1
3 3 2
3 3 3
```

****
****
Q26)
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:

```
preksha@DESKTOP-A0UDC7F:~$ ls -l
total 0
-rwxrwxrwx 1 neel neel 13 Dec  9 16:38 Hello.sh.82
-rwxrwxrwx 1 neel neel 58 Dec  9 16:38 file1.txt.82
-rwxrwxrwx 1 neel neel  0 Dec  9 14:42 neel.sh.82
-rwxrwxrwx 1 neel neel  0 Dec  9 16:43 s_s_26.sh
-rwxrwxrwx 1 neel neel 55 Dec  9 16:39 s_s_26.sh.82
```

****

****

Q27)
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.

****

****

```
 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
```

****

****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a27.sh wordFile.txt findFile.txt
1
Searching word: 'samsung' ...
~
```

****

****

Q28)
Write a shell script which deletes all the lines containing the word "unix"
in the files   supplied as arguments to it.

****

****

```
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" $
done
```

****

****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ cat unix.txt
unix
helloi
i love unix
linux is best

preksha@DESKTOP-A0UDC7F:~$ sh a28.sh unix.txt

Output :

preksha@DESKTOP-A0UDC7F:~$ cat unix.txt
hello
linux is best
```

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

Q29)
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:

preksha@DESKTOP-A0UDC7F:~$ sh a29.sh temp.txt
Searching file : temp.txt...
Found in temp.txt

preksha@DESKTOP-A0UDC7F:~$ cat temp.txt
hello
unix
how r u
unix

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

Q30)
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:

preksha@DESKTOP-A0UDC7F:~$ cat temp.txt
hello
unix
how r u
```

```
unix

preksha@DESKTOP-A0UDC7F:~$ cat newdemo.txt
prekshasheth
fgdffgef
fdgfdvd
preksha@DESKTOP-A0UDC7F:~$ cat demo.txt
preksha/sheth
fgdf/fgef
fdgf/dvd
preksha@DESKTOP-A0UDC7F:~$ cat demotest.txt
preksha/sheth
fgdf/fgef
fdgf/dvd
preksha@DESKTOP-A0UDC7F:~$ cat demo2.txt
```

****
****

Q31)
The script displays a list of all files in the current directory to which
you have read,   write and execute permissions.

****
****

```
echo "The list of File Names in the current Directory"
echo "Which have read,write and execute permission"

for file in *
do
        if [ -f $file ]
        then
               if [ -r $file -a -w $file -a -x $file ]
               then
                       ls $file
                fi
        fi
done
```

****
****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a31.sh
The list of File Names in the current Directory
Which have read,write and execute permission
```

a1.sh

****

****

Q32)

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:

```
preksha@DESKTOP-A0UDC7F:~$ sh a32.sh demo.txt
demo.txt -> file with lines: 3
```

****

****

Q33)

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 "directory exists.."
                else
                        mkdir mydir
                fi
                touch mydir/$i
                echo "$i file created in \"mydir\""
        fi
done
```

****
****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a33.sh preksha
directory exists..preksha file created in "mydir"
```

****
****

Q34)
Accept the marks of 5 subjects and calculate the percentage and grade.

****
****

```
echo "Enter Marks for subject 1 : "
read s1
echo "Enter Marks for Subject 2 : "
read s2
echo "Enter Marks for Subject 3 : "
read s3
echo "Enter Marks for Subject 4 : "
read s4
echo "Enter Marks for Subject 5 : "
read s5

total=`expr $s1 + $s2 + $s3 + $s4 + $s5`
```

```
echo "Total = $total"

per=$(($total * 100 / 500 ))
echo "Percentage = $per"

if [ $per -gt 80 ]
then
        echo "Grade is A.."
elif [ $per -lt 80 -a $per -gt 60 ]
then
        echo "Grade is B.."
elif [ $per -lt 60 -a $per -gt 40 ]
then
        echo "Grade is c.."
else
        echo "Fail.."
fi
```

****

****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a34.sh
Enter Marks for subject 1 :
80
Enter Marks for Subject 2 :
89
Enter Marks for Subject 3 :
87
Enter Marks for Subject 4 :
78
Enter Marks for Subject 5 :
80
Total = 414
Percentage = 82
Grade is A.
```

****

****

Q35)
Print armstrog nos. from 1 to 500.

****

****

```
i=1
while [ $i -lt 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=`expr $i + 1`
done
```

****
****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a35.sh
Armstrong number :  1
Armstrong number :  153
Armstrong number :  370
Armstrong number :  371
Armstrong number :  407
```

****
****

Q36)
Accept the measure (angles) of a triangle and displa the type of triangle.
(eg. acute, right,obtuse)

****
****

```
echo "Enter angle "
read angle

if [ $angle -ge 0 -a $angle -lt 90 ]
then
        echo "Acute angle"
elif [ $angle -eq 90 ]
then
        echo "Right angle "
elif [ $angle -gt 90 -a $angle -le 180 ]
then
```

```
        echo "Obtuse angle "
else
        echo "Incorrect input"
fi
```

****

****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a36.sh
Enter angle
120
Obtuse angle
```

****

****

Q37)

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

****

****

```
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
```

****

****

Output:

```
preksha@DESKTOP-A0UDC7F:~$ sh a37.sh
7 14 21 28 35 42 49 56 63 70 77 84 91 98
```

```
**********************************************************************
**********************************************************************
Q38)
Find the largest and smallest of 3 different numbers.

**********************************************************************
**********************************************************************
echo "Enter 1st Number :"
read no1
echo "Enter 2nd Number :"
read no2
echo "Enter 3rd Number :"
read no3

if [ $no1 -gt $no2 -a $no1 -gt $no3 ]
then
        echo "$no1 is Largest.."
elif [ $no2 -gt $no1 -a $no2 -gt $no3 ]
then
        echo "$no2 is Largest.."
else
        echo "$no3 is Largest.."
fi


if [ $no1 -lt $no2 -a $no1 -lt $no3 ]
then
        echo "$no1 is Smallest.."
elif [ $no2 -lt $no1 -a $no2 -lt $no3 ]
then
        echo "$no2 is Smallest.."
else
        echo "$no3 is Smallest..."
fi

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

Output:

preksha@DESKTOP-A0UDC7F:~$ sh a38.sh
Enter 1st Number :
3
Enter 2nd Number :
1
Enter 3rd Number :
2
```

```
3 is Largest..
1 is Smallest..
**********************************************************************
**********************************************************************

Q39)
Find HCF and LCM of a given no.
**********************************************************************
**********************************************************************
echo -n "Enter first number : "
read num1
echo -n "Enter second number : "
read num2

max=$num1
den=$num2

if [ $num2 -gt $max ]
then
        max=$num2
        den=$num1
fi

        rem=$((max % den))

while [ $rem -ne 0 ]
do
        max=$den
        den=$rem
        rem=$((max % den))
        max=$((max - 1))
done

        gcd=$den
        lcm=`expr $num1 \* $num2 / $gcd`

echo "HCF of $num1 and $num2 = $gcd"
echo "LCM of $num1 and $num2 = $lcm"

**********************************************************************
**********************************************************************
Output:

preksha@DESKTOP-A0UDC7F:~$ sh a39.sh
Enter first number : 15
Enter second number : 55
HCF of 15 and 55 = 5
```

```
LCM of 15 and 55 = 165
**********************************************************************
**********************************************************************
Q40)
Display the dates falling on Sundays of the current month.
**********************************************************************
**********************************************************************
echo "Sundays in current month are:"

echo " ----- Using AWK ----- "
cal | awk 'FNR > 2{print $1}'


#Sundays in current month are:
 #----- Using AWK -----
 #1
 #6
 #13
 #20
 #27

**********************************************************************
**********************************************************************
Output:

Sundays in current month are:
----- Using AWK -----
1
6
13
20
27
**********************************************************************
**********************************************************************
```