Q 1 :-

In dead lock prevention strgtegy do you think it is necessary to check that either safe state exists or not ?

A state is safe if the system can allocate all the sources requested by all processes up to their stated maximum without entering a deadlock state if a safe sequences does not exist, then the system is an unsafe state, which may lead to deadlock. all safe states are deadlock free but not all unsafe states lead to deadlocks.

Q2 Dynamic loading & dynamic linking with the help of example?

The memory mangment of operationg system is to increase the degree of mutiprograming is based on the following concepts in which two are explained.

Dynamic loading :-

to increase the degree of mutiprograming. we just laod the main pant of the process to main memory then the other pants loaded only if it is called or executed. That's why for each process will be required only less memory space within the main memory, hence we will be able to load more numbers of processes to the main memory which increase the degree of multiprograming Hence the CPU utilization will also be increased.

Example :-

We write a program in C language and when we execute the same program so here how dynamic loading works.

```
main ()
    if ( )
    move ()

    else
    rain ()
```

First loader will only load main function to the main memory. later on if the if statement is true, it will call the move () function or when the if statement is false it will call the run () function.

As we noticed that using dynamic loading we save the space in the main memory instead of loading the whole program to the main memory. Hence the

Dynamic linking :-

Every dynamically linked program contains a small, statically linked function that is called when the program starts. This function only maps the links library into memory and runs the code that the function contains. The link library determines what are all the dynamic libraries which the program requires along with the names of the variables and functions needed from those libraries by reading the information contained in sections of the library.

After which it maps the libraries into the middle of virtual memory and resolves the refrences to the symbols contained in those libraries we don't know where in the memory these shared libraries are actually mapped. They are compiled into position independant code (PIC) that can at any address in memory.

Q3 Which component of an operating system is best suited to ensure fair, secure orderly and effitient use of menory?

The most suitable component of an os that provide suited security, ensure fair use of memory is memory mangrment.

It is done through the tracks keeping of used and free memory space as well as when, where how memory to allocate and de-allocate it.

It is also responsible for Swaping processes in and out of main memory.

Q4 Differentiate b/w Symatric & Asymatric encryption with the help of Example?

Symatric Encryption :-

is a type of Encryption in which we share the information with the same key, which means we use the same key to encrypt the information as well as decrypt the information it was good method at first time to Encrypt the data still it is widely so used but there is issue Symatric encryption while sharing the same key by encrypting and decrypting because this key might be stolen & hacked during sharing through network.

In order to avoid this problem / issue we use another type of encryption that's called A Symatric encryption

Symatric encryption is also convential encryption and Secret key encryption.

Examples :-
practical example is seinding verification to the mobile to access the data.

Examples :-
Serpent, DES

also our whatsapp data use encryption method.

A Symatric Ecryption is an Encryption in which we use two different keys.

public key :-
to encrypt the information

private key :-
to decrypt the information.

If the public key is visible to the other people but still can not decrypt the information It is more safe encryption method as compared to symatric

Examples :-
Algorithms
Diffie-Hellmen, ELgamel.

Differences b/w Symatric & Asymatric encryption.

	Symatric	Asymatric
(1)	It is fast way of encryption.	public key can shared.
(2)	Hard to transport shared key	slow way of encryption.
(3)	Effitient for large data	Designed for small data ineffitient for large data.

Q5 Difference b/w External & internal fragmentation, explain why they should be avoided?

Differences

	External fragmentation	External Fragmentation.
(1)	In internal fragmentation fixed sized memory, blocks, square measure appointed to process.	In external fragmentation variable sized memory blocks, square measure appointed method.
(2)	It happens when the method or process is larger than the memory.	It happens when method or process is removed.
(3)	The difference b/w memory allocated & required space in memory is called internal fragmentation	The spare spaces formed b/w contigous memory fragments are too small to serve a new process is known as external fragmentation.
(4)	it's solution is best fit block	it's solution is compaction pagging & segmentation.

Checked By: Remark: Excellent Good

(5)	It occurs when memory is divided into fixed sized partitions.	It occurs when memory is divided into variable sized partitions base on the size of process.

Q5 part B :-

External Fragmentation is the breaking up of free memory into small chunks via partitioning which can mean a request for larger partition later may fail due to lack of contigous memory, even though enough total memory is available External fragmentation can be reduced by compaction, or shuffle memory contents to place all free memory together in one large block. External fragmentation is also avoided by using paging techniques.

internal fragmentation occurs when the memory is divided into fixed sized blocks whenever a process request for the memory, the fixed sized block is allocated to the process is somewhat larger than the memory requested then the difference b/w assigned and requested memory is the internal fragmentation. This left over space inside the fixed sized block can not be allocated to any process as it would not be suffitient to satisfy the request of memory by the process, internal fragmentation can be reduced by effectively assiging the smallest partition but large enough for the process.

Q6 List and describle the four memory allocation algorithms covered in lectures which two of the four are more commonly used in practice?

The following are four memory allocation algorithms.

(1) First fit :-
allocate into the first available gap found in adequate size, starting from begining of memory.
Scan memory region list from start, for first fit must always slip over potentially many regions at the start of list

(2) Next fit :-
Allocate into the first gap found reasoning the search from the last allocation
Scan memory region list from point of last allocation to next fit breaks up large block at the end of memory.

(3) Best fit :-
picks the closet free region in the entire list leaves small unusable regions and slower due to searching of entire list.

(4) Worst-fit :-
finds the worst fit in the entire list slower as it searches the entire list fragmentation is still an issue.

Q7 Why is the context switch overhead of a user-level threading as compared to the overhead for process? Explain?

User level threads implements in user level libraries, rather than via system calls, so thread switching does not need to call the operating system and to cause interruption to the kernal. In fact, the kernal knows nothing about user-level threads and manages them as if they were single threaded processes.
Threads are very inexpensive to create and destroy and they are inexpensive to represent.

Example: They require space to store
The PC, the SP and the general purpose registers, but they don't require to space

share memory information.

The most obvious advantage of this technique is that a user level threads package can be implemented on an operating system that does not support threads.

(1) User level threads do not require modification to operating systems.

(2) Simple Representation: Each thread is represented simply by a PC, registers, stack and small control bank, all stored in the user process address space.

(3) Simple mangment, This simply means that creating a thread switching between threads and Synchronization between threads can all be done without

intervention of the kernal.

(4) Fast and Effitient:-

Thread switching is not much more expensive than an procedure call.