Define the Term "Overlay", with an example.

Overlays refer to a technique used to manage memory efficiently by overlaying a portion of memory with another program or data.

Example –
The best example of overlays is assembler. Consider the assembler has 2 passes, 2 pass means at any time it will be doing only one thing, either the 1st pass or the 2nd pass. This means it will finish 1st pass first and then 2nd pass. Let assume that available main memory size is 150KB and total code size is 200KB

Pass 1.....70KB

Pass 2.....80KB

Symbol table.....30KB

Common routine.....20KB

As the total code size is 200KB and main memory size is 150KB, it is not possible to use 2 passes together. So, in this case, we should go with the overlays technique.

According to the overlays concept at any time only one pass will be used and both the passes always need symbol table and common routine. Now the question is if overlays-driver* is 10KB, then what is the minimum partition size required? For pass 1 total memory needed is = (70KB + 30KB + 20KB + 10KB) = 130KB and for pass 2 total memory needed is = (80KB + 30KB + 20KB + 10KB) = 140KB.So if we have minimum 140KB size partition then we can run this code very easily.

List the differences between Deadlock and Starvation with suitable example.

Features	Deadlock	Starvation
Definition	Deadlock happens when every process holds a resource and waits for another process to hold another resource.	Starvation happens when a low priority program requests a system resource but cannot run because a higher priority program has been employing that resource for a long time.
Basic	A deadlock occurs when no process can proceed and becomes blocked.	Starvation occurs when low priority procedures are blocked while high priority operations proceed.
Other names	Deadlock is also known as circular wait.	Starvation is known as a Lived lock.
Resources	Other processes block requested resources while a process is deadlocked.	High-priority processes continue to use the requested resources.
Arising Condition	Mutual exclusion's occurrence, Hold and wait, No preemption, and Circular wait all happen simultaneously.	Uncontrolled resource management, enforcement of priorities.

Prevention	It can be prevented by avoiding the situations that lead to deadlock.	Aging may prevent it.

Explain Critical Section problem and its solution using Dekker's Algorithm.

Critical Section is the part of a program which tries to access shared resources. The critical section cannot be executed by more than one process at the same time. The critical section problem is used to design a set of protocols which can ensure that the Race condition among the processes will never arise.

Dekker's Algorithm:

The two processes, P_0 and P_1, share the following variables:

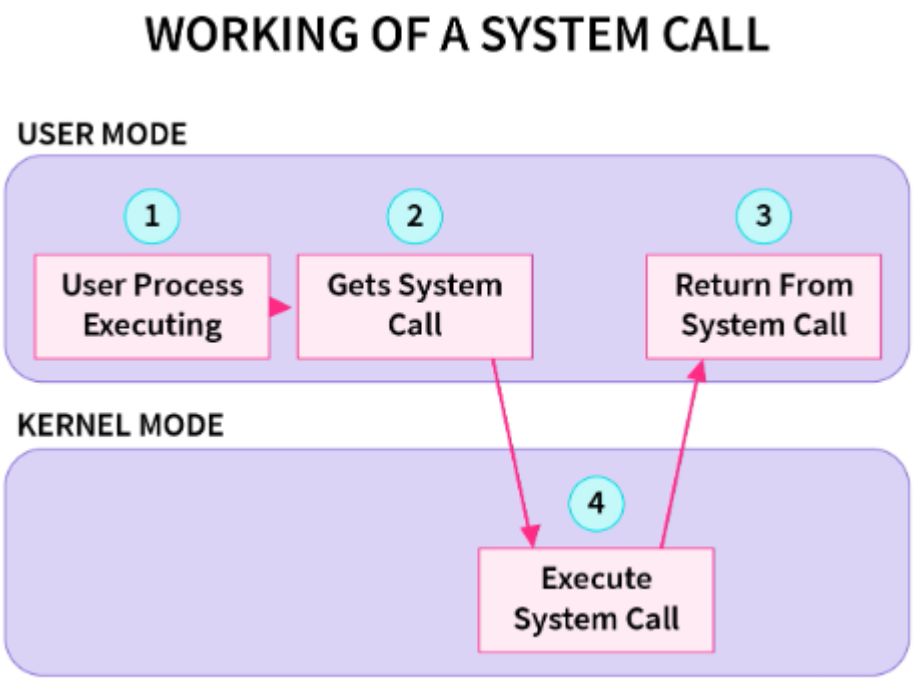
```
boolean flag[2]; /* initially false */
int turn;
```

```
do {
   flag[i] = TRUE;

   while (flag[j]) {
      if (turn == j) {
         flag[i] = false;
         while (turn == j)
            ; // do nothing
         flag[i] = TRUE;
      }
   }

      // critical section

   turn = j;
   flag[i] = FALSE;

      // remainder section
} while (TRUE);
```

Figure 6.25 The structure of process P_i in Dekker's algorithm.

The structure of process P_i (i == 0 or 1) is shown above; the other process is P_j (j == 1 or 0).

Explain the system call execution process flow chart in detail.



In our computer system, we have two modes available.

1. **User Mode:** In this mode, execution is done on behalf of the user.
2. **Monitor/Kernel-Mode:** In this mode, execution is done on behalf of OS.

System Call Execution Process:

- So when the process is under execution, process executes under user mode, but when the process requires some OS resources to complete the task, the process makes a system call.
- System calls in OS are executed in kernel mode on a priority basis.
- On completion of the system call, the control is passed to process in user mode.
- The process continues the execution in user mode.