1 - B) Multi threading

In main() we have declared a variable called thread_id, which is of type pthread_t, which is an integer used to identify the thread in the system. After declaring 3 thread_id, we have called pthread_create() function to create three threads.

In between we have called mutex to synchronize all the threads

In an Operating System, we have a number of processes and these processes require a number of resources. In this situation 3 threads are using the same variable "num". They are reading the variable and then updating the value to the variable and finally writing the data in the memory.

In the above, you can see that a process after doing some operations will have to read the value of num, then increment the value of num by 1 and at last write the value of num in the memory. Now, we have three threads that needs to be executed.

Now to Synchronization all three threads we have used mutex lock in task function by using that we have confirmed that not more than one thread is using the value of num togather and then we have printed the num’s ascii value and integer value . Then we unlocked the mutex so that other thread can make chnages too.

2 - B)

Buddy memory allocation is a memory allocation strategy that divides memory into divisions and tries to fulfil a memory request.
This technique satisfies criteria while utilising memory by splitting it into halves.

Buddy memory allocation is simple to implement and has fewer hardware requirements than recent memory allocation algorithms.
In addition, compared to other memory allocation strategies such as dynamic allocation, it has no internal fragmentation and has a low overhead when it comes to memory compaction.

Memory is wasted in buddy memory allocation because the memory requested is somewhat greater than a small block but significantly smaller than a large block.
(For example, if a software seeks 65 kilobytes of memory, the memory allocated is 128 kilobytes, wasting 63 kilobytes of RAM).

Memory is allocated in powers of two, i.e. 2x, where x is an integer, using the buddy memory allocation mechanism.
When a request for memory allocation is received, we check for the smallest block larger than it on the remaining free list, and if one is located, allocation is completed.
If the bigger block isn't located, we move up the free list until we find it.
We keep dividing it into two sections.
One is for adding the next free list, while the other is for traversing down the tree until we reach the destination and returning the request memory block to the user.
We return null if no allocation is possible.