1. Copy-on-write: I understand that during a fork operation the parent's page frames are assigned to the child and made read-only to both parent and child. Later when one of them wants to change something then a new frame area gets allocated with read-write access to both parent & child. My question is why is the OS giving a new frame area to the parent with read-write access instead of just changing the permission of the old area. Is it not true by this time the child would have got its own frame area with read-write access?

   If the child has got his own area with read-write access, the count of processes using the read-only page goes down to 1. If the count is 1 and the page is read-only, the page can be changed to read-write and does not need to be copied.

2. Calculation of address using shift operations.

   Let us look at the translation of m = 6, using the shift operations. In the logical memory, location 6 contains g.

   a) Our first step is to find the page number in logical memory. Since each page is 4 bytes (the least significant two bits), we can right shift the address by two bits (or divide by 4) to get the page number. Consider logical address a = 6 = 00110. If you right shift 00110 by two bits, the least significant two bits (10) will be lost and you have 00001 (which is the same result as 6/4). 1 is the index of page in page table and you can find the corresponding frame number from that index in memory. So, the frame number now is (from the page table) 6 (entry at index 1 in the page table). We have a = 00110, a>>2 = 00001, and if m signifies page table, the frame number as p = m[a>>2] = m[2] = 6 = 00110.

   b) We also need to keep track of the page offset which is the least significant two bits (the ones we let fall off when we right shifted). We can get those by changing all the bits corresponding to page numbers to zero. This, we compute d = a & 0X3 = 00110 & 00011 = 00010. That corresponds to byte 2 in the page.

   c) Now, we can combine the above two results to get the final physical address A by shifting the frame number two bits to the left and adding the page offset. So we do A = (p<<2)|d = (00110 << 2) | 00010 = 11000 | 00010 = 11010 which translates to location 26 in decimal which is where you find g in physical memory.

3. Difference between pages and segments.
   Pages are all fixed size while segments are variable –sized. Other than that, there are some similarities (page-table/segment-table; base address+offset; permissions on page/segment). I hope that as you listened to further lectures, the concepts have become clear. If not, we can always do a zoom session.