CSCI350 FRQ 4

1- Explain walkpgdir() line by line referring code below. When do you use it and what arguments should you include? (3pts)

```c
// Return the address of the PTE in page table pgdir
// that corresponds to virtual address va.  If alloc!=0,
// create any required page table pages.
static pte_t *
walkpgdir(pde_t *pgdir, const void *va, int alloc)
{
  pde_t *pde;
  pte_t *pgtab;

  pde = &pgdir[PDX(va)];
  if(*pde & PTE_P){
    pgtab = (pte_t*)P2V(PTE_ADDR(*pde));
  } else {
    if(!alloc || (pgtab = (pte_t*)kalloc()) == 0)
      return 0;
    // Make sure all those PTE_P bits are zero.
    memset(pgtab, 0, PGSIZE);
    // The permissions here are overly generous, but they can
    // be further restricted by the permissions in the page table
    // entries, if necessary.
    *pde = V2P(pgtab) | PTE_P | PTE_W | PTE_U;
  }
  return &pgtab[PTX(va)];
}
```

2 - What is the functionality of mappages() and when do we need it? Explain **a** and **last** pointers and how they are used referring code below.(3pts)

```c
static int
mappages(pde_t *pgdir, void *va, uint size, uint pa, int perm)
{
  char *a, *last;
  pte_t *pte;

  a = (char*)PGROUNDDOWN((uint)va);
  last = (char*)PGROUNDDOWN(((uint)va) + size - 1);
  for(;;){
    if((pte = walkpgdir(pgdir, a, 1)) == 0)
      return -1;
    if(*pte & PTE_P)
      panic("remap");
    *pte = pa | perm | PTE_P;
    if(a == last)
      break;
    a += PGSIZE;
    pa += PGSIZE;
  }
  return 0;
}
```

3 - What is the difference between memmove() and kalloc() functions? (2pts)

4 - In discussion, we covered an edge case while decrementing counter. Explain what is that and which functions needs to be implemented considering this edge case? (3pts)

5 - (T / F) A counter for each process is not necessary. (1pt)

6 - (T / F) We need to allocate memory in the implementation of cow(). (1pt)

7 - Why do we implement cow() and pagefault() separately? List at least 2 differences between cow and pagefault functions. (2pts)