1.

Initially, a TLB miss occurs, prompting the access of the page table. Subsequently, the operating system locates the Page Table Entry (PTE) associated with the page and determines that it is not present in physical memory (indicated by the present bit being 0). The operating system proceeds to search for an available physical page frame to map the virtual page to. If no such page frame exists, the replacement algorithm is invoked. Following this, the operating system identifies a page to be swapped out based on a specific rule and releases the corresponding page frame. Lastly, the operating system swaps the desired page into the found physical frame and retries the instruction.

2.

```c
static int _clock_init_mm(struct mm_struct *mm) {
    // TODO
    list_init(&pra_list_head);
    mm->sm_priv = &pra_list_head;
    curr_ptr = &pra_list_head;

    return 0;
}

static int _clock_map_swappable(struct mm_struct *mm, uintptr_t addr,
                                struct Page *page, int swap_in) {
    // TODO
    list_entry_t *head = (list_entry_t *)mm->sm_priv;
    list_entry_t *entry = &(page->pra_page_link);

    assert(entry != NULL && head != NULL);
    list_add(curr_ptr, entry);
    struct Page *p = le2page(entry, pra_page_link);
    pte_t *ptep = get_pte(mm->pgdir, p->pra_vaddr, 0);
    *ptep |= PTE_A;

    return 0;
}

static int _clock_swap_out_victim(struct mm_struct *mm, struct Page **ptr_
                                  int in_tick) {
    // TODO
    list_entry_t *head = (list_entry_t *)mm->sm_priv;
    assert(head != NULL);
    assert(in_tick == 0);
    if (curr_ptr == head) {
        curr_ptr = list_prev(curr_ptr);
        if (curr_ptr == head) {
            *ptr_page = NULL;
            return 0;
        }
    }

    while (true) {
        if (curr_ptr == head)
            curr_ptr = list_prev(curr_ptr);
        struct Page *p = le2page(curr_ptr, pra_page_link);
        pte_t *ptep = get_pte(mm->pgdir, p->pra_vaddr, 0);
        if ((*ptep & PTE_A)) {
            *ptep ^= PTE_A;
            curr_ptr = list_prev(curr_ptr);
        } else {
            list_entry_t *entry = curr_ptr;
            curr_ptr = list_prev(curr_ptr);
            list_del(entry);
            *ptr_page = le2page(entry, pra_page_link);
            return 0;
        }

    }
```

```
}

static int _clock_check_swap(void) {
```

```
memory management: default_pmm_manager
membegin 80200000 memend 88000000 mem_size 7e00000
physcial memory map:
  memory: 0x07e00000, [0x80200000, 0x87ffffff].
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
check_vma_struct() succeeded!
Store/AMO page fault
page falut at 0x00000100: K/W
check_pgfault() succeeded!
check_vmm() succeeded.
SWAP: manager = clock swap manager
BEGIN check_swap: count 3, total 31660
setup Page Table for vaddr 0X1000, so alloc a page
setup Page Table vaddr 0~4MB OVER!
set up init env for check_swap begin!
Store/AMO page fault
page falut at 0x00001000: K/W
Store/AMO page fault
page falut at 0x00002000: K/W
Store/AMO page fault
page falut at 0x00003000: K/W
Store/AMO page fault
page falut at 0x00004000: K/W
set up init env for check_swap over!
---------Clock check begin----------
write Virt Page c in clock_check_swap
write Virt Page a in clock_check_swap
write Virt Page d in clock_check_swap
write Virt Page b in clock_check_swap
write Virt Page e in clock_check_swap
Store/AMO page fault
page falut at 0x00005000: K/W
swap_out: i 0, store page in vaddr 0x1000 to disk swap entry 2
write Virt Page b in clock_check_swap
write Virt Page a in clock_check_swap
Store/AMO page fault
page falut at 0x00001000: K/W
swap_out: i 0, store page in vaddr 0x3000 to disk swap entry 4
swap_in: load disk swap entry 2 with swap_page in vadr 0x1000
write Virt Page b in clock_check_swap
write Virt Page c in clock_check_swap
Store/AMO page fault
page falut at 0x00003000: K/W
swap_out: i 0, store page in vaddr 0x4000 to disk swap entry 5
swap_in: load disk swap entry 4 with swap_page in vadr 0x3000
write Virt Page d in clock_check_swap
Store/AMO page fault
page falut at 0x00004000: K/W
swap_out: i 0, store page in vaddr 0x5000 to disk swap entry 6
swap_in: load disk swap entry 5 with swap_page in vadr 0x4000
```

```
write Virt Page e in clock_check_swap
Store/AMO page fault
page falut at 0x00005000: K/W
swap_out: i 0, store page in vaddr 0x2000 to disk swap entry 3
swap_in: load disk swap entry 6 with swap_page in vadr 0x5000
write Virt Page a in clock_check_swap
Clock check succeed!
check_swap() succeeded!
QEMU: Terminated
root@Bill:~/codes/CS334-OS/ass09#
```