Data structures and algorithms

1) This question considers the Heap data structure and sorting algorithms. In the discussion to follow, assume that the binary heap “H” has “n” nodes, and that each node stores a single integer value.

a. [20 points] What is one fundamental difference between a binary heap and a binary tree?

b. [30 points] Use pseudo-code to describe an algorithm HeapInsert(H,v) that inserts a new value “v” into “H” and preserves the heap property. What is its time complexity?

c. [30 points] Use pseudo-code to describe an algorithm v=HeapRemove(H) that returns the value “v” stored in the root of “H”, and removes the root node from “H” (preserving the heap property). What is its time complexity?

d. [20 points] Use pseudo-code to describe an algorithm for sorting using heap inserts and removes. What is its time complexity?

Operating Systems

2) Consider a single-processor computer and a multi-task O/S with 32-bit virtual and physical address spaces. The page table is flat; page sizes are fixed (4Kbytes). Page table entries (PTEs) are byte-aligned and hold, in addition to the physical page frame address, the following bits:

RW	(RW=0: read-only, RW=1: readable and writable)
X	(X=0: not executable, X=1: executable)
V	(V=0: translation invalid; V=1: valid translation)

a) [10 points] **What is the total memory size occupied by page tables if there are three processes running in the system?**

b) [45 points] Consider the following PTEs for processes P1, P2, P3. The virtual address space starts at 0x00000000h, and only the first four PTEs of the page table are shown. **Explain what actions (if any) the O/S kernel needs to perform to handle the memory accesses described below:**

b.i) P1 issues a "load" instruction to read a byte from virtual address 0x00000002
b.ii) P2 issues a "store" instruction to write a byte to virtual address 0x00000000
b.iii) P3 issues a "store" instruction to write a byte to virtual address 0x00003000

Process P1	**Page frame**	**RW**	**X**	**V**
PTE 0	0x00001	0	0	1
PTE 1	0x00003	0	0	0
PTE 2	0x00002	0	0	0
PTE 3	0x00000	0	0	1

Process P2	**Page frame**	**RW**	**X**	**V**
PTE 0	0x00001	1	0	0
PTE 1	0x00002	0	0	0
PTE 2	0x00003	1	0	1
PTE 3	0x00000	1	0	0

Process P3	**Page frame**	**RW**	**X**	**V**
PTE 0	0x00001	1	0	0
PTE 1	0x00003	0	0	0
PTE 2	0x00000	1	0	0
PTE 3	0x00002	0	0	1

c) [45 points] Assume this system has a translation look-aside buffer (TLB) with two entries. It has its own valid bit V (V=1: valid entry, V=0: invalid). The TLB does *not* have address space identifiers. Tables I, II and III below show different snapshots of the state of the TLB. **For each table, explain in one sentence why it is possible (or why not possible) for the TLB to be in the specified state**, assuming there is only one process running in the system (P1,P2, or P3 with page tables as specified above).

Virt. Addr. tag	**Page frame**	**RW**	**X**	**V**
0x00000	0x00001	0	0	1
0x00003	0x00000	0	0	1

Table I

Virt. Addr. tag	**Page frame**	**RW**	**X**	**V**
0x00002	0x00003	1	0	1
0x00003	0x00002	0	0	1

Table II

Virt. Addr. tag	**Page frame**	**RW**	**X**	**V**
0x00000	0x00001	0	0	0
0x00003	0x00002	0	0	1

Table III