<u>Assignment 5</u>
Satyaki Ghosh
10077685
T #01

1. Diagrams of the partitions after the OS has placed the 4 processes using different algorithms.
   P1 – 212 KB, P2 – 417 KB, P3 – 112 KB, P4 – 426 KB

Free	P 10	**P 1**	**P 3**	Free	P 11	Free	P 12	Free	P 13	**P 2**	Free
100 KB	30 KB	**212 KB**	**112 KB**	176 KB	30 KB	200 KB	30 KB	300 KB	30 KB	**417 KB**	183 KB

<u>First-Fit</u>: P4 could not be placed because there were no free spots that could hold 426 KB

Free	P 10	**P 2**	Free	P 11	**P 3**	Free	P 12	**P 1**	Free	P 13	**P 4**	Free
100 KB	30 KB	**417 KB**	83 KB	30 KB	**112 KB**	88 KB	30 KB	**212 KB**	88 KB	30 KB	**426 KB**	174 KB

<u>Best-Fit</u>: All the process could be placed

Free	P 10	**P 2**	Free	P 11	Free	P 12	Free	P 13	**P 1**	**P 3**	Free
100 KB	30 KB	**417 KB**	83 KB	30 KB	200 KB	30 KB	300 KB	30 KB	**212 KB**	**112 KB**	276 KB

<u>Worst-Fit</u>: P4 could not be placed because there were no free spots that could hold 426 KB

Free	P 10	**P 1**	Free	P 11	Free	P 12	Free	P 13	**P 2**	**P 3**	Free
100 KB	30 KB	**212 KB**	288 KB	30 KB	200 KB	30 KB	300 KB	30 KB	**417 KB**	**112 KB**	71 KB

<u>Next-Fit</u>: P4 could not be placed because there were no free spots that could hold 426 KB

2. Page numbers and offsets for a system with 1KB (1024 bytes or 2^{10} bytes):

Address	**Page Number**	**Offset (10 bits)**
2375	0010 00	11 0111 0101
19366	0001 1001 00	11 0110 0110
30000	0011 0000 00	00 0000 0000
256	00	10 0101 0110
16385	0001 0110 00	11 1000 0101

3. For a 32-bit logical address space, 4 KB (2^{12} bytes) page size and 512 MB (2^{29} bytes) physical memory system:
   a. Conventional Single-Level Page Table:
      # of entries = **1,048,576 entries = $\mathbf{2^{20}}$** = (2^{32} / 2^{12})

   b. Inverted Page Table:
      # of entries = **131,072 entries = $\mathbf{2^{17}}$** = (2^{29} / 2^{12})

4. A direct memory reference (ma) takes 200ns
   a. Single-Level Page Table:
      It will take **400ns** to locate and reference a page in memory. 200ns to access the page table + 200ns to access the word in memory, thus in total it will take 400ns.

b. If a TLB is added:
75% (0.75) TLB Hit (p)
10ns TLB search time (tlbs)
Effective memory-access time = (1 – p) * (tlbs + 2 * ma) + p * (tlbs + ma)
Effective memory-access time = (1 – 0.75) (10 + 2*200) + 0.75 (10 + 200)
= 0.25 * 410 + 0.75 * 210 = 260ns
The effective access time is **260ns** if we add a TLB.

5. 3 Frames in physical memory.
Page Reference string: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6
Yellow – Fault **Green** – No Fault **Red** – Least Recently Used Page

Using LRU algorithm:

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
	2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
		3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6

Thus, 15 page faults using LRU

Yellow – Fault **Green** – No Fault
Using Optimal Replacement algorithm:

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	6
	2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2
		3	4	4	4	5	6	6	6	6	6	6	6	6	6	1	1	1	1

Thus, 11 page faults using OPT

6. Code in file pagesim.cpp
7. Code in file impl.cpp