Operating Systems

Homework Assignment# 3

[Note: Some problems *may* contain less information than needed to solve it. In that case, you need to make some assumptions to continue the solution. In other problems more information may be given than what is needed to solve the problem. You can neglect that extra information.]

---

1. A 2D long integer array, A[64][64], is stored in memory in row-major order (i,e, A[0][0] is followed by A[0][1], then A[0][2], … etc). Assume the whole physical memory can hold only four pages and that the size of each page is 512 bytes. Remember that A[x][y] where x is the row number and y is the column number. Let's assume the page replacement policy used by the OS is first-in-first-out.

Piece A	Piece B

for (int j = 0; j < 64; j++)
  for (int i = 0; i < 64; i++)
  A[i][j] = 0;
	for (int i = 0; i < 64; i++)
	  for (int j = 0; j < 64; j++)
	  A[i][j] = 0;

a) [4] Which of the two pieces, piece A and piece B, of code results in more page faults? Justify your choice.
b) [2] How many elements of array A can each virtual page hold? Show your calculations to get full credit.
c) [2] How many elements of array A can each physical page (sometimes called page slot) hold? Show your calculations to get full credit.
d) [2] How many pages faults does piece A of code cause? Show your calculations to get full credit.
e) [2] How many pages faults does piece B of code cause? Show your calculations to get full credit.

---

2. Suppose we have a system with four CPUs. Each CPU can execute one thread at a time. Two processes start. No other processes exist in the system. Neglect the OS system overhead and its usages of CPUs.

a) [4] What is the minimum number of CPUs that these two processes can use, assuming none of them will be blocked or exits? Explain.
b) [4] What is the maximum number of CPUs that these two processes can use? Explain.

---

3. [10] Suppose you wrote a multithreaded program where you spawn x threads. During execution time we may see *less than x* threads, from that process, executing in parallel. That is, as a programmer, you want, for example, five threads to be executing in parallel. But, during execution, you see that there are less than five threads really executing. State *five* scenarios that may cause this to happen.
