Solution 4

1.

```
for (p = 0; p < 2; p++) {
  for (q = 0; q < 2; q++) {
    for (i = p*64; i < (p+1)*64; i++) {
      for (j = q*64; j < (q+1)*64; j++) {
        Y[i] = Y[i] + A[i][j]*X[j];
      }
    }
  }
}
```

When `p = 0` we compute `Y[0:63]` in 2 steps: first, we use `A[0:63][0:63]` and `X[0:63]` when `q = 0`; then, we use `A[0:63][64:127]` and `X[64:127]` when `q = 1`. When `p = 1` we compute `Y[64:127]` in 2 steps: first, we use `A[64:127][0:63]` and `X[0:63]` when `q = 0`; then, we use `A[64:127][64:127]` and `X[64:127]` when `q = 1`.

Storing one **`64x64`** block of 32-bit numbers (for matrix **A**) requires 64*64*4=**16KB** of memory, and storing two **`128x1`** blocks of 32-bit numbers (for vectors **X** and **Y**) requires 2*128*4=**1KB** of memory. Hence, the cache size should be at least **17KB**.

2. The size of `double X[256][256]` is 256*256*8=**512KB**, and each row of `X` requires 256*8=**2KB**. For every iteration of the outer loop (index `i`), we have 1 read per row element `X[i][j]` in the first inner loop, plus 1 read and 1 write per row element `X[i][j]` in the second inner loop, i.e., each row `i` requires (2+1)*256=768 accesses to it. The total number of accesses is 256*768=196,608.

If we have two **2-KB** pages, we get 1 page fault per row (per 768 accesses), or 256 page faults in total (per 196,608 accesses). Therefore, the page fault rate is 1/768, or equivalently, 256/196,608 = 0.130%.

If we have one **4-KB** page, we get 1 page fault per 2 rows (per 2*768 accesses), or 256/2=128 page faults in total (per 196,608 accesses). Therefore, the page fault rate is 1/(2*768), or equivalently, 128/196,608 = 0.065%.

In both cases the allocated memory amount is the same (**4KB** total), but the page fault rates are different.

3. The size of `float X[256][256]` is 256*256*4=**256KB**, where each row requires 256*4=**1KB**. For the first loop (computing `trace`), we have 1 read per row `i`, i.e., there are 256 accesses required. For the other two nested loops (normalizing `X[i][j]`), we have 1 read and 1 write per row element `X[i][j]`, i.e., each row `i` requires (1+1)*256=512 accesses to it. Hence, the total number of accesses is 256+256*512=131,328.

If we have four **1-KB** pages, we get 256 page faults due to the first loop, and 256 page faults due to the other two nested loops; therefore, the page fault rate is (256+256)/131,328 = 0.3899%.

If we have one **4-KB** page, we get 256/4=64 page faults due to the first loop, and 256/4=64 page faults due to the other two nested loops; therefore, the page fault rate is (64+64)/131,328 = 0.0975%.

In both cases the allocated memory amount is the same (**4KB** total), but the page fault rates are different.