Solution 5

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 64-bit numbers (for matrix **A**) requires 64*64*8=**32KB** of memory, and storing two `128x1` blocks of 64-bit numbers (for vectors **X** and **Y**) requires 2*128*8=**2KB** of memory. Hence, the cache size should be at least **34KB**. Rounding up to the closest power of 2 gives us the cache size of **64KB**.

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; hence, 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 `int X[256][256]` is 256*256*4 = **256KB**, and each row of `X` requires 256*4 = **1KB**. For every iteration of the outer loop (index `i`), we have 2 reads and 1 write per row element `X[i][j]`; hence, each row `i` requires (2+1)*256 = 768 accesses to it. The total number of accesses is 256*768 = 196,608.

If we have four **1-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 = 0.1302%.

If we have one **4-KB** page, we get 1 page fault per 4 rows (per 4*768 accesses), or 256/4 = 64 page faults in total (per 196,608 accesses). Therefore, the page fault rate is 1/(4*768), or equivalently, 64/196,608 = 0.0326%.

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