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

2. The size of `float X[512][512]` is 512*512*4=**1 MB**, where each row requires 512*4=**2 KB**. The first two loops over all row indices `i` and column indices `j` perform 512*512 reads. The third loop (over diagonal elements of `X`) performs 1 read and 1 write 512 times, which amounts to 1024 accesses. The total number of accesses is 512*512+1024 = 263,168.

With four **1-KB** pages (1 page = 1/2 row), for each outer loop iteration `i`, we get 1 page fault at column `j=0` and 1 page fault at column `j=256`, i.e., 2 page faults per row (for the first two loops). For the third loop, we get 1 page fault at every index `i`, or 512 faults in total. The page fault rate is (2*512+512)/263,168 = 0.5837%.

With two **2-KB** pages (1 page = 1 row), we get 1 page fault per iteration `i` (per row) for the first two loops and the third loop. The page fault rate is (512+512)/263,168 = 0.3891%.

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

3. The size of `unsigned int X[1024][1024]` is 1024*1024*4 = **4096 KB**, and each row of `X` requires 1024*4 = **4 KB**. For every iteration of the outer loop (row index `i`), we have two inner loops. In the first double-loop (finding `Max`), for each row index `i` and each column index **`j`**, there is 1 (read) access to **`X`**. In the second double-loop (normalizing to `Max`), for each row index `i` and each column index **`j`**, there are 2 (read + write) accesses to **`X`**. Hence, the total number of accesses is 1024*1024*1 + 1024*1024*2 = 3,145,728.

If we have four **1-KB** pages, we get 4 page faults per row (as each row requires four **1-KB** pages) in the first double-loop, as well as 4 page faults per row in the second double loop, or 8,192 page faults in total (per 3,145,728 accesses). Therefore, the page fault rate is 8,192/3,145,728 = 0.2604%.

If we have one **4-KB** page, we get 1 page fault per row in both double-loops, or 2,048 page faults in total (per 3,145,728 accesses). Therefore, the page fault rate is 2,048/3,145,728 = 0.0651%.

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