Exercise 5

1. Turn each of the loops in Figure 1 into branch-free loops.

```c
int count = 0;
for (int i = 0; i < array_size; ++i){
    if (array[i] < 100)
        count++;
}
```

```c
int count = 0;
for (int i = 0; i < array.size; ++i){
    if (array[i] < 100)
        count++;
    else
        count--;
}
```

Figure 1: Loops for different countings.

2. Considering an unrolling of a selection that is using branching code. What does the optimal unrolling depth depend on in this case? What could be done to have a stable optimal unrolling depth?

3. For which of the following database tasks does an acceleration using SIMD capabilities make sense?

   - Selections
   - Compression
   - Hashing
   - Sorting
   - Grouping
   - Aggregation
   - Insert/Update/Delete

4. Recapitulate the possible ways to program for SIMD. Characterize them regarding level of needed skills, portability, performance boost.

5. Why is unaligned access for SIMD much slower? What is the alignment offset and how could you overcome this alignment hazard?

6. Express the following code using SSE intrinsics

```
int a[N], b[N];
int x = 0;
for (i = 0; i < N; i++)
x = x + a[i]*b[i];
```

7. Given the numbers (4, 7, 1, 5), sort them using the sorting network presented in the lecture. Additionally, use the min/max operations to get the sorted output.

8. Construct the sequence of min/max operations for the odd-even merge network. What is the reason, that a bitonic merge network is better suited for SIMD than the odd-even merge network?

Good Luck!