key points

use different ways to define the split of data. Like equal_sized partitioning, range-based partitioning and dynamic partitioning.

1

1.1

That is 12, which is B[3]. So the co-rank is (5,4).

1.2

the co-rank is (4,2)

the calculation for thread 2: it start at position k = 6. so the for co-rank, it should be: i+j = 6. The possible can be (2, 4),(3, 3),(4, 2),(5, 1).

After comparing B[1], A[4] and A[5], we get the result of (4,2).

1.3

```
int counter = 0;
int C_length = C_next - C_curr;
int A_length = A_next - A_curr;
int B_length = B_next - B_curr;
int total_iteration = ceilf((float)C_length / tile_size);
int C_completed = 0;
int A_consumed = 0;
int B_consumed = 0;
while(counter < total_iteration){
    for(int i = 0; i < tile_size; i += blockDim.x){
        int idx = A_curr + A_consumed + i + threadIdx.x;
        if(i + threadIdx.x < A_length - A_consumed) {
            int coranked_idx = corank(idx); // Apply corank to adjust the
                index
            A_S[i + threadIdx.x] = A[coranked_idx];
        }
    }
    for(int i = 0; i < tile_size; i += blockDim.x) {
        int idx = B_curr + B_consumed + i + threadIdx.x;
        if(i + threadIdx.x < B_length - B_consumed) {
            int coranked_idx = corank(idx); // Apply corank to adjust the
                index
            B_S[i + threadIdx.x] = B[coranked_idx];
        }
    }
    __syncthreads();
    A_consumed += tile_size;
    B_consumed += tile_size;
    C_completed += tile_size;
    counter++;
}
```

1.4

1.4.1

total number elements is: 1030400 + 608000 = 1638400 elements.

Total Threads = $\frac{\text{Total Elements to Merge}}{\text{Elements per Thread}} = \frac{1{,}638{,}400}{8} = 204,800\text{threads}$

1.4.2

it is $\frac{1638400}{8} = 204800$ threads devided by thread block size: $\frac{204800}{1024} = 200$ thread blocks.
For each block 2 binary seaches is in needed, so the resilt is 200 * 2 = 400 times.

1.4.3

in shared memory each one executes once, so the answer is 200 times.