key points

Primary part of parallel concept.

cudaMemcpy('desintation','source','size','type'):
where type:cudaMemcpyHostToDevice,cudaMemcpyDeviceToDevice...
cudaMalloc((void **) &A, size):
a variable's address and its size.
cudaError_t err='function'
where 'function' is like cudaMemcpy,cudaMalloc...

1 question

Compare with your question:

If we want to use each thread in a grid to calculate one output element of a vector addition, what would be the expression for mapping the thread/block indices to the data index (i)? (A) i=threadIdx.x + threadIdx.y; (B) i=blockIdx.x + threadIdx.x; (C) i=blockIdx.x blockDim.x + threadIdx.x; (D) i=blockIdx.x threadIdx.x;

1.C

i = blockIdx.x · blockDim.x + threadIdx.x

blockIdx.x gives the index of the current block.

blockDim.x gives the number of threads per block.

threadIdx.x gives the index of the current thread within its block.

2.C

Each thread hanles two elements, so it needs to multiply the overall thread index by 2 to get the index of the first element the thread will process.

3.D

2 · blockDim.x is two consecutive elements, and each thread processes one element from each section sequentially. Consider the entire block offset and the per-thread offset within the block.

4.C

Number of blocks = $\frac{8000}{1024} \approx 7.8$

ceil(N) = 8

Total threads = N × 1024 = 8192.

5.D

To allocate an array of v integer elements in CUDA device global memory, you need to specify the total amount of memory required in bytes. Since each integer typically requires sizeof(int) bytes, the correct expression for the second argument of the cudaMalloc call would be:

$$v \times sizeof(int) \tag{1}$$

6.D

When using cudaMalloc() to allocate memory on the device and have a pointer variable A_d point to it, you need to pass the address of the pointer to cudaMalloc() so that it can modify the pointer to point to the allocated memory.

(void**) &A_d

7.C

cudaMemcpy(A_d, A_h, 3000, cudaMemcpyHostToDevice)

explaination:

A_d as the destination pointer on the device.

A_h as the source pointer on the host.

3000 as the number of bytes to copy.

cudaMemcpyHostToDevice as the direction of the copy.

8.C

From the formula on book, it is: cudaError_t err.

9.1:128

The number of threads per block is specified in the kernel launch configuration. In line 9.

9.2:200064

$$Numberofblocks = \frac{200000}{128} \approx 1563(ceil) \tag{2}$$

so the total number of threads is:

$$Totalthreads = 1563 \times 128 = 200064 \tag{3}$$

9.3:1563

As calculated above.

9.4:200064

All threads in the grid execute line 02, as it is part of the kernel's main body. Therefore, the number of threads is the same as the total number of threads in the grid

9.5:200000

Line 04 is executed only if the condition in line 03 is true, i.e., i ¡ N. Since i ranges from 0 to 200063, only the first 200000 threads meet this condition

10. He can use function like __global__ __host__ function...

```
__host__ __device__ void function() {
    // Function implementation
}
```