CS3210

Parallel Computing

Tut M

Mon (4pm)
Tues (2pm)

Admin

Updates

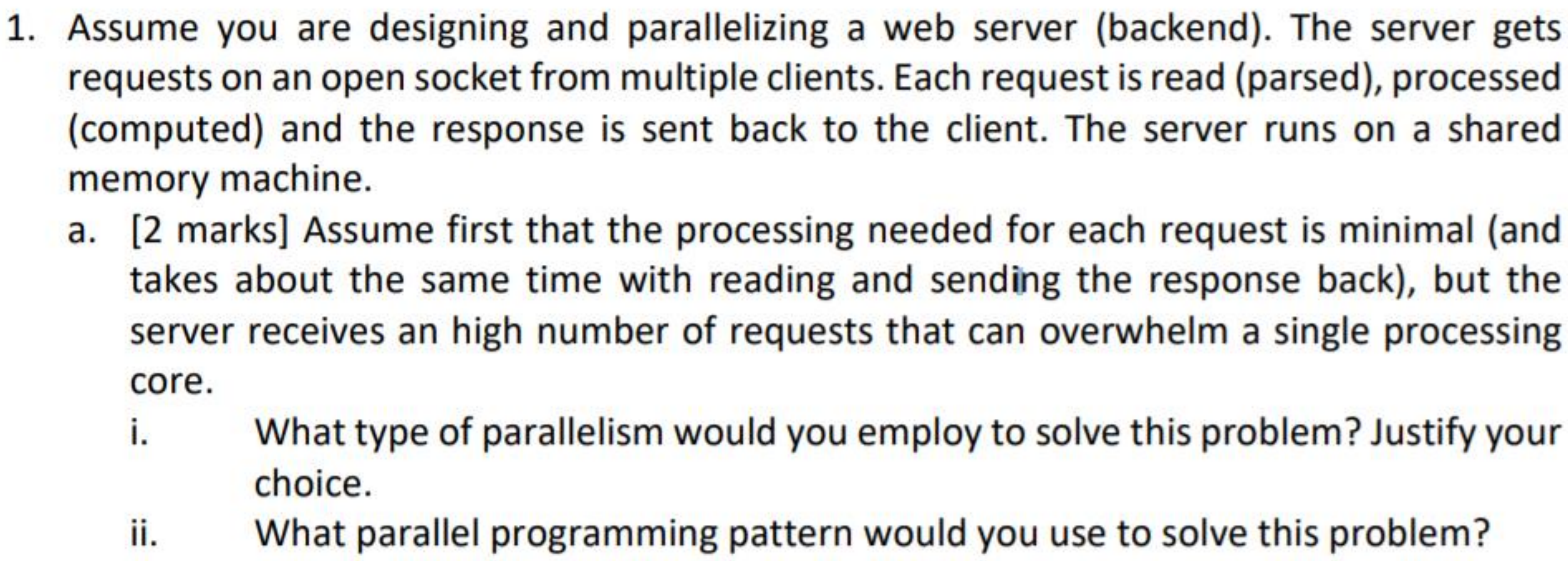
- Assignment 1 still being graded (no ETA)
  - ➢ Late submissions for Part 2 are still accepted

- Mid-terms also being graded (ETA next week?)

- No tutorial quiz today ☺

Q1(a)

Parallel Programming Models



1. Assume you are designing and parallelizing a web server (backend). The server gets requests on an open socket from multiple clients. Each request is read (parsed), processed (computed) and the response is sent back to the client. The server runs on a shared memory machine.
   a. [2 marks] Assume first that the processing needed for each request is minimal (and takes about the same time with reading and sending the response back), but the server receives an high number of requests that can overwhelm a single processing core.
      i. What type of parallelism would you employ to solve this problem? Justify your choice.
      ii. What parallel programming pattern would you use to solve this problem?

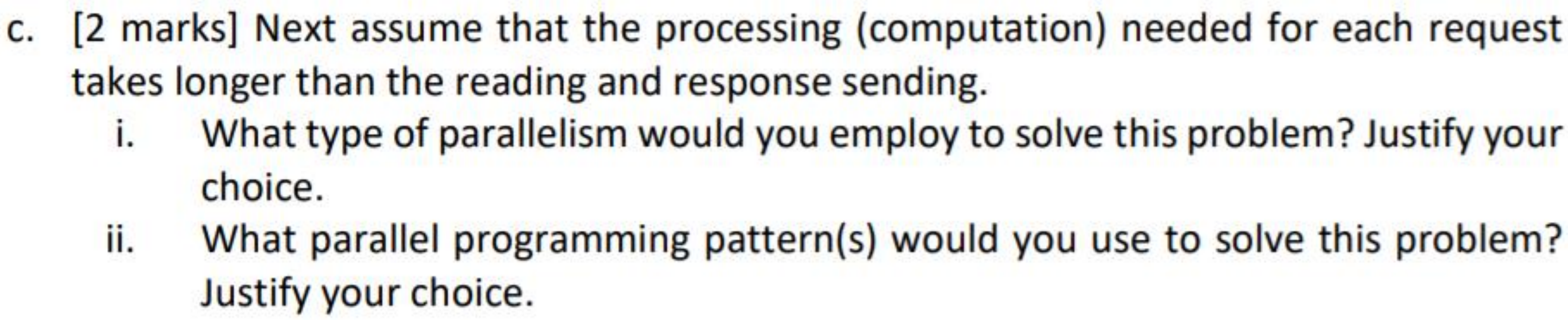
Q1(b)
Parallel Programming Models

b. [2 marks] Now assume that each of the reading of the request and response sending take much longer time than the processing (computation) phase.

i. What type of parallelism would you employ to solve this problem? Justify your choice.

ii. What parallel programming pattern(s) would you use to solve this problem? Justify your choice.

Q1(c)
Parallel Programming Models

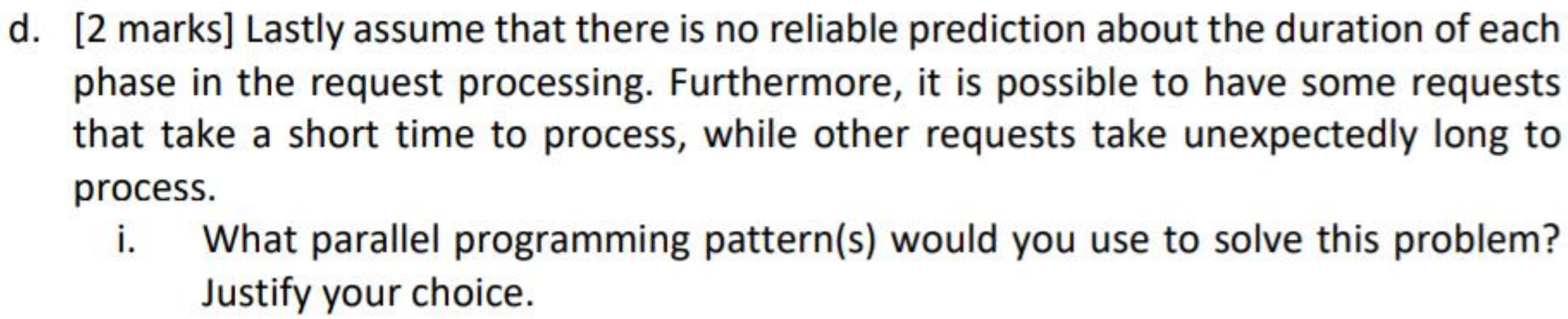


c. [2 marks] Next assume that the processing (computation) needed for each request takes longer than the reading and response sending.

   i. What type of parallelism would you employ to solve this problem? Justify your choice.

   ii. What parallel programming pattern(s) would you use to solve this problem? Justify your choice.

Q1(d)
Parallel Programming Models

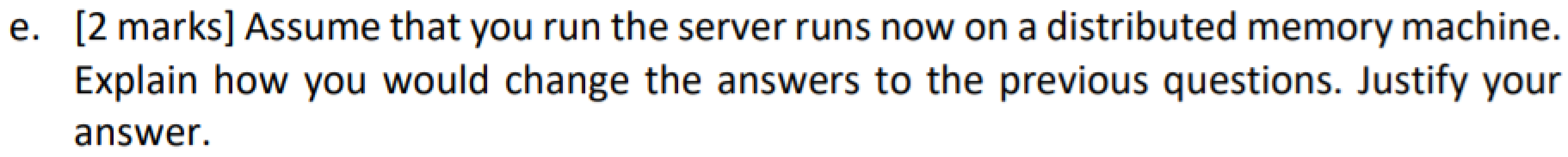


d. [2 marks] Lastly assume that there is no reliable prediction about the duration of each phase in the request processing. Furthermore, it is possible to have some requests that take a short time to process, while other requests take unexpectedly long to process.

   i. What parallel programming pattern(s) would you use to solve this problem? Justify your choice.

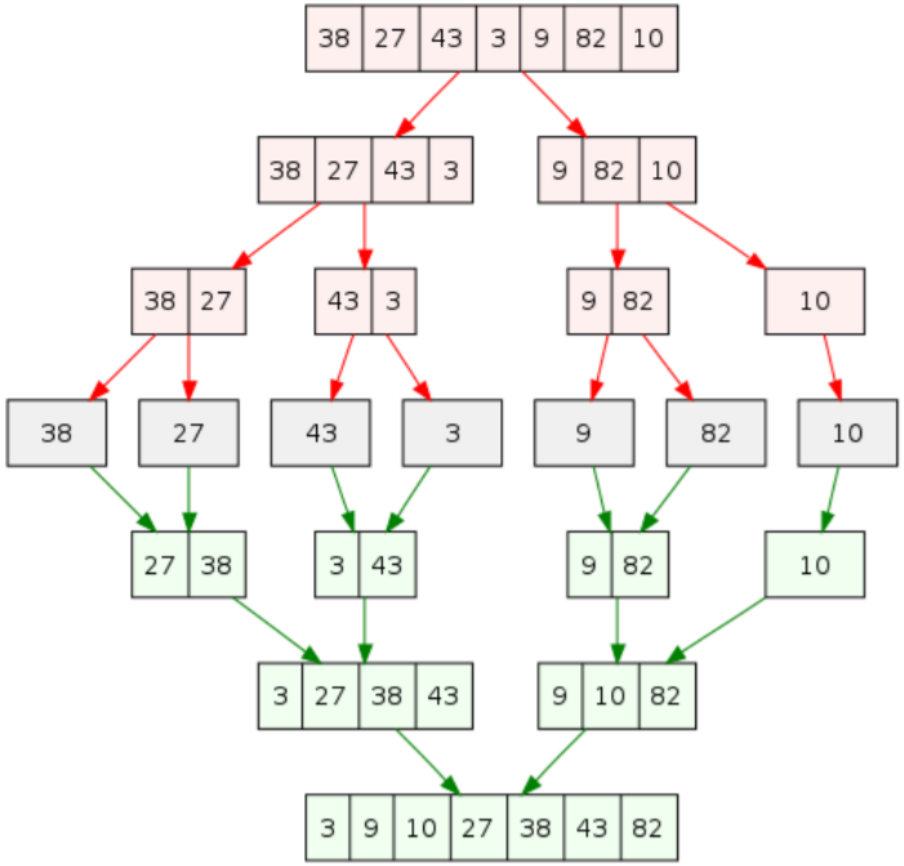
Q1(e)

Parallel Programming Models



e. [2 marks] Assume that you run the server runs now on a distributed memory machine. Explain how you would change the answers to the previous questions. Justify your answer.

Q2
Parallel Mergesort



Q2(a)
Parallel Mergesort

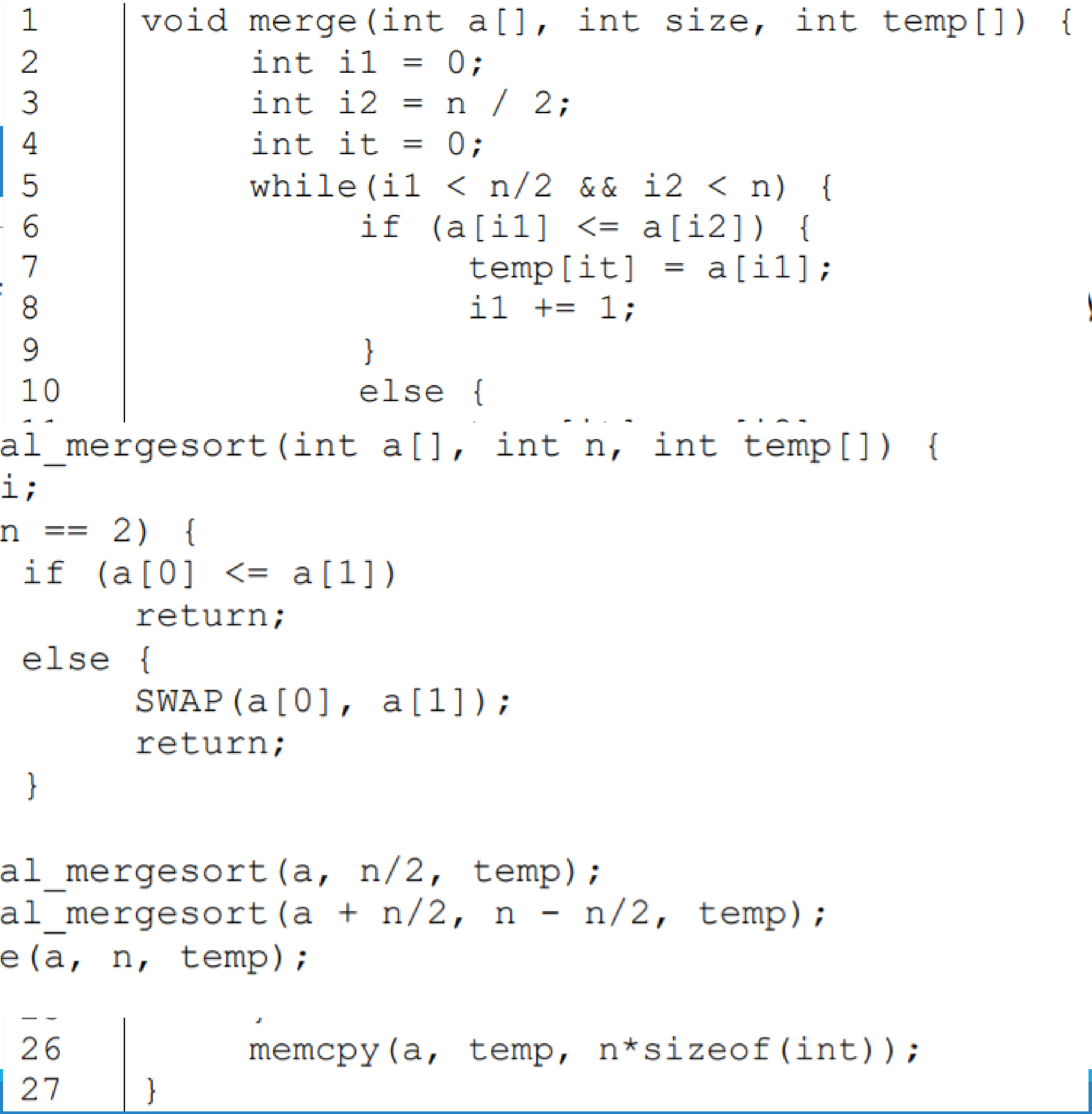
2. Suppose you are working on parallelizing SortN problem from Figure 1. Figure 2 shows a visual representation of the MergeSort algorithm.

    a. [2 marks] What type of parallelism do you observe in the sequential implementation from Figure 1 (data or task)? Briefly justify your choice.

Q2(b)

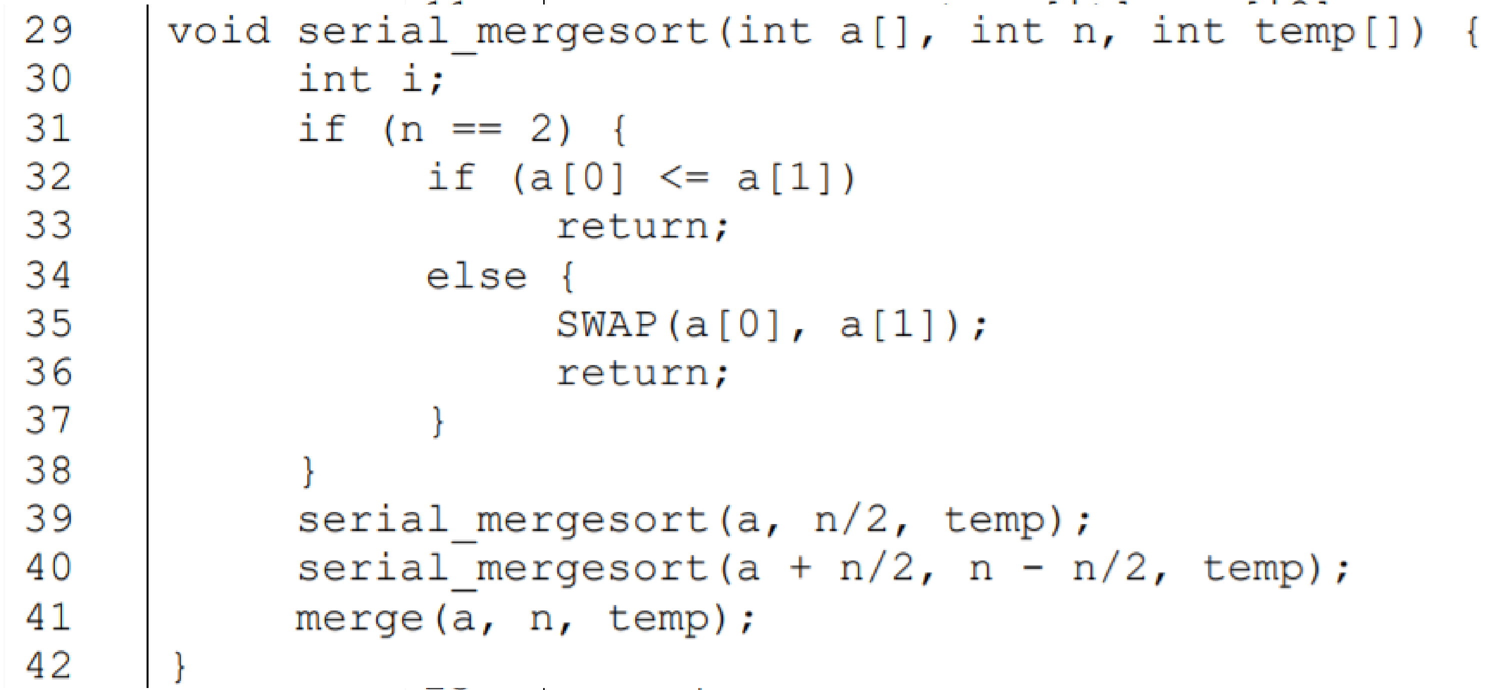
Parallel I

b. [2 marks] What part of y
justify your choice.



```
void merge(int a[], int size, int temp[]) {
      int i1 = 0;
      int i2 = n / 2;
      int it = 0;
      while(i1 < n/2 && i2 < n) {
            if (a[i1] <= a[i2]) {
                  temp[it] = a[i1];
                  i1 += 1;
            }
            else {
```

```
      memcpy(a, temp, n*sizeof(int));
}
```



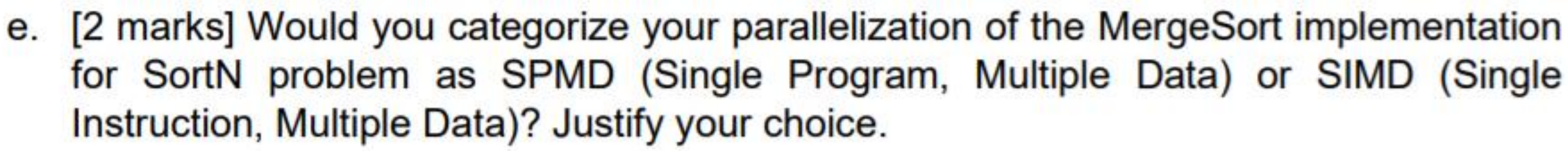
```
void serial_mergesort(int a[], int n, int temp[]) {
      int i;
      if (n == 2) {
            if (a[0] <= a[1])
                  return;
            else {
                  SWAP(a[0], a[1]);
                  return;
            }
      }
      serial_mergesort(a, n/2, temp);
      serial_mergesort(a + n/2, n - n/2, temp);
      merge(a, n, temp);
}
```

Q2(c, d)
Parallel Mergesort

c. [4 mark] What are two problems that you observe when you try to parallelize this sequential implementation for SortN? Explain these problems with examples (Note that there is no need to give a solution for these problems).

d. [2 marks] Next you are required to parallelize the sequential implementation shown in Figure 1 using OpenMP. How would you implement this parallelization (you can use the line numbers to refer to the lines of pseudo-code in Figure 1)?

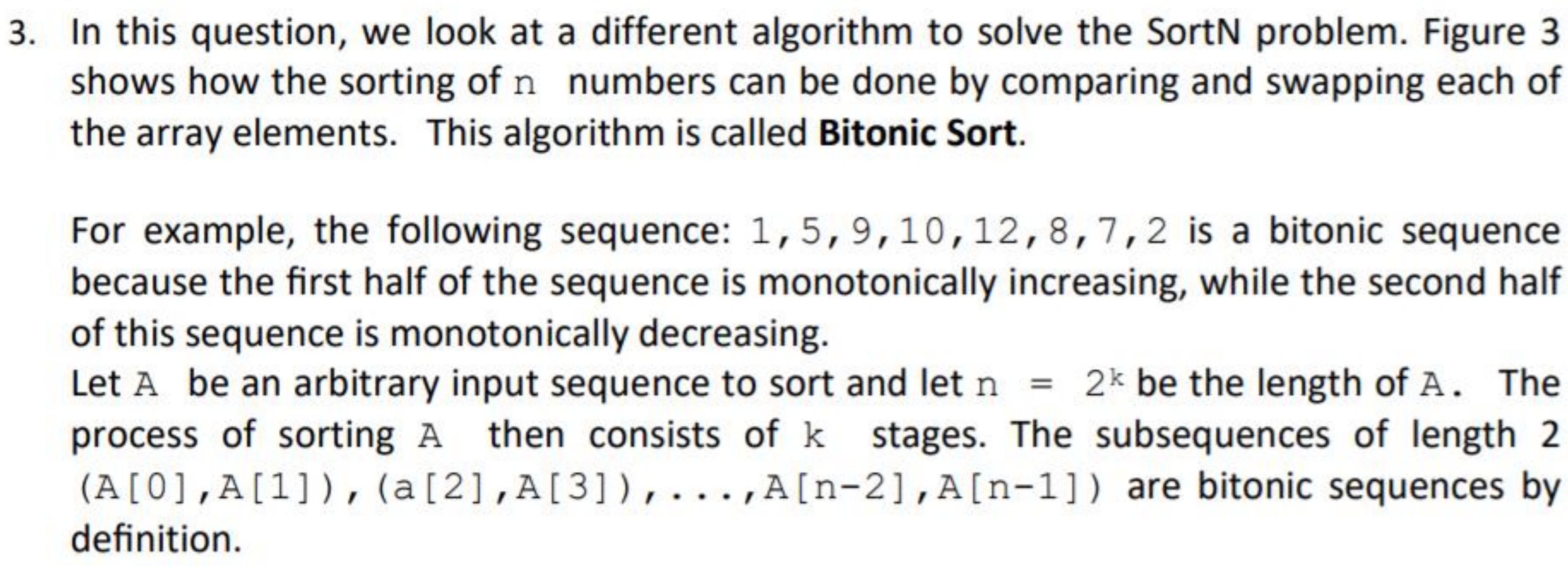
Q2(e)
Parallel Mergesort



e. [2 marks] Would you categorize your parallelization of the MergeSort implementation for SortN problem as SPMD (Single Program, Multiple Data) or SIMD (Single Instruction, Multiple Data)? Justify your choice.

Q3

Bitonic Sort



3. In this question, we look at a different algorithm to solve the SortN problem. Figure 3 shows how the sorting of n numbers can be done by comparing and swapping each of the array elements. This algorithm is called **Bitonic Sort**.

For example, the following sequence: 1,5,9,10,12,8,7,2 is a bitonic sequence because the first half of the sequence is monotonically increasing, while the second half of this sequence is monotonically decreasing.
Let A be an arbitrary input sequence to sort and let n = 2^k be the length of A. The process of sorting A then consists of k stages. The subsequences of length 2 (A[0],A[1]),(a[2],A[3]),...,A[n-2],A[n-1]) are bitonic sequences by definition.

Q3

Bitonic Sort

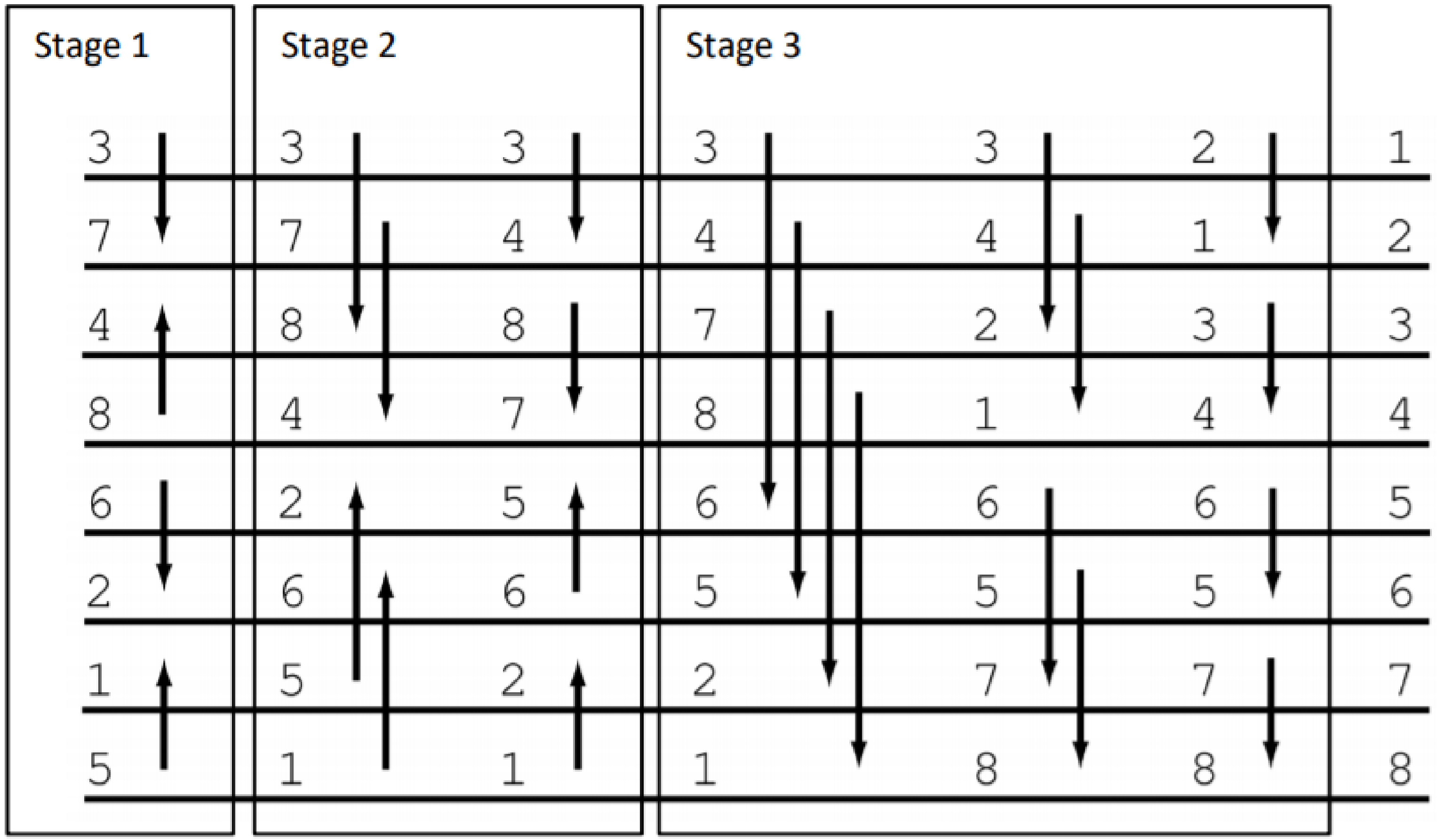
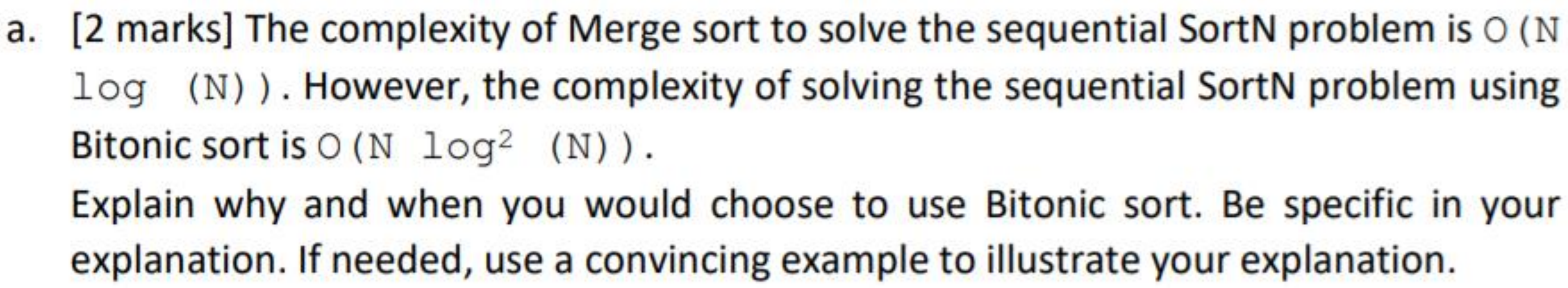


Figure 3: Applying Bitonic Sort for an Array with 8 Values.

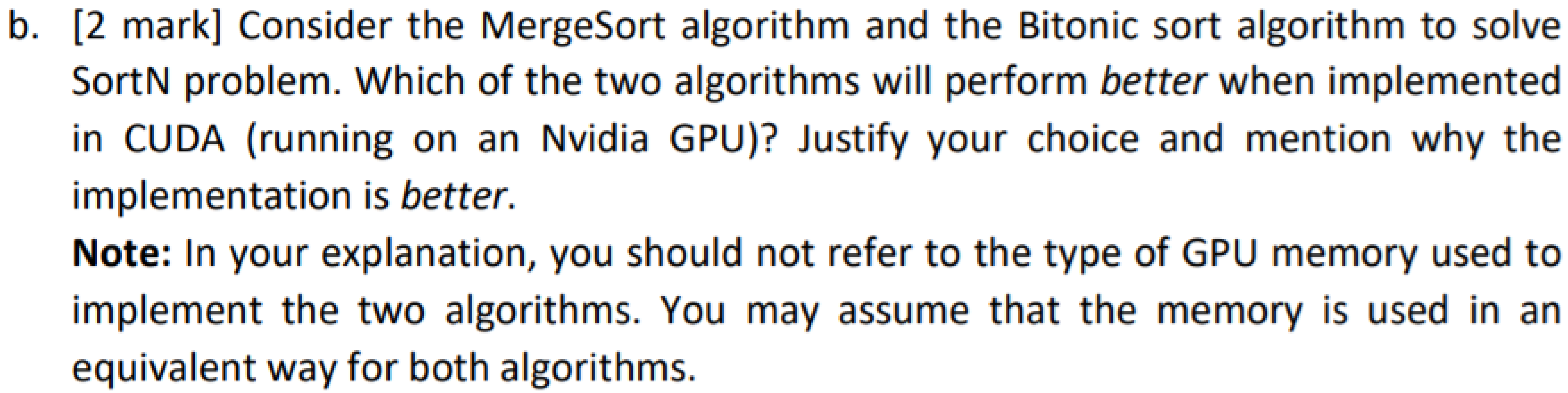
Q3(a)
Bitonic Sort



a. [2 marks] The complexity of Merge sort to solve the sequential SortN problem is `O(N log (N))`. However, the complexity of solving the sequential SortN problem using Bitonic sort is `O(N log² (N))`.

Explain why and when you would choose to use Bitonic sort. Be specific in your explanation. If needed, use a convincing example to illustrate your explanation.

Q3(b)
Bitonic Sort



b. [2 mark] Consider the MergeSort algorithm and the Bitonic sort algorithm to solve SortN problem. Which of the two algorithms will perform *better* when implemented in CUDA (running on an Nvidia GPU)? Justify your choice and mention why the implementation is *better*.

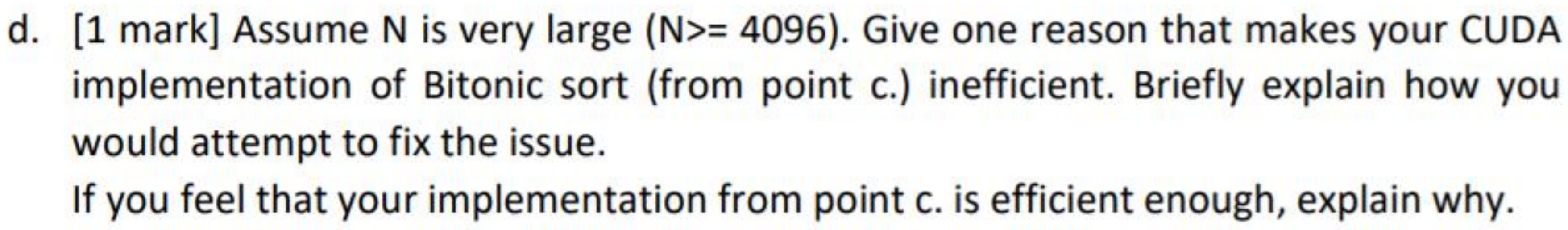
Note: In your explanation, you should not refer to the type of GPU memory used to implement the two algorithms. You may assume that the memory is used in an equivalent way for both algorithms.

Q3(c)
Bitonic Sort

c. [3 marks] Briefly describe the jobs (tasks) done in parallel by each CUDA thread when Bitonic sort algorithm is implemented to solve SortN problem. You should minimally show:

- Kernel code in pseudo-code (or explanation)
- When and how many times is the kernel called
- Number of blocks, number of threads in a block, virtual arrangement of the threads (1D/2D/3D)
- Type of memory that the kernel is using

Q3(d)
Bitonic Sort



d. [1 mark] Assume N is very large (N>= 4096). Give one reason that makes your CUDA implementation of Bitonic sort (from point c.) inefficient. Briefly explain how you would attempt to fix the issue.
If you feel that your implementation from point c. is efficient enough, explain why.

Review

Consistency Models

- Summary table (weak consistency models not covered)

Property	SC	RC: TSO	RC: PC	RC: PSO
Preserves data dep?	Yes	Yes	Yes	Yes
Preserve $W \rightarrow R$	Yes	No	No	No
Preserve $R \rightarrow R$	Yes	Yes	Yes	Yes
Preserve $R \rightarrow W$	Yes	Yes	Yes	Yes
Preserve $W \rightarrow W$	Yes	Yes	Yes	No
Write propagation	Yes	Yes	No	Yes

Return value of any write (even from other processor) before write is propagated or serialised

Q4 [2018 Mid-terms]
Memory Consistency

```
        P1
[ 1] while (Z == 0);
[ 2] print B

        P2
[ 3] while (B == 0);
[ 4] Y = 2
[ 5] A = B

        P3
[ 6] while (B == 0);
[ 7] Z = 1

        P4
[ 8] B = 1
[ 9] while (Y == 0);
[10] print A
```

- Consider this shared memory program that synchronises amongst 4 processors
- Goal of this program: ensure **`print A`** and **`print B`** print same value
  - All variables are initialized to **`0`**

```
        P1
[ 1] while (Z == 0);
[ 2] print B

        P2
[ 3] while (B == 0);
[ 4] Y = 2
[ 5] A = B

        P3
[ 6] while (B == 0);
[ 7] Z = 1

        P4
[ 8] B = 1
[ 9] while (Y == 0);
[10] print A
```

Q4 [2018 Mid-terms]

Memory Consistency

- [1m] Give an execution sequence to show the program fails even under sequential consistency

- [1m] Modify this program to pass under sequential consistency, and explain why it does

```
        P1
[ 1] while (Z == 0);
[ 2] print B

        P2
[ 3] while (B == 0);
[ 4] Y = 2
[ 5] A = B

        P3
[ 6] while (B == 0);
[ 7] Z = 1

        P4
[ 8] B = 1
[ 9] while (Y == 0);
[10] print A
```

Q4 [2018 Mid-terms]
Memory Consistency

- [1m] Does the modified program fail under TSO? If yes, how?
- [1m] Does the modified program fail under PC? If yes, how?
- [1m] Does the modified program fail under PSO? If yes, how?

Review
Cache Coherence Protocols

- Explicit protcols not covered in syllabus

- Further reading on snooping protocols:
  - Write-through protocols: two-state protocol
  - Write-back invalidate protocols: MSI, MESI
  - Write-back update protocols: Firefly, Dragon

CS3210

Parallel Computing

Thank you! Any questions?

Tut ?

Mon (4pm)
Tues (2pm)

Review

Cache Coherence Protocols