National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	Parallel and Distributing Computing	Course Code:	CS3006
Degree Program:	BS (CS)	Semester:	Spring 2022
Exam Duration:	60 Minutes	Total Marks:	30
Paper Date:	07/05/22	Weight	15
Exam Type:	Midterm II	Page(s):	4

Student : Name:_*SOLUTION* ___ Roll No.________________ Section:_______

Instruction: Attempt all questions on the question paper. Rough sheets can be used but it should not be attached. If you think some information is missing then assume it and mention it clearly.

Question No 1: **[marks: 5, CLO: 2]**

(a) We apply `#omp single` within a parallel region to make sure?

a. ***Only one thread executes the code block succeeding this line ****
b. Only the master thread executes the code block succeeding this line
c. Only one thread has access to the shared variables
d. Only p-1 threads will execute the code block succeeding this line

(b) When `OMP_NESTED` is set to `FALSE`, this means:

a. For each new level of nesting within a parallel region, a new team of threads is created
b. ***For each new level of nesting within a parallel region, no new team of threads is created ****
c. Only the master thread will execute any code within the parallel region
d. We cannot use any `for` loops within the parallel region

(c) When we use the `(guided, C)` form of scheduling in OpenMP, loop iterations are scheduled such that:

a. Each thread is always assigned a chunk size equal to C
b. Each thread is always assigned a chunk size not larger than C
c. ***Each thread is always assigned a chunk size of size C or greater ****
d. Each thread is assigned a chunk size of size C^2

(d) All-reduce could also be considered as equivalent to:

a. ***All-to-one reduction by `process x` and then one-to-all broadcast by `process x` ****
b. One-to-all broadcast by `process x` and then all-to-one reduction by `process x`
c. All-to-all broadcast
d. All-to-one reduction by `process 0`, then one-to-all broadcast by `process p-1`

(e) For communication (e.g., all-to-all broadcast or prefix sum) within a hypercube, we used an expression (`my_id XOR` 2^i) within the `for` loop. What was its purpose?

a. ***To decide the ID of the neighbor with which we will exchange messages in this iteration ****
b. To decide the size of the message to be sent in this iteration
c. To decide the total dimensions existing within the hypercube
d. To decide the ID of the farthest node with which we will exchange messages in this iteration

Question No. 2: **[marks: 10, CLO: 2]**

In the code below, some requirements are mentioned as comments. Your task is to write the missing OMP constructs to achieve the desired functionality. Also write the program output in the space given below the code.

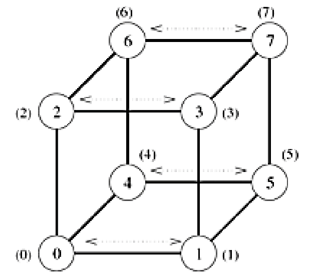
```
int _tmain(int argc, _TCHAR* argv[]) {
      int i;
      int total = 0;

      // the following block of code should be run by 4 threads.
      #pragma omp parallel num_threads(4)

      {
            int Id = omp_get_thread_num();
            int sum = 0;

            // the following line of code should be run by one thread only

            #pragma omp single
            i = 10;

            // each of the following two functions should be called by one
            thread only however it should be executed in parallel
            #pragma omp sections nowait
            {
              #pragma omp section
              functA();


              #pragma omp section
              functB();
            }

            // the following line of code could be executed by the free threads
            before the return of above function calls


            sum = sum + Id;

            // the following line of code should only be executed when all
            threads have executed the above instruction
            #pragma omp barrier

            sum = sum + i;

            printf("The value of sum is %d \n", sum);

            //Add an appropriate construct to perform the following operation
            correctly

            #pragma omp critical
            total = total + sum;
      }
      printf("The value of total is %d \n", total);
      return 0;
}
```

Output: *(0.5*4 marks + 1 mark)*

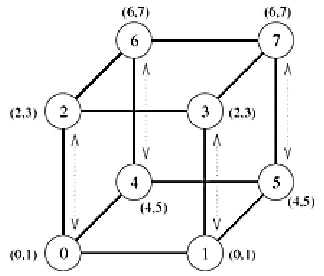
```
The value of sum is 10
The value of sum is 11
The value of sum is 12
The value of sum is 13
The value of total is 46
```

Question No. 3: **[marks: 5, CLO: 3]**

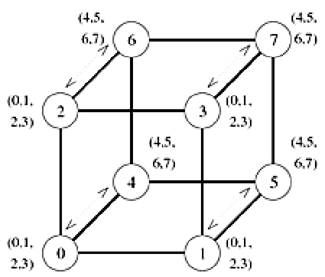
In All-to-All Reduction communication operation, derive the cost expression for the given Hypercube below:



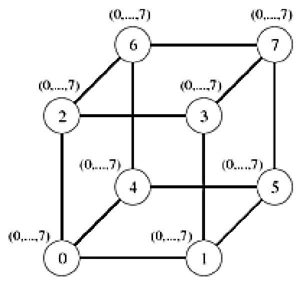
(a) Initial distribution of messages



(b) Distribution before the second step



(c) Distribution before the third step



(d) Final distribution of messages

Solution:

1st step: $t_s + mt_w$

2nd step: $t_s + 2mt_w$

3rd step: $t_s + 4mt_w$

Generalizing: $\sum_{i=1}^{\log p}\{t_s + 2^{i-1}mt_w\} = (t_s * \log p) + \left(2^0 + 2^1 + \cdots + 2^{\log(p-1)}\right) * mt_w$

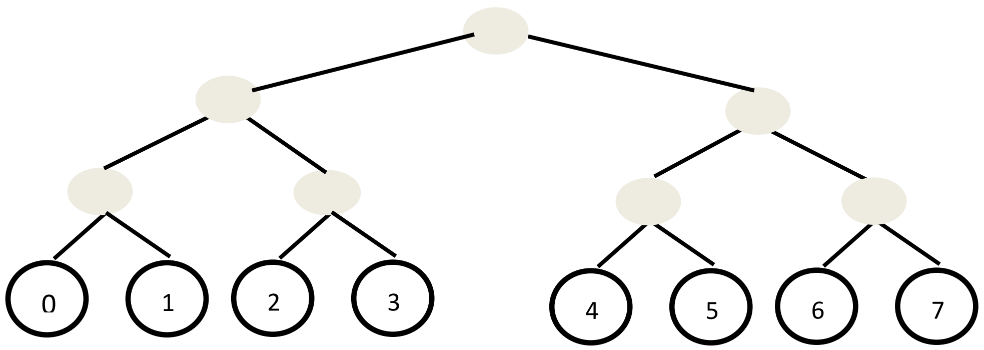
$$(t_s * \log p) + \left(1(\frac{1-2^{\log p}}{1-2})\right) * mt_w$$

$$(t_s * \log p) + \left(2^{\log p} - 1\right) * mt_w$$

$$(t_s * \log p) + (p-1) * mt_w$$

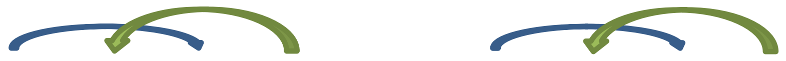
Question No. 4: **[marks: 10, CLO: 3]**

Briefly explain the concept of "Prefix-sum". Now consider the following complete binary tree where we have to perform the operation "Prefix-sum". Assume that the value to be contributed by each node is equal to its Id. Show the calculation at each step and provide the final value at each node at the end of the operation.

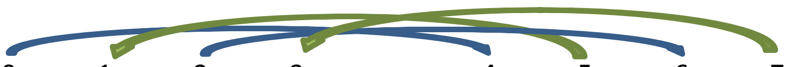


Prefix sum is the distribution of messages such that node i possesses the combined messages of lower nodes. For example, node 4 possesses 0+1+2+3+4. The process works similar to all-to-all broadcast except an associative operator is used on messages received from IDs lower than the current ID. Messages received from higher IDs are forwarded. *(4 marks)*

Step 0	0	1	2	3	4	5	6	7	
Message	0	1	2	3	4	5	6	7	
Result	0	1	2	3	4	5	6	7	
Step 1	0 ←→	1	2 ←→	3	4 ←→	5	6 ←→	7	*(2 marks)*
Message	0+1	0+1	2+3	2+3	4+5	4+5	6+7	6+7	
Result	0	1	2	5	4	9	6	13	



Step 2	0	1	2	3	4	5	6	7	*(2 marks)*
Message	0+..+3	0+..+3	0+..+3	0+..+3	4+..+7	4+..+7	4+..+7	4+..+7	
Result	0	1	3	6	4	9	15	22	



Step 3	0	1	2	3	4	5	6	7	*(2 marks)*
Message	0+..+7	0+..+7	0+..+7	0+..+7	0+..+7	0+..+7	0+..+7	0+..+7	
Result	0	1	3	6	10	15	21	28	