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:	180 Minutes	Total Marks:	70
Paper Date:	23/06/22	Weight	45
Exam Type:	Final	Page(s):	9

Student : Name:__________________________ Roll No._________________ Section:________

Instructions:
1. Attempt all question of **Section A** on the **given answer sheet (submit in the first 30 minutes)**, and **Section B** questions on this **question paper**.
2. Rough sheets can be used but it should not be attached.
3. If you think some information is missing then make some appropriate assumptions.

Section A [marks: 20, CLO: 1]

Mark your answers for all MCQs on the Answer Sheet [submit in the first 30 minutes]

Questions 1 – 10 are worth 1 mark each

1. **OpenMP allows the following operations in the reduction clause:**
   a. **Sum, product**
   b. **Bitwise AND, bitwise OR**
   c. **Logical AND, logical OR**
   d. **Sum, product, bitwise AND, bitwise OR, logical AND, logical OR**

2. **A hypercube has ______**
   a. **d nodes**
   b. **2d nodes**
   c. **2^dnodes**
   d. **None of the above**

3. **There are N processors. Computations of a matrix are divided among N processors. Once they are done, each one of them sends their computed result to a single processor (part of N processors) and it is summed there. How many All-to-One reductions are required?**
   a. **1**
   b. **N - 1**
   c. **N/2**
   d. **N**

4. **For task dependency graphs that are trees, the maximum degree of concurrency is:**
   a. **The ratio of the total amount of work to the critical path length**
   b. **Equal to the sum of the weights of nodes/tasks on the critical path**
   c. **Equal to the sum of the weights of leaves/number of leaves in a tree**
   d. **Equal to the weight of the leaf with the largest weight (among leafs) in a tree**

5. **An example of hybrid decomposition, assuming we want the minimum value within a 16 integer array, using 4 processors, would be:**
    a. **Sending equal chunks (size 4) to each processor, and then using recursive decomposition to find the overall minimum**
    b. **Finding the minimum value using a recursive algorithm on just one processor, and then broadcasting this value to all remaining processors**
    c. **Sending equal chunks (size 4) to each processor, and then selecting a random number out of 4 on each processor, and then randomly choosing one of the processors' selection.**
    d. **Decomposing data based on the output desired**

6. **Speedups can become _________ when we use exploratory decomposition, because of the _______ position of the goal state within the search space**
    a. **anomalous; deterministic**
    b. **anomalous; uncertain**
    c. **deterministic; uncertain**
    d. **deterministic; random**

7. **With the naïve solution for one-to-all broadcast on a ring, we would expect to send ____ messages to the other _____ processes, and this may lead to an __________ of the communication network**
    a. **p; p-1; overutilization**
    b. **p-1; p-1; underutilization**
    c. **p-1; p-1; overutilization**
    d. **$(p-1)^2$; p-1; underutilization**

8. **On a ring, if we move from the naïve solution to recursive doubling for one-to-all broadcast, we would decrease the number of cycles consumed by approximately (assuming p processors):**
    a. **(p-1) – log(p)**
    b. **$(p-1)^2$ – log(p)**
    c. **(p-1) – (2*log(p))**
    d. **((p-1)/2) – (2*log(p))**

9. **Using OpenMP, if we don't use the OpenMP for construct, loop work-sharing can be done by:**
    a. **Modifying the start and end values of the for loop, using a combination of omp_get_thread_num() and omp_get_num_threads()**
    b. **Modifying the start and end values of the for loop, using a combination of omp_set_thread_num() and omp_set_num_threads()**
    c. **Modifying the start and end values of the for loop, using a combination of omp_get_thread_ID() and omp_get_num_threads()**
    d. **Modifying the start and end values of the for loop, with the random() function and omp_get_num_threads()**

10. **In OpenMP, a private variable has _________ address in the _______ context of every thread:**
    a. **the same; execution**
    b. **the same; memory**
    c. **a different; execution**
    d. **a different; variable**

Questions 11 – 15 are worth 2 marks each

11. Assume a sequential program S has an execution time of 650 seconds. Now assume a parallel variant of S takes 85.55 seconds to complete when we have 8 processors available. The Karp-Flatt metric is approximately equal to:
   - **a. 1.15**
   - **b. 0.129**
   - **c. 0.99**
   - **d. 0.007**

12. Considering a 2-D mesh (without wraparound) with M rows and N columns, we would expect arc connectivity of ______, bisection width of ________, and link cost _____________.
   - **a. 3; minimum(M, N); 2MN – (M+N)**
   - **b. 4; minimum(M, N); 2MN + (MN)**
   - **c. 2; maximum(M, N); 2MN – (M+N)**
   - **d. 2; minimum(M, N); 2MN – (M+N)**

13. If we use the schedule(static, 8) clause within the #pragma omp parallel for, we are enabling:
   - **a. Each thread is assigned 1/8th of the total iterations of the for loop in round-robin manner**
   - **b. Each thread is assigned 8 contiguous iterations of the for loop in round-robin manner**
   - **c. Each idle thread is dynamically assigned 1/8th of the remaining iterations of the for loop**
   - **d. Each idle thread is dynamically assigned the 8 leftmost contiguous remaining iterations of the for loop**

14. We would use the `lastprivate`() clause in OpenMP when we want the master thread:
   - **a. To copy the private copy of the variable from the thread that executed the last iteration**
   - **b. To copy the private copy of the variable from the thread that executed the first iteration**
   - **c. To copy the private copy of the variable from the thread that was created last**
   - **d. To copy the private copy of the variable from the thread that has the largest thread ID**

15. Assume a sequential program S has an execution time of 400 seconds. Now assume a parallel variant of S takes 55.55 seconds to complete when we have 8 processors available. The speedup approximately equal to:
   - **a. 50**
   - **b. 7.2**
   - **c. 0.9**
   - **d. 57.6**

Section B (50 marks – 5 questions)

Question No. 1: **[marks: 9, CLO: 2]**

Explain the following terms. You can draw a diagram if required to provide further detail.

a) Fat Tree:

b) Non-uniform Memory Access

c) Amdahl's law and its shortcomings

Shortcomings:

Assumes fixed problem size: Doesn't account for increased workload with more processors.

Ignores overheads: Like communication and synchronization.

Doesn't scale: Predicts diminishing returns with more processors.

Neglects memory hierarchy and data locality.

Question No. 2: **[marks: 8, CLO: 2]**

Write output for the following OpenMP-based piece of code. Assume the code does not have any syntax error and is written in the main function. If you think, there is some logical error then identify the error and propose a solution to correct it.

```
int k;
omp_set_num_threads(8);
int c = 10;
printf("total threads are:%d \n", omp_get_num_threads());

#pragmaomp parallel
{
      #pragmaomp master
      printf("total threads are:%d \n", omp_get_num_threads());

      #pragmaompfor schedule(static) firstprivate( c ) lastprivate( c )
      for (i = 1; i<= 16; i++) {
            c = c + i;
      }
}

printf("The value of counter is:%d \n", c);
```

Output:

```
int k;
omp_set_num_threads(8);
int c = 10;
printf("total threads are:%d \n", omp_get_num_threads());

#pragma omp parallel
{
  #pragma omp master
  printf("total threads are:%d \n", omp_get_num_threads());

  #pragma omp for schedule(static) firstprivate(c) lastprivate(c)
  for (i = 1; i<= 16; i++) {
    c = c + i;
  }
}
printf("The value of counter is:%d \n", c);
```

Question No. 3: **[marks: 10, CLO: 2]**

Parallelize following piece of code using OpenMP with a team of 8 threads.

Hint: Sometimes the parallelism is not inherent in original formulation.

```
int brr[1000];

int i;

brr[0]=5;

for (i = 1; i< 1000; ++i)

{

brr[i]=brr[i-1]+1;

}

printf("Sum at last index=%d\n", brr[999]);
```

Parallel Code:

```
#include <omp.h>
#include <stdio.h>

int main() {
    int brr[1000];
    int i;

    brr[0] = 5;

    #pragma omp parallel for
    for (i = 1; i < 1000; ++i) {
        brr[i] = i + 5;
    }

    printf("Sum at last index = %d\n", brr[999]);
    return 0;
}
```

Question No. 4: **[marks: 8, CLO: 2]**

Write output for following piece of code assuming that that there are 4 MPI processes.

[Assume there is no syntax error]

```
int main(intargc, char** argv) {

MPI_Init(&argc,&argv);

MPI_Statusstatus;

int p;

int i;

MPI_Comm_size(MPI_COMM_WORLD, &p);

int my_rank;

MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

int a = my_rank;

int b;

int sendTag=1;

int recvTag=1;

int next=(my_rank+1)%p; //determine my right node

int previous=((my_rank-1+p)%p); //determine my left node

MPI_Sendrecv(&a,1,MPI_INT,next,sendTag,

&b,1,MPI_INT,previous,recvTag, MPI_COMM_WORLD, &status );


printf(“I\’m %d: Received:%d from %d and Sent:%d to %d\n “,

my_rank ,b,previous, a,next);

MPI_Finalize();

}
```

Output:

```
I'm 0: Received:3 from 3 and Sent:0 to 1
I'm 1: Received:0 from 0 and Sent:1 to 2
I'm 2: Received:1 from 1 and Sent:2 to 3
I'm 3: Received:2 from 2 and Sent:3 to 0
```

Question No. 5: **[marks: 4+6+5, CLO: 3]**

a) **Briefly describe the following parameters to evaluate static interconnections:**

- **Diameter**

- **Bisection Width**

- **Arc connectivity**

- **Cost (No. of links)**

b) **Draw a 2-D mesh without wraparound having 16 nodes and calculate the values of above parameters by first mentioning or deriving the formulas for these parameters for a 2-D mesh without wraparound.**

c) **Suppose we have to perform one-to-all broadcast in this mesh where the source is node 0. How this operation could be performed using recursive doubling with minimum number of steps. Explain with details and mention the cost of this operation!**