National University of Computer and Emerging Sciences, Lahore Campus

Course:	Parallel and Distributed Computing	Course Code:	CS-3006
Program:	BS (Computer Science)	Semester:	Spring 2023
Duration:	N/A	Total Marks:	35
Submit Date:	06-Apr-2023	Weight	2.5%
Section:	BCS (6E-6F)	Page(s):	6
Exam:	Assignment 02	Roll No.	

Name & Section:

Submit assignment in Wednesday's class on 05 Apr 2023. Submission after that will not be accepted.
Attempt all questions on the assignment 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 # 1: [6+4 marks, CLO # 1] – Distributed Operating Systems

Provide a comparison of Network Operating System, Distributed Operating System, and Middleware. Briefly explain the goals of a middleware platform.

System	Description	Main Goal
DOS	Tightly-coupled operating system for multiprocessors and homogeneous multicomputers	Hide and manage hardware resources
NOS	Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middle-ware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

Goals of a middleware platform:

Resource sharing

The ability to access and share resources in a distributed environment

- The bread and butter of distributed systems

Transparency

- The ability to view a distributed system as if it were a single computer
- Varying dimensions of transparency incl. location, access, migration, etc.
- The degree of transparency is a key decision in any systems architecture

Openness

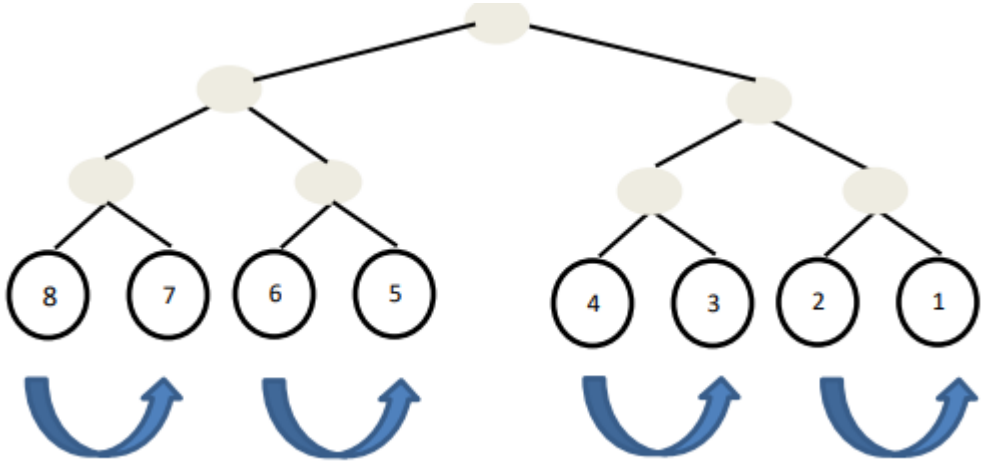
- The offering of services according to standard rules (syntax and semantics)
  - E.g. using IEEE standard
- Openness provides support for the key properties of portability and interoperability
- Again, the degree of openness is a key factor in systems design

Extensibility

- The ability to be able to introduce new or modified functionality

Question # 2: [2 + 2 + 2 + 2 + 2 marks, CLO # 3] – Prefix Sum

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 (8 – ID). Show the calculation at each step and provide the final value at each node at the end of the operation.



Stage 1:

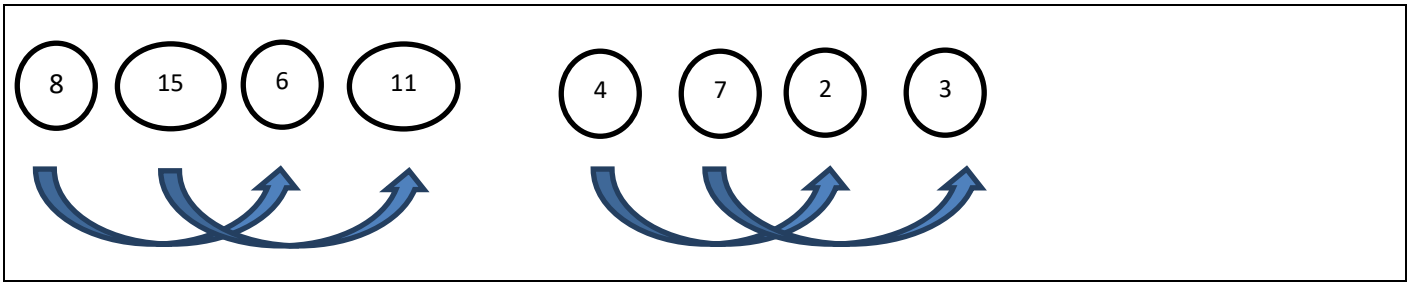
Node 0 (value: 8) and node 1 (value: 7) exchange with each other, after which node 0 has (value: 8, stored: 7, 8) and node 1 has (value: 15, stored: 7, 8). Reasoning: since value 8 was received by node 1 from node 0 (smaller rank), it will add this to its value (7+8).

Node 2 (value: 6) and node 3 (value: 5) exchange with each other, after which node 2 has (value: 6, stored: 5, 6) and node 3 has (value: 11, stored: 5, 6). Reasoning: since value 6 was received by node 3 from node 2 (smaller rank), it will add this to its value (5+6).

Node 4 (value: 4) and node 5 (value: 3) exchange with each other, after which node 4 has (value: 4, stored: 3, 4) and node 5 has (value: 7, stored: 3, 4). Reasoning: since value 4 was received by node 5 from node 4 (smaller rank), it will add this to its value (3+4).

Node 6 (value: 2) and node 7 (value: 1) exchange with each other, after which node 6 has (value: 2, stored: 1, 2) and node 7 has (value: 3, stored: 1, 2). Reasoning: since value 2 was received by node 7 from node 6 (smaller rank), it will add this to its value (1+2).

Stage 2:



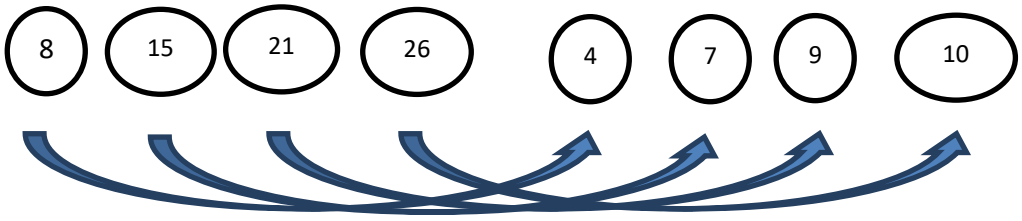
Node 0 (value: 8) and node 2 (value: 6) exchange with each other, after which node 0 has (value: 8, stored: 5, 6, 7, 8) and node 2 has (value: 21, stored: 5, 6, 7, 8). Reasoning: since (7, 8) was received by node 2 from node 0 (smaller rank), it will add this to its value (6+7+8).

Node 1 (value: 15) and node 3 (value: 11) exchange with each other, after which node 1 has (value: 15, stored: 5, 6, 7, 8) and node 3 has (value: 26, stored: 5, 6, 7, 8). Reasoning: since (7, 8) was received by node 3 from node 1 (smaller rank), it will add this to its value (5+6+7+8).

Node 4 (value: 4) and node 6 (value: 2) exchange with each other, after which node 4 has (value: 4, stored: 1, 2, 3, 4) and node 6 has (value: 9, stored: 1, 2, 3, 4). Reasoning: since (3, 4) was received by node 6 from node 4 (smaller rank), it will add this to its value (2+3+4).

Node 5 (value: 7) and node 7 (value: 3) exchange with each other, after which node 5 has (value: 7, stored: 1, 2, 3, 4) and node 7 has (value: 10, stored: 1, 2, 3, 4). Reasoning: since (3, 4) was received by node 7 from node 5 (smaller rank), it will add this to its value (1+2+3+4).

Stage 3:

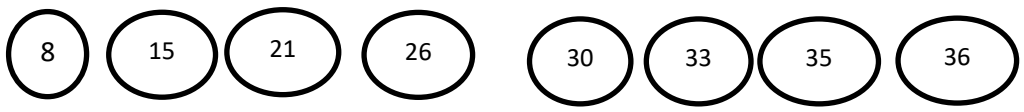


Node 0 (value: 8) and node 4 (value: 4) exchange with each other, after which node 0 has (value: 8, stored: 1 to 8) and node 4 has (value: 30, stored: 1 to 8). Reasoning: since (5, 6, 7, 8) was received by node 4 from node 0 (smaller rank), it will add this to its value (4+5+6+7+8).

Node 1 (value: 15) and node 5 (value: 7) exchange with each other, after which node 1 has (value: 15, stored: 1 to 8) and node 5 has (value: 33, stored: 1 to 8). Reasoning: since (5, 6, 7, 8) was received by node 5 from node 1 (smaller rank), it will add this to its value (3+4+5+6+7+8).

Node 2 (value: 21) and node 6 (value: 9) exchange with each other, after which node 2 has (value: 21, stored: 1 to 8) and node 6 has (value: 35, stored: 1 to 8). Reasoning: since (5, 6, 7, 8) was received by node 6 from node 2 (smaller rank), it will add this to its value (2+3+4+5+6+7+8).

Node 3 (value: 26) and node 7 (value: 10) exchange with each other, after which node 3 has (value: 26, stored: 1 to 8) and node 7 has (value: 36, stored: 1 to 8). Reasoning: since (5, 6, 7, 8) was received by node 7 from node 3 (smaller rank), it will add this to its value (1+2+3+4+5+6+7+8).



Question # 3: [8 + 6 + 1 marks, CLO # 2] – Message Passing Interface

(i) Write the output for the following piece of code assuming that that there are 4 MPI processes. Assume there is no syntax error.

```
int main(int argc, char** argv) {
      MPI_Init(&argc, &argv);
      MPI_Status status;
      int p, i, b;
      MPI_Comm_size(MPI_COMM_WORLD, &p);
      int my_rank;
      MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
      int a=my_rank;
      int sendTag=a;
      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:

```
Output (Solution):

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
```

(ii) Assume these are the contents of sbuf in the processes P0, P1 and P2:

P0	2	3	1	7	11	0
P1	5	2	5	1	7	11
P2	2	4	4	10	4	5

Show the contents of rbuf in the same processes after this MPI command:

```
MPI_Scan(sbuf, rbuf, 6, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
```

```
Hint: (MPI_Scan is used for prefix-sum).
```

Contents of rbuf:

P0	***2***	***3***	***1***	***7***	***11***	***0***
P1	***7***	***5***	***6***	***8***	***18***	***11***
P2	***9***	***9***	***10***	***18***	***22***	***16***

Rough work:

(iii) When we need the result of a reduction operation by all of the processes, which MPI operation will we use (just one)?

```
All_Reduce operation is the best suited operation.
```