Started on	Tuesday, 23 April 2024, 10:47 AM
State	Finished
Completed on	Tuesday, 23 April 2024, 11:06 AM
Time taken	19 mins 36 secs
Grade	**3.50** out of 4.00 (**87.5**%)
Feedback	It's important to study in time. Well done!

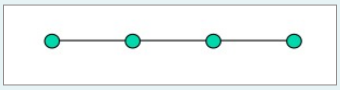
Question **1**

Correct

Mark 1.00 out of 1.00

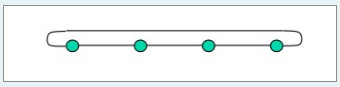
Nodes in a network can be connected in different ways. This applies also to processes, for which a logical neighborhood can be established.
We ask you to associate each name of a topology to its visual representation. Drag the correct figure besides its proper name.

Linear array



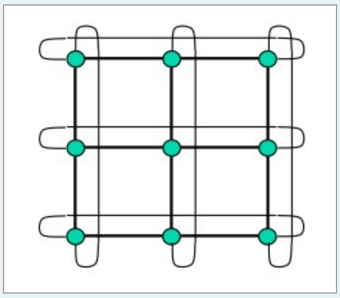
✓

Ring (also know as 1D torus)



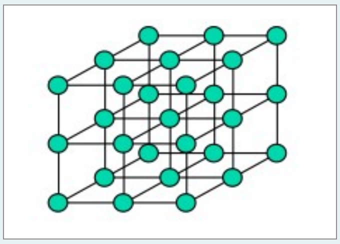
✓

2D torus



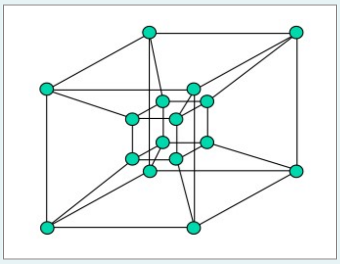
✓

3D mesh

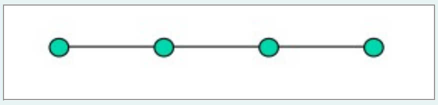


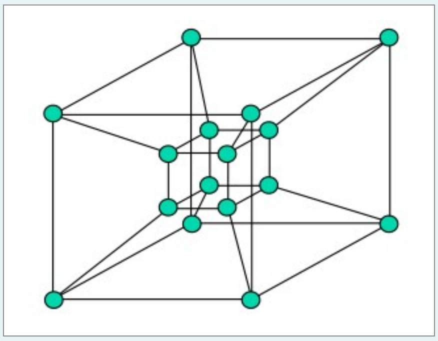
✓

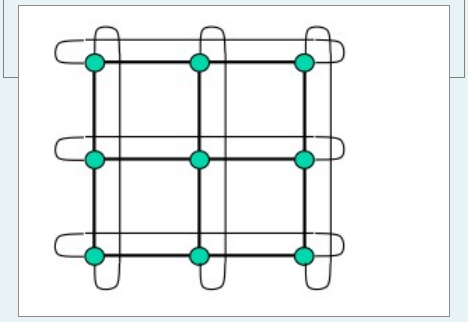
Hypercube

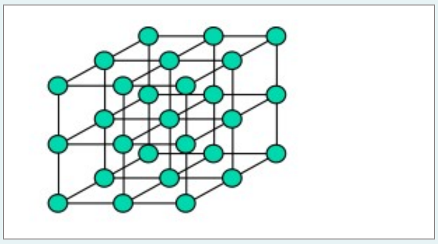


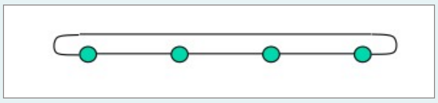
✓











Your answer is correct.

Question 2

Correct

Mark 0.50 out of 1.00

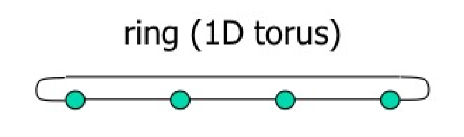
We want to establish a logical topology for a group of MPI processes.

Given the following code excerpt:

```
int              rank,  size,  next_nbr,  prev_nbr;
MPI_Status status;
...
MPI_Comm_rank( MPI_COMM_WORLD, &rank );
MPI_Comm_size ( MPI_COMM_WORLD, &size  );
...
next_nbr = rank + 1;
if (next_nbr >= size)
   next_nbr = ... ;
prev_nbr = rank - 1;
if (prev_nbr < 0)
   prev_nbr = ... ;
...
MPI_Irecv(..., ... , ..., prev_nbr, ..., MPI_COMM_WORLD, &status );
MPI_Send(..., ... , ..., next_nbr, ..., MPI_COMM_WORLD );
...
```

We define process with rank+1 as the next neighbor in the logical topology, the one to which we want to send a message. And process with rank-1 as the previous one, the one from which we want to receive some message.

Indicate how we can define variables next_nbr and prev_nbr so that we create a *Ring* (also known as *1D torus*) of MPI processes, where the first and last processes are logically connected to each other.



What should appear instead of ... ?

if (next_nbr >= size)
   next_nbr = ... ; [0 ◆] ✓

if (prev_nbr < 0)
   prev_nbr =   ... ; [size - 1 ◆] ✓

Your answer is correct.

The correct answer is:
if (next_nbr >= size)
   next_nbr = ... ; → 0,

if (prev_nbr < 0)
   prev_nbr =   ... ; → size - 1

Question 3

Correct

Mark 1.00 out of 1.00

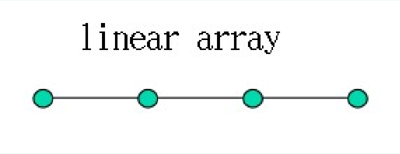
We want to establish a logical topology for a group of MPI processes.

Given the following code excerpt:

```
int              rank,  size,  next_nbr,  prev_nbr;
MPI_Status status;
...
MPI_Comm_rank( MPI_COMM_WORLD, &rank );
MPI_Comm_size ( MPI_COMM_WORLD, &size  );
...
next_nbr = rank + 1;
if (next_nbr >= size)
   next_nbr = ... ;
prev_nbr = rank - 1;
if (prev_nbr < 0)
   prev_nbr = ... ;
...
MPI_Irecv(..., ... , ..., prev_nbr, ..., MPI_COMM_WORLD, &status );
MPI_Send(..., ... , ..., next_nbr, ..., MPI_COMM_WORLD );
...
```

We define process with rank+1 as the next neighbor in the logical topology, the one to which we want to send a message. And process with rank-1 as the previous one, the one from which we want to receive some message.

Indicate how we can define variables next_nbr and prev_nbr so that we create a *Linear Array* of MPI processes, where the first and last processes are NOT logically connected to each other, i.e. the first process does not have a previous neighbor and the last process does not have a next neighbor.



What should appear instead of ... ?

if (next_nbr >= size)
   next_nbr = ... ; MPI_PROC_NULL ✓

if (prev_nbr < 0)
   prev_nbr = ... ; MPI_PROC_NULL ✓

Your answer is correct.

The correct answer is:

if (next_nbr >= size)
   next_nbr = ... ; → MPI_PROC_NULL,

if (prev_nbr < 0)
   prev_nbr = ... ; → MPI_PROC_NULL

Question 4

Correct

Mark 1.00 out of 1.00

Given a loop with **N** iterations, we want to distribute those iterations among **P** processes (or threads) in such a way such the load is well balanced when P does not divide N exactly. Assuming each process (or thread) has a unique identifier **Id**, being these identifiers in the range from 0 to P-1, give an expression to compute the number of iterations for each process (or thread). Beware that all of N, P and Id are integers. Do not leave blank spaces within the expression.

Answer: (N/P)+(N%P>ID) ✓

Well done!

Note that ”/” is an integer division; and % is evaluated before the comparison; thus, the result of the
modulus ”%” operation is compared > against the identifier. The comparison returns a 0 for FALSE and a 1 for TRUE. Thus,
(N % P > Id) will add one iteration to the initial N % P processes (or threads) and none to the rest (i.e. those with an Id larger than or equal to the result of N%P will not get an extra iteration).

The correct answer is: (N/P)+(N%P>Id)