Q 1. Give an embedding of 4*6 2D mesh onto hypercube using gray codes (8 points). Argue why this yields a dilation-1 embedding

	00	01	11	10
000	00000 (0)	00001 (1)	00011 (3)	00010 (2)
001	00100 (4)	00101 (5)	00111 (7)	00110 (6)
011	01100 (12)	01101 (13)	01111 (15)	01110 (14)
010	01000 (8)	01001 (9)	01011 (11)	01010 (10)
110	11000 (24)	11001 (25)	11011 (27)	11010 (26)
111	11100 (28)	11101 (29)	11111 (31)	11110 (30)

Immediate neighbors of in the mesh are mapped to hypercube nodes whose label differ in exactly one bit position. Therefore, this mapping has dilation of one.

Q.2. State four decomposition techniques and illustrate them with an example problem/application

a) recursive = quicksort
b) speculative = discrete event simulation
c) exploratory = puzzle game
d) data decomposition = matrix multiplication

Q.3. State the basic steps of an all-to-all broadcast algorithm for a hypercube using log p steps using log p ts + mtw(p-1). Justify its time complexity. Why this is an extension of ring algorithm. Ans.

On a p node hypercube, size of each message exchanged in the ith of the log p steps is 2^(i-1)m

A pair of nodes take time (ts + 2^(i-1)tw.m) ----- expression 1 to send and receive messages from each other in the ith step.

If we sum expression 1 from 1 to log p,

we get total time as ts log p + tw.m(p-1)

Why this is an extension of ring algorithm?

Ans. The ring algorithm is to rotate the messages in the ring with processors picking up what they need. Same algorithm was employed in 2D mesh with ring algorithm along the rows and them columns (i.e., in each dimension). The hypercube is a 2x2x2… mesh. Thus, the exchange across a dimension is a ring algorithm with a ring of size 2.

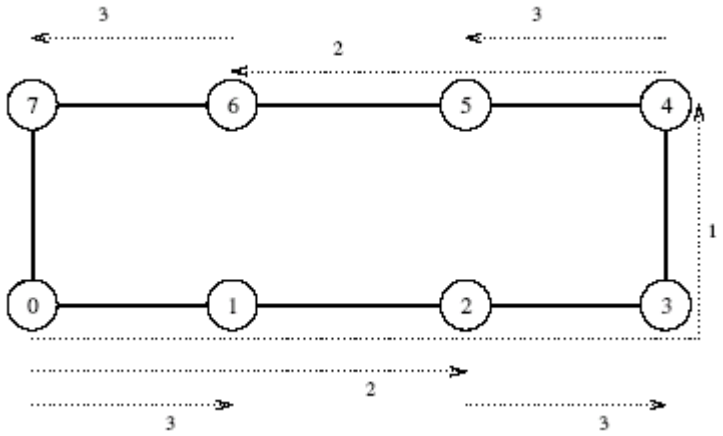
OR part

One to all broadcast in ring.

This can be done using recursive doubling: source sends a message to a selected processor. We now have two independent problems defined over halves of machines.

It takes ts + log p tw. m since using recursive doubling, log p iterations are required for sending the message m such that all the p nodes receive m

Here is an example:



One-to-all broadcast on an eight-node ring. Node 0 is the source of the broadcast. Each message transfer step is shown by a numbered, dotted arrow from the source of the message to its destination. The number on an arrow indicates the time step during which the message is transferred.

Cut through routing is critical for this algorithm to work in log p steps, with each step taking constant time, with communication pattern similar to hypercube which happen to be distant nodes in a ring.

Q. 4. Programming Question.

Q.5. Prove that there cannot be a deadlock if resources are linearly ordered and processes always acquire them in order.

Proof: If resources are ordered linearly we need to get resources in that order.

Let resources order be

R1, R2, .., Rn

We prove by contradiction that there can not be a deadlock if the resources are acquired in order.

Suppose there is a deadlock. P1 is holding i and P2 is holding j. Deadlock

happens when a processor P1 is waiting for resource j while another process P2 is waiting for resource i.

But, resources have to be acquired in order. So, if P1 has i and waits for j then i<j
If P2 has j and waits for i then j<i.

Both i<j and j<i can not be true.

So, there is a contradiction.

Another way: Let us assume that the rules are followed and yet there is a deadlock. Therefore, there must be a cyclic wait. Let us assume that the processor Pi, holding the highest indexed resource Rk, is waiting on processor Pj. However, Pj could not be holding a resource greater than Rk (by assumption). Therefore, Pi is trying to access a resource with index less than k, a contradiction. Thus, Pi can finish, and eventually will release all its resources.