4.1

- 4.1) Add an extra iteration to the loop, i.e., from *d* instead of *d-1*. Add extra tests to check if the nodes you communicate to and from are < N, where N is your number of processes. What we do here is to consider the smallest hypercube to contain all N processes, although some nodes will be missing. In the algorithm, reverse the loop from 0 to d and see that we will follow the natural construction of a hypercube, which gives a natural proof of why the modification is correct.

4.3

- Recall: Standard algorithms for all-to-all broadcast on a ring and a hypercube give the following times:
  - $T_{ring}=(t_s+t_w m)(p-1)$ and $T_{hcube}=t_s \log p + t_w m(p-1)$
  - with the assumption that there is no congestion. If ts=100tw we have $T_{ring}=(100+m)(p-1)t_w$ and $T_{hcube}=(100 \log p + m(p-1))t_w$.
- On a ring, the standard algorithm gives $T_a=T_{ring}$ and the hypercube algorithm suffers from congestion (ratio of bisection width=p/4, we take dimension≥2) for all communications except the one that corresponds to the dimension of the ring. We have a correction factor $f=p(\log p-1)/4 \log p$. For the hypercube we have $T_b=fT_{hcube}$.
- If we look at the ratio T_a/T_b we see that for large messages the ring algorithm is better and for small messages, *depending on p*, the hypercube algorithm may be better.

$$\frac{T_a}{T_b}=\frac{100(p-1)+m(p-1)*4}{(100\log p+m(p-1))*p}$$

4.5

- In the first iteration, the following pairs of processors exchange their data: (0,4), (1,5), (2,6), and (3,7). This step takes time $t_s+4t_w m$ because 4 messages of size m pass through the root of the tree. After this step each processor has data of size 2m. Now the following pairs of processors exchange their data: (0,2), (1,3), (4,6), and (5,7). The size of each message is now 2m and two messages traverse the channels in the same direction. This step takes also time $t_s+4t_w m$. There are 3 steps here (8 processors).
- In general there are $\log p$ steps, each taking time $t_s+t_w mp/2$. Note that we need this particular ordering of messages to limit congestion and keep the communication cost constant for every step.