Question 7. RAID Regained [8 MARKS]

You have a RAID-4 device (parity-based RAID + a single parity disk) with 5 disks and a 4KB chunk size, as shown below:

<table>
<thead>
<tr>
<th>Disk-0</th>
<th>Disk-1</th>
<th>Disk-2</th>
<th>Disk-3</th>
<th>Disk-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>block0</td>
<td>block1</td>
<td>block2</td>
<td>block3</td>
<td>parity(0..3)</td>
</tr>
<tr>
<td>block4</td>
<td>block5</td>
<td>block6</td>
<td>block7</td>
<td>parity(4..7)</td>
</tr>
</tbody>
</table>

You are not sure that writes to the RAID device are working correctly. You write the following skeleton code for writing to the RAID device. Fill in the rest of the code in the empty boxes to ensure that writes work correctly.

```c
void write(int block, char *data) {
    char buf[4096];
    char parity[4096];
    int disk = \[\];
    int phy_block_nr = \[\];

    \[disk, phy_block_nr, \];
    read(4, phy_block_nr, parity);
    xor(\[\], \[\], \[\], parity);
    xor(parity, data, parity);
    \[disk, phy_block_nr, \];
    write(4, phy_block_nr, parity);
}
```
Question 7. RAID Regained [8 MARKS]

You have a RAID-4 device (parity-based RAID + a single parity disk) with 5 disks and a 4KB chunk size, as shown below:

<table>
<thead>
<tr>
<th>Disk-0</th>
<th>Disk-1</th>
<th>Disk-2</th>
<th>Disk-3</th>
<th>Disk-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>block0</td>
<td>block1</td>
<td>block2</td>
<td>block3</td>
<td>parity(0..3)</td>
</tr>
<tr>
<td>block4</td>
<td>block5</td>
<td>block6</td>
<td>block7</td>
<td>parity(4..7)</td>
</tr>
</tbody>
</table>

You are not sure that writes to the RAID device are working correctly. You write the following skeleton code for writing to the RAID device. Fill in the rest of the code in the empty boxes to ensure that writes work correctly.

```c
/*
 * This write() routine takes a logical block number (block), as shown above, 
 * and writes 4KB (data) to the device.
 *
 * It uses the underlying primitives:
 * read(int disk, int block_nr, char *data)
 * write(int disk, int block_nr, char *data)
 * xor(char *d1, char *d2, char *d3) ;; xor d1 and d2, place result in d3
 */

void write(int block, char *data)
{
    char buf[4096];
    char parity[4096];
    int disk = block % 4;  
    int phy_block_nr = block / 4;

    read(disk, phy_block_nr, buf);  
    read(4, phy_block_nr, parity);  
    xor(parity, buf, parity);  
    xor(parity, data, parity);  
    write(disk, phy_block_nr, data);  
    write(4, phy_block_nr, parity);
}
```