5. *Fork (5 points)*  Consider this code using Linux's *fork*:

```c
int x = 7;
if(fork()) {
   x++;
   printf(" %d ", x);
   fork();
   x++;
   printf(" %d ", x);
} else {
   printf(" %d ", x);
}
```

What are *all* the different possible outputs (order of things printed) for this code? *(Hint: There are four of them.)*

1) 7 8 9 9
2) 8 7 9 9
3) 8 9 7 9
4) 8 9 9 7

The two 9's must come after the 8, and the 7 from the first child process can print anywhere between these.

**Note:** If you actually try this in C, you may see five numbers printed, which is rather surprising. The issue is the implementation of *printf* may buffer output and the second *fork* call in the code then copies the not-empty output buffer into the child process. You can fix this in the code by putting *fflush(stdout)*; after the first call to *printf*. In terms of the exam, the four outputs above *are all still possible* and of course we didn't expect you to also list the fifth possible output that arises from this buffered-output issue.
Exercises

1) What are three specific benefits of using virtual memory?

Bridges memory and disk in memory hierarchy.
Simulates full address space for each process.
Enforces protection between processes.

2) Fill in the following formulas below.

Page offset bits = $\log_2($page size in bytes$)$

Virtual address bits = virtual page number bits + page offset bits

Physical address bits = physical page number bits + page offset bits

Virtual page number bits = $\log_2($number of virtual pages$)$

Entries in a page table = number of virtual pages

3) Fill in the following table:

<table>
<thead>
<tr>
<th>VA width $(n)$</th>
<th>PA width $(m)$</th>
<th>Page size (P)</th>
<th>VPN width</th>
<th>PPN width</th>
<th>Bits in PTE (assume V, R, W, X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>32</td>
<td>16KiB</td>
<td>18</td>
<td>18</td>
<td>22</td>
</tr>
<tr>
<td>32</td>
<td>26</td>
<td>8KiB</td>
<td>19</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>36</td>
<td>32</td>
<td>32KiB</td>
<td>21</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>40</td>
<td>36</td>
<td>32KiB</td>
<td>25</td>
<td>21</td>
<td>25</td>
</tr>
<tr>
<td>64</td>
<td>40</td>
<td>64KiB</td>
<td>48</td>
<td>24</td>
<td>28</td>
</tr>
</tbody>
</table>