1.) Fill in the blanks to make true statements, for a 64-bit Linux machine (like NetRun's default).

   • A function's return value is stored ________________________________.
   • A function's first parameter is stored ________________________________.
   • A class with one long uses ________ bytes of memory.
   • You would write a C++ prototype for an assembly function named “bar” with no parameters or return value as:

(Score: 10 points.)

2.) Convert each piece of C++ code to assembly, and vice versa:

<table>
<thead>
<tr>
<th>C++</th>
<th>Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>grabit:</code></td>
<td>`mov rax, 5</td>
</tr>
<tr>
<td><code>    ret</code></td>
<td></td>
</tr>
</tbody>
</table>

// The functions phase1 and phase2  
// both take and return one long. 
long pipeline(long x) {  
    return phase2(phase1(x)+3); 
}

(Score: 30 points. Each piece of code is separate.)

3.) There are several memory allocation styles. Fill in the remaining table entries.

<table>
<thead>
<tr>
<th>What?</th>
<th>Why?</th>
<th>How? (allocate one long)</th>
<th>Static or Dynamic?</th>
</tr>
</thead>
<tbody>
<tr>
<td>section .data</td>
<td></td>
<td>dq 3</td>
<td></td>
</tr>
<tr>
<td>the stack</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>malloc</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Score: 10 points. “Static” allocation happens once per program; “Dynamic” can happen again and again.)
4.) Fix the errors in this assembly translation of this C++:

```c++
for (int i=0; i<n; i++) arr[i]=3;
```

; input: rsi == arr, rdi == n

```assembly
mov rcx, 0 ; i
start:
mov DWORD[rsi + rcx], 3
add rcx, 1
cmp rcx, rdi
jle start
```

(Score: 15 points.)

5.) You see the compiler uses “push rbx” at the start of your function, and “pop rbx” at the end. Why?

(Score: 10 points.)

6.) If each piece of code returns a value, write the value. If it'll crash or hang, write that, and why.

<table>
<thead>
<tr>
<th>push rax</th>
<th>push 2</th>
<th>mov rax, 9</th>
<th>push 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>push 3</td>
<td>push 7</td>
<td>push rax, QWORD[rsp]</td>
<td>mov rax, 0x3F</td>
</tr>
<tr>
<td>pop rax</td>
<td>pop rax</td>
<td>sub rsp, 8</td>
<td>and rax, 0x55</td>
</tr>
<tr>
<td>ret</td>
<td>pop rcx</td>
<td>ret</td>
<td>ret</td>
</tr>
</tbody>
</table>

(Score: 15 points. Several of these are ... subtle. Be careful.)

7.) You had asked your newest employee to measure the speed of an empty function on Intel's new server chip, and his entire email in response was “503,148.7 per second”. What's wrong with his response?

(Score: 10 points.)