Quantitative Reasoning 20
Practice midterm exam

Suggested answers: These are not the only answers accepted, but should give a clear idea of what we had in mind.

1. This is, in ANT, the problem we have run into several times in a Python "while" loop. The value in the while clause is never changed so we go into an infinite loop. In this case, we need to decrement the count of the number of inputs still to go, in r4, so that the "branch on greater than" instruction will eventually fail and the output and halt can be executed in sequence. The way to do this is to insert the instruction
   \texttt{inc r4
-1}
just in front of the bgt instruction.

2a. The point is that the ASCII codes for the letters follow in numerical sequence according to the alphabetic sequence of the letter. For example, the code for "b" is 1 plus that for "a" and for "C" is 2 plus that of "A". Going from the decimal value of the code to hexadecimal can be done by the "change making" algorithm, first pulling out units of 16 for the high order digit and then having the remainder as the low order digit. Remember to count \ldots 8,9,a,b,c\ldots for the units position. We then have the table,

<table>
<thead>
<tr>
<th>letter</th>
<th>decimal code</th>
<th>hexadecimal code</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;b&quot;</td>
<td>98</td>
<td>62 (6 times 16=96)</td>
</tr>
<tr>
<td>&quot;C&quot;</td>
<td>67</td>
<td>43 (4 times 16=64)</td>
</tr>
<tr>
<td>&quot;Q&quot;</td>
<td>81</td>
<td>51 (Q is the 16th letter)</td>
</tr>
</tbody>
</table>

2b. First, the unsigned and signed binary equivalents are the same. Whether the number is signed or unsigned depends on which machine instructions are interpreting it, eg. ANT branches assume unsigned, inc, add etc., assume signed. So, conversion to binary just involves converting each hex digit to its 4 bit binary. HOWEVER, conversion to decimal does differ depending on whether or not the value is assumed to be signed or unsigned. To find the signed decimal of a negative value, take the 2s complement, replace 0 by 1 and 1 by 0 and add 1. Both equivalents are given.

<table>
<thead>
<tr>
<th>Hex given</th>
<th>binary</th>
<th>decimal (unsigned value)</th>
<th>decimal (signed value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xff</td>
<td>1111111</td>
<td>255 15x16+15</td>
<td>-1 negation= 00000001</td>
</tr>
<tr>
<td>0x32</td>
<td>00110010</td>
<td>50 3x16+2</td>
<td>50</td>
</tr>
<tr>
<td>0x84</td>
<td>10000100</td>
<td>132 8x16+4</td>
<td>-124 negation= 01111100</td>
</tr>
</tbody>
</table>

So - of 64+32+16+8+4=124

3. The syntax errors are:
   a. the colon, instead of a comma, separating the arguments to the range function
   b. the use of then instead of a colon in the two if commands

The logical errors are that the < and > in the if commands are reversed from the correct direction. The program as listed would have ihigh containing the position of the smallest value and ilow containing the position of the largest value. Of course, you could correct this by just reversing the order of ihigh and ilow in the return command (leaving a very strange and non-mnemonic pair of memory names).
4a.
def palindrome(s):
    #we will compare each char in s counting up from position 0
    #with the one counting down from len(s)-1, the last position
    #if ever we find a difference, we can return False
    #if all match, we can return True
    for i in range(len(s)//2):  #we don't have to go past midway
        if s[i] != s[len(s)-1-i]:
            return False
    return True  #we tried them all with no mismatch

4b.
def justpalindromes(slist):
    #build up a new list in name output for pals only
    output=[]
    for s in slist:  #for every string
        if palindrome(s):  #is it a pal?
            output=output+[s]  #if so, add it to output
    return output

4c.
def longestpal(slist):
    #we will start longest with "" and try to find a longer pal
    longest=""
    for s in slist:
        if palindrome(s) and len(s) > len(longest):  #is a pal and longer
            longest=s  #than our previous best, improve the result
    return longest  #when we leave the for loop we have compared them all