1. Truth Table

- Using “and”, “or”, “not, “A”, and “B” as many times as you want, provide a logical expression for this truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
2. Binary Math

a. What is the decimal number 113 in binary?
b. What is 10110010 in decimal?
c. Add 10110010 and 00111010 in binary; show the carries.

Example Gates

• These two gates define an 8-bit adding gate.
  • addc(a,b,c) takes three bits and returns two bits (the sum and the carry).
  • addbyte(x,y) takes two 8-bit numbers and produces the 8-bit sum (throwing away the highest bit), using 8 copies of “addc”.
addc and addbyte

```python
def addc(a, b, c):
    bit = (a and not b and not c) or (not a and b and not c) or (not a and not b and c) or (a and b and c)
    carry = (a and b and not c) or (a and not b and c) or (not a and b and c) or (not a and not b and c) or (a and b and c)
    return ([carry, bit])

def addbyte(x, y):
    z = [0] * 8
    sum7 = addc(x[7], y[7], 0)
    z[7] = sum7[1]
    sum6 = addc(x[6], y[6], sum7[0])
    sum5 = addc(x[5], y[5], sum6[0])
    sum4 = addc(x[4], y[4], sum5[0])
    sum3 = addc(x[3], y[3], sum4[0])
    sum2 = addc(x[2], y[2], sum3[0])
    sum1 = addc(x[1], y[1], sum2[0])
    z[1] = sum1[1]
    sum0 = addc(x[0], y[0], sum1[0])
    z[0] = sum0[1]
    return z
```

3. Gate Counting

a. To create a copy of the “addc” gate, how many “and”, “or”, and “not” gates do you need?

b. To create a copy of the “addbyte” gate using the “addc” gate, how many “and”, “or”, and “not” gates do you need?
Example ML$^3$ Program

- Initially, PC = 0 and acc = 0.

<table>
<thead>
<tr>
<th>Memory Location</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ACC = 0</td>
</tr>
<tr>
<td>76</td>
<td>01001100 ACC = ACC + mem[12]</td>
</tr>
<tr>
<td>138</td>
<td>10001010 if ACC == 0: PC = 10</td>
</tr>
<tr>
<td>0</td>
<td>00000000 ACC = 0</td>
</tr>
<tr>
<td>77</td>
<td>01001101 ACC = ACC + mem[13]</td>
</tr>
<tr>
<td>138</td>
<td>10001010 if ACC == 0: PC = 10</td>
</tr>
<tr>
<td>1</td>
<td>00000001 ACC = 1</td>
</tr>
<tr>
<td>206</td>
<td>11001110 mem[14] = ACC</td>
</tr>
<tr>
<td>0</td>
<td>00000000 ACC = 0</td>
</tr>
<tr>
<td>139</td>
<td>10001011 if ACC == 0: PC = 11</td>
</tr>
<tr>
<td>206</td>
<td>11001110 mem[14] = ACC</td>
</tr>
<tr>
<td>139</td>
<td>10001011 if ACC == 0: PC = 11</td>
</tr>
<tr>
<td>1</td>
<td>00000001 ACC = 1</td>
</tr>
<tr>
<td>0</td>
<td>00000000 ACC = 0</td>
</tr>
<tr>
<td>14</td>
<td>00000010 ACC = 2</td>
</tr>
</tbody>
</table>

4. ML$^3$

- What number will be stored in memory location 14 when the program on the previous page is run (“cycle” is hit about 10 times)?

- The python code for this example is available at: [http://www.cs.rutgers.edu/~mlittman/courses/cs442-06/python/ml3b.py](http://www.cs.rutgers.edu/~mlittman/courses/cs442-06/python/ml3b.py)
5. Lyrics

- Choose a song and write a Python program to output its lyrics. Again, be creative and try to use subroutines appropriately.

Extra Credit

- The remaining problems require a bit more thought!
- Try these for fun.
Lock Circuit

6. Combination Lock

- This circuit takes five bits of input and produces one bit of output.
- Imagine that we connect the input to some push buttons.
- Also imagine that we connect the output to an electronic lock that unlocks when the output is one.
- What’s the combination that unlocks the lock?
7. Name That Program

- The program in Question 4 looks at the values in memory locations 12 and 13 and stores something in memory location 14. Assuming mem[12] and mem[13] are zero or one, what function does the program compute?

3-sock Sorting

- The subroutine “sorter5” grabs 3 socks from the basket, looks for a pair among the 3, then returns whatever is left to the basket.

```python
def sorter5():
    x = getSock()
    y = getSock()
    if emptyBasket():
        match(x, y)
    else:
        z = getSock()
        if match(x, z):
            replaceSock(y)
        elif match(y, z):
            replaceSock(x)
        elif match(x, y):
            replaceSock(z)
        else:
            replaceSock(x)
            replaceSock(y)
            replaceSock(z)
```
8. sorter5 Analysis

a. Do you think sorter5 will do better or worse than sorter2 from class? Why?

b. Run the python program at http://www.cs.rutgers.edu/~mlittman/courses/cs442-06/python/socks.py. What is the max/min/mean reported for sorter5?

c. Does it beat sorter2?