CSCI 145 Quiz 8 practice questions on representing values and data **ANSWER KEY**

1. Convert each number. The binary numbers are unsigned: every place is positive, as in Lecture 15. No calculator needed.

`11010` in decimal	**26**	52 in binary	**110100**
`10010011` in decimal	**147**	83 in binary	**1010011**

2. Add these two unsigned binary numbers. Write each carry in the box above its column, and the sum in the boxes below.

carries	**1**	**1**	**1**		
		1	1	1	0
+		0	0	1	1
sum	**1**	**0**	**0**	**0**	**1**

3. Answer each question with a number. All the numbers are unsigned: 0 and up.

a) How many different patterns can 6 bits hold? **64**

b) What is the largest unsigned whole number 6 bits can hold? **63**

c) What is the fewest bits that can hold every unsigned whole number from 0 to 300? **9**

4. Convert each unsigned number. Use the hex table on the reference sheet.

7D (hex) in decimal	**125**	`10101100` in hex	**AC**
E3 (hex) in decimal	**227**	5E (hex) in binary	**01011110**

5. Colors.

a) Write #2E8B57 as a (red, green, blue) tuple.

`( 46 , 139 , 87 )`

b) Write `(255, 20, 147)` as a hex color code.

`# FF1493`

6. Write the value Python gives for each expression. Put quotes around a string.

`bin(12)`	**'0b1100'**	`hex(60)`	**'0x3c'**
`int("11100", 2)`	**28**	`int("B4", 16)`	**180**
`0x2C`	**44**	`0b101 * 2`	**10**

7. Each bit sequence is a signed 8-bit two's complement number. Write it in decimal.

`11101100` **-20** `01011010` **90** `10000110` **-122**

8. Write each number in signed 8-bit two's complement. Show your work.

a) `-58`

```
58:     00111010
flip:   11000101
add 1:  11000110
```

b) `-7`

```
7:      00000111
flip:   11111000
add 1:  11111001
```

9. Which number can't be stored in signed 8-bit two's complement?

A. 127 **B. 128** C. -128 D. -1

10. Write True or False for each.

a) `0.25 + 0.125 == 0.375` **True**

b) `0.1 + 0.7 == 0.8` **False**

c) A float can store every number between 0 and 1 exactly. **False**

d) A 64-bit float can store a number bigger than 2^{64}. **True**

11. What happens when this code runs?

```
fuel = 0.5
while fuel != 0:
    fuel = fuel - 0.1
print("empty")
```

A. It prints `empty` after 5 passes through the loop

B. It prints `empty` after 6 passes through the loop

C. It never stops

D. It never runs the loop body

12. Write the value Python gives for each expression. Put quotes around a string. Use the ASCII table on the reference sheet.

`ord("K")`	**75**	`chr(100)`	**'d'**
`chr(ord("b") + 3)`	**'e'**	`ord("7") - ord("0")`	**7**
`"Dog" < "cat"`	**True**	`chr(0x7A)`	**'z'**

13. These hex bytes are ASCII codes. What word do they spell?

4A 61 7A 7A **"Jazz"**

14. Write a function `count_ones(num)` that returns how many 1s there are when the whole number `num` is written in binary. Don't use `bin`. If `num` is 0, it returns 0. Otherwise, it returns the number of 1 bits. Do not include a docstring. For example:

```
>>> count_ones(25)   # 11001
3
>>> count_ones(8)    # 1000
1
>>> count_ones(0)
0
```

```
def count_ones(num):
    count = 0
    while num > 0:
        count = count + num % 2
        num = num // 2
    return count
```