Key: SA = short answer, CR = code reading, FR = free response, CW = code writing

Note 1: any topic listed at CW or FR rank may be tested at all ranks.
Note 2: any topic listed at the CR rank may also be tested at the SA rank.

Algorithms and Abstraction (SA, FR)

1. Give a high level algorithm for printing out a list in sorted order.
   ANS: Find the smallest item of the list and print it. Then discard it from the list. Continue until the list is empty.

2. Give a high level algorithm for returning the sum of every other element in a list.
   ANS: Print one item of list and discard it, only discard the next, and keep going until the list is empty.

Programming Basics (CW)

1. Write a Python program that prints an item and its corresponding type.
   ANS:
```
def fn(item):
    print(item)
    if type(item)==str:
        print('str')
    elif type(item)==int:
        print('int')
    elif type(item)==bool:
        print('bool')
```

2. Output the results of the following statements:
   a. `float(int(32.58))` ANS: `32.0`
   b. `type(7//2)` ANS: `int`
   c. `type("01151")` ANS: `string`

How Python Works (CR)

1. What is the job of the interpreter?
   ANS: The job of the interpreter is to translate your python code into bytecode, which the computer can then run.

2. What type of error is each of the following?
   a.
```
x = 5
x = x + y
```
   ANS: Name Error: used a missing variable (runtime error)

b.
```
if x = 2:
  print ("Hello")
```
ANS: Syntax error (x==2)

c.
```
x = 0
x = x + "no"
```
ANS: Type error (adding string to a number)

Functions (FR, CW, SA)

1. If we have the the following function:
   ```
   def summation(a,b):
        print(a+b)

   c = summation(2,4)
   ```
   What will c be equal to after we call this function? If there is an error, fix and explain it.

   ANS: C will be equal to None since summation returns None. We have to change the print to return a + b since printing wont return a value.

2. What does the following function returns?
   ```
   def f(x):
          x + 42

   print(f(5))
   ```
   ANS: `None`

Data Representation (SA, FR)

1. If we only had 5 bits to use, what is the minimum and maximum number we can represent using 5 bits?
   ANS: Minimum: 0, Maximum: 31

2. Convert the following decimal numbers into their binary representation using only 4 bits. If there aren't enough bits then only represent the lower 4 bits: 0, 17, 23, 5, 8, 2.

   ANS:
   ```
   0: 0000
   17: 0001
   23: 0111
   5: 0101
   8: 1000
   2: 0010
   ```

3. Explain the difference in the simple approach and actual approach in the binary representation of negative numbers.
   ANS: Simple Approach: reserve one bit to represent whether the number is positive or negative. Convert the rest normally. Actual Approach: use a bit to represent whether it's positive or negative, but flip the rest of the bits, to avoid double-representing zero.

Booleans and Conditionals (CW)

1.
```
def f(x, y, z):
      result = ""
      if (x + y) % 2 == 0:
            result += str(x)
      if (y + z) % 2 == 1:
            result = str(y) + result
      if z % 4 == 3:
            result = ""
      return result

print(f(1, -7, 526), f(8, 43, 2), f(9, 101, 11))
```

   ANS: `-71 43`

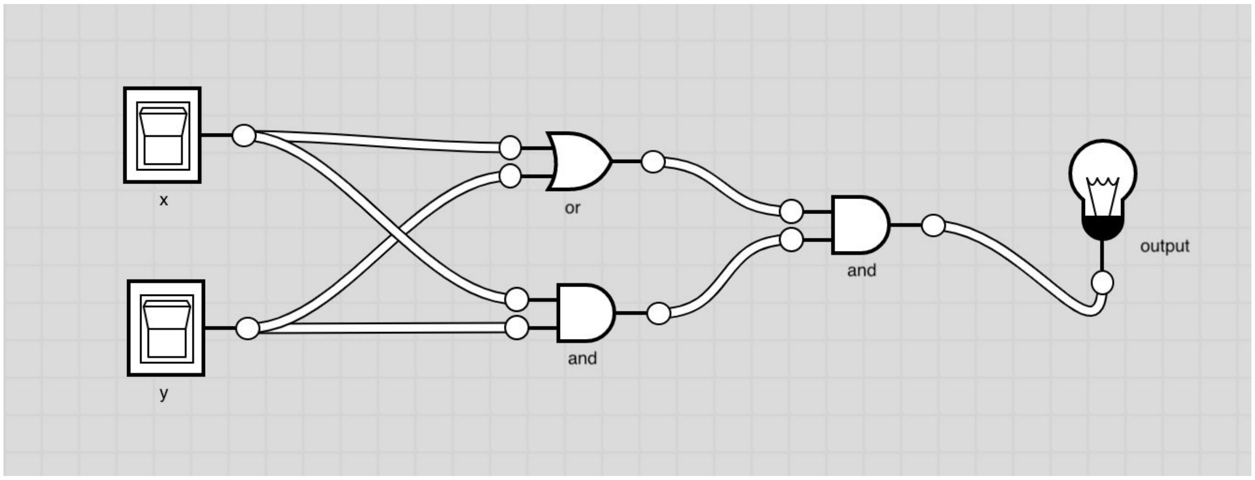
2. Write a function to determine whether somebody should eat ice cream on a hot day based on temp (must be greater than 60 degrees) and hunger (must be greater than 0.5)
   ANS:
```
def iceCream(temp, hunger):
      if temp > 60:
          if hunger > 0.5:
              return True
      return False
```

3. What is the difference between the "and" vs. "or" operations in terms of their relationship with the boolean True?
   ANS: "and" evaluates to True only when both values are True, while "or" evaluates to True when either value is True

Circuits and Gates (FR, SA)

1. How does a half adder work? How does a full adder work? What are the differences?
   ANS: A half adder takes in two 1-bit inputs and adds them to give two outputs: sum and carry out. A full adder takes in 3 1-bit inputs, a, b, carry-in and also has 2 outputs: sum and carry out. A full adder can be chained together to make a multi-bit adder since it has a carry in and carry out.

2. What boolean operation does the following logic circuit behave like?



ANS: `AND`

3. What is the purpose of C_in and C_out in a full adder?
ANS: To carry an additional value while working with multi digit numbers

While Loops (CW, FR)

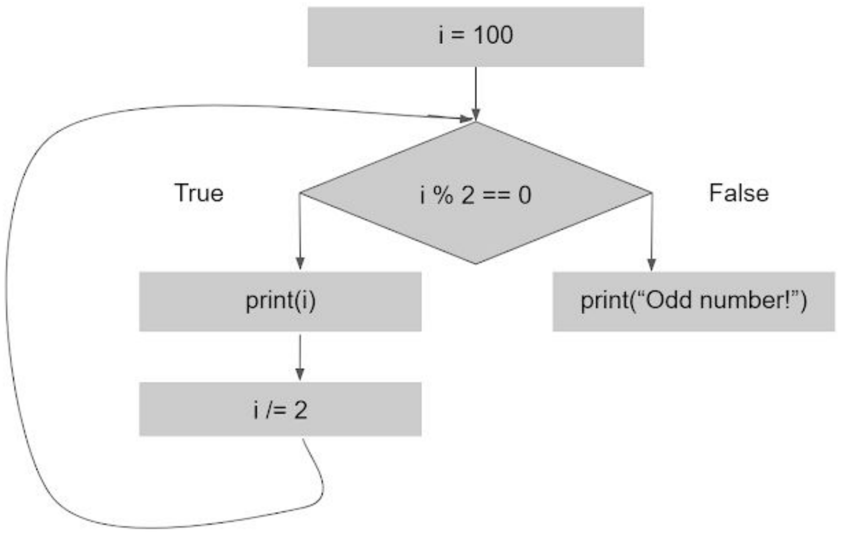
1. Write the function `createTriangle(n)` to recreate the following pattern with a while loop given n number of rows.

```
print(createTriangle(3))
*
**
*
```

ANS:

```
def createTriangle(n):
      half = n//2 + 1
      i = 0
      while i < half:
            s1 = "*" * (i+1)
            print(s1)
            i += 1
      #now i = half, reduce by 1
      i -= 1
      while i > 0:
            s2 = "*" * (i)
            print(s2)
            i -= 1
```

2. Write the while loop that corresponds with this flow chart.



ANS:

```
i = 100
while i % 2 == 0:
    print(i)
    i /= 2
print("Odd number!")
```

3. Use while loop to write function `hasConsecutiveDigits(n)` that takes in a possibly-negative int value n and returns True if that number contains two consecutive digits that are the same, False otherwise.

ANS:

```
def hasConsecutiveDigits(n):
    n = abs(n)
        prevDigit = -1
        while (n>0):
            onesDigit = n%10
            n//=10
            if (prevDigit == onesDigit):
                return True
            prevDigit = onesDigit
        return False
```

4. Write the function `isPowerFour(n)` that takes in a number n and returns True if n is a power of 4, returns False otherwise.

   ANS:
```
def isPowerFour(n):
        x = -1
        while ((4**x) <= n):
            x += 1
            if (4**x == n):
                return True
        return False
```

Testing and Debugging (FR, CR, SA)

1. List 5 categories of test cases, and give an example for each

   ANS: Normal Case: assert(digitCount(1234) == 4)
   Edge Case: assert(digitCount(7) == 1)
   Special Case: assert(digitCount(0) == 1)
   Varying Result: assert(digitCount(20) == 2)
   Large Input Case: assert(digitCount(54365463734365) == 14)

2. Indicate if there's anything wrong with the following statements/functions:
   a) Kevin wrote a function that takes in a number n and returns the number of multiples of 3 up to that number.
```
def f(n):
    count = 0
    number = 1
    while (number < n):
        if (number % 3 == 0):
            count =  count -1
   return count
```
   ANS: This function is not correct. Right now we have an infinite loop that won't break. To fix it, we need to change "count = count - 1" to "count = count + 1". We also need to increment number in the loop (outside the if statement) so that there is no infinite loop and so that the statement number < n will eventually be false.

b) Zack wrote this function called `same(s)` trying to count the number of pairs of the same character inside a string. (for example: `same("dad")` returns `1`)

```
def same(s):
    counter = 0
         for i in range (len(s)-1):
              for j in range (1, len(s)):
                    if (s[i] == s[j]):
                          counter = counter + 1
    return counter
```

ANS: This function is not correctly implemented. The range of the second loop is wrong. Instead of searching from index 1, it should start at index i+1 so that the same character won't be checked multiple times.

For Loops (CW, FR)

1. Explain when you might use a for-range loop and when you might use a for-each loop.
   ANS: I would use a for-range loop when I want to repeat actions for a specified number of times. I would use a for-each loop when I want to loop over iterable objects.

2. Similarly, when would you use a while loop versus a for loop? Can you always convert a for loop to a while loop? Can you always convert a while loop to a for loop?
   ANS: You usually use a while loop when you don't know how many iterations are going to occur. You can always convert a for loop into a while loop but not the other way around for the reason stated earlier.

3. Write a function `numberOfFactors(n)` which takes in a natural number (not including 0) and returns the number of factors it has.
   ANS:

```
def numberOfFactors(n):
    counter = 0
     for i in range(1,n+1):
         if (n%i == 0):
              counter += 1
     return counter
```

4. Using a for loop, write the function `fizzBuzz(n)` that prints every number from 0 to n-1 inclusive. If the number is divisible by 3, print “fizz” instead of the number. If the number is divisible by 5, print “Buzz” instead of the number. If divisible by both 3 and 5, print “fizzBuzz” instead of the number.

   ANS:
   ```
   def fizzBuzz(n):
           for i in range(n):
                if (i % 3 == 0 and i % 5 == 0):
                     print(“fizzBuzz”)
                elif (i % 3 == 0):
                     print(“fizz”)
                elif (i % 5 == 0):
                     print(“Buzz”)
                else:
                     print(i)
   ```

5. Using a for loop, write the function `sumAllEven(n)` that finds the sum of all even numbers less than or equal to n.
   ANS:
   ```
   def sumAllEven(n):
       sum = 0
       for i in range(n+1):
           if i % 2 == 0:
                sum += i
       return sum
   ```

Strings (CW, CR)

1. Write a function `reverseString(s)` that returns the string s reversed.
   ANS:
   ```
   def reverseString(s):
           return s[::-1]

       def reverseString(s):
           reversed = “”
           for c in s:
                reversed = c + reversed
           return reversed
   ```

2. What would the following code print?

```
def mystery(s, n):
     for word in s.split(“ “):
          if len(word) == n:
               return word
     return “Darn!”

print(“She sells seashells down by the seashore”, 4)
```

ANS: “down”