ANSWER SHEET

These problems were generated by TAs and instructors in previous semesters. They may or may not match the actual difficulty of problems on Quiz2.

Booleans, Conditionals, and Errors

1. What will the following code output?

```
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))
```

ANSWER:

```
-71 43
```

2. Write a function `canEatIceCream(temp, hunger)` to determine whether somebody should eat ice cream on a hot day based on the integer temp (must be greater than 60 degrees) and the float hunger (must be greater than 0.5). Return the result.

ANSWER:

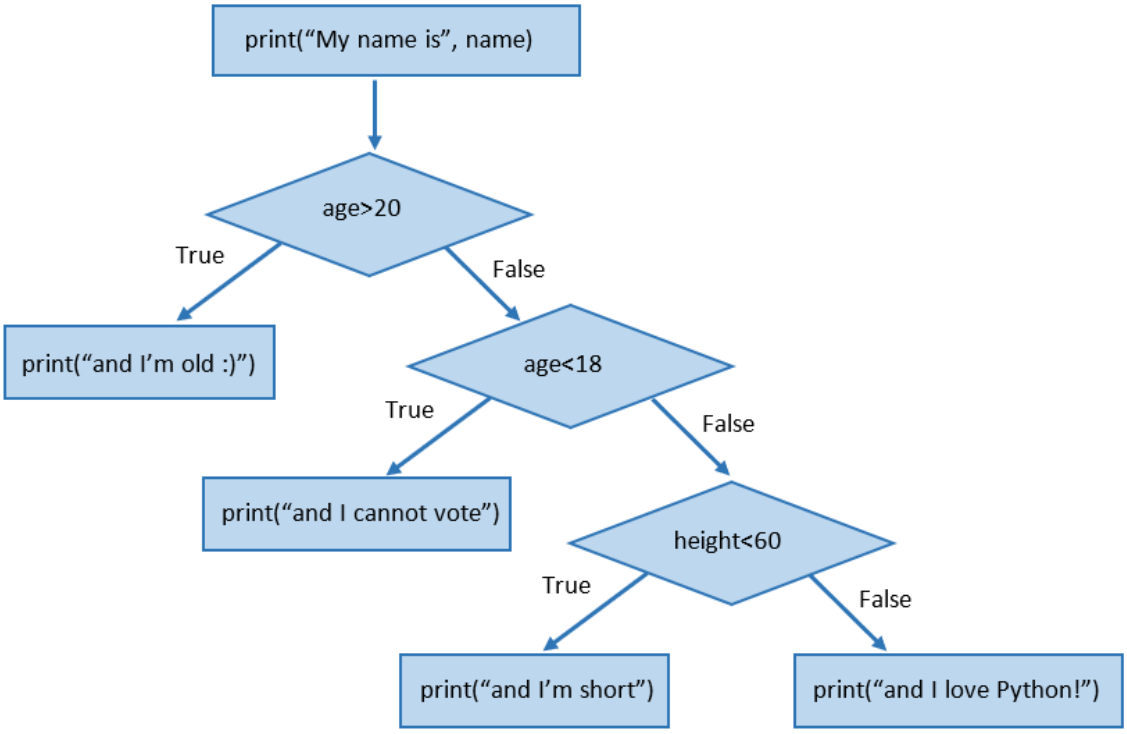
```
def iceCream(temp, hunger):
      if temp > 60:
            if hunger > 0.5:
                  return True
      return False
```

3. What is the difference between the `and` vs. `or` operations in terms of their relationship with the boolean `True`?

ANSWER:

and evaluates to True only when both values are True, while or evaluates to True when either value is True

4. Convert the following flow chart into equivalent Python code.



Answer:

```
print("My name is", name)
if age>20:
      print("and I'm old :)")
elif age<18:
      print("and I cannot vote")
elif height<60:
      print("and I'm short")
else:
      print("and I love Python!")
```

5. Explain the following error messages and provide one possible fix:

1)

```
Traceback (most recent call last):
  File "<tmp 1>", line 1, in <module>
    print("hello"+1)
TypeError: can only concatenate str (not "int") to str
```

2)

```
    print("name"",name)
                  ^
SyntaxError: EOL while scanning string literal
```

3)

```
def f():
    print("hello 15-110")
        return 15-110
```

```
  File "<tmp 1>", line 3
    return 15-110
    ^
IndentationError: unexpected indent
```

Answer:

1) Type error: You cannot add a string to an integer. You can change 1 to the string of “1” to add them together, assuming the string you would like to print is “hello1”.
2) Syntax error: The quotation marks do not match properly and Python cannot interpret them. Fix this by removing one of the quotes that are next to each other, so the statement should read print(“name”, name).
3) Indentation Error: The return statement should not be indented. Fix this by deleting the indent and realigning the return statement with the print statement.

6. The following function takes in two numbers a and b, prints them as "a, b" and then returns their sum (for example, given a=3 and b=7 the function will print "3, 7" and it will return 10). Find the errors and classify them (into one of the three broad types of errors).

```
def printValuesReturnSum(a, b)
    print(a, b)
    return "a" +  b
```

Answer:

Line 1 - Syntax Error (missing colon)
Line 2 - Logical error (this will print “a b” without the comma)
Line 3 - Runtime error (can’t concatenate string and int)

7. The following lines of code contain multiple errors.

```
numerator = 0                # line 1
denominator = 6              # line 2
x = denominator/Numerator    # line 3
print("x', x)               # line 4
```

1) Find the errors and categorize them into the three general types of errors discussed in class.
2) If you run these lines of code in Pyzo, it will produce an error message. State which error in the code will cause the error message and explain your answer. In other words, which error is detected first and why? (You should be able to answer this without actually running the code)

Answer:

1) Line 3 - Runtime error: name error, capital "N" used in the variable numerator (remember that case matters when assigning variables) (even though this might have been caused by a simple typo, remember that it is a runtime error not a syntax error!)
   Line 3 - Runtime error: division by zero
   Line 3 - Logical error: numerator and denominator are mixed up
   Line 4 - Syntax error: EOL due to miss-matched parentheses

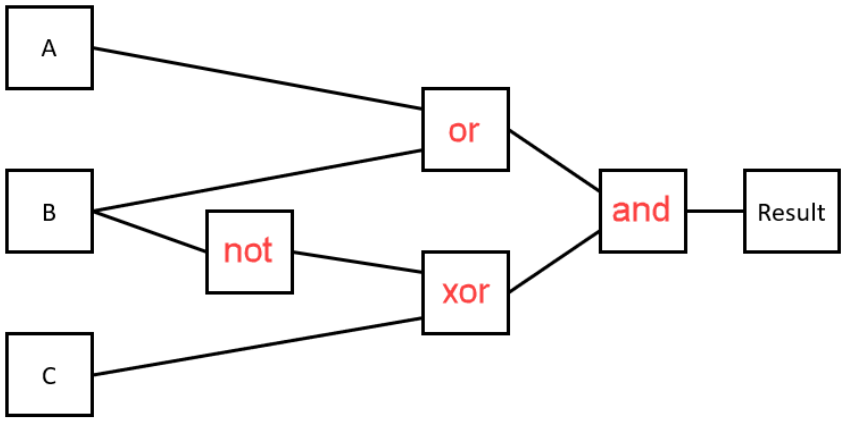
2) The syntax error on line 4 causes the error. Even though there are runtime errors occurring in an earlier line, syntax errors are always recognized first. We know this because syntax errors are caused by issues with tokenizing or parsing, but runtime errors occur later when bytecode is being executed.

Circuits and Gates

1. Given the following boolean expression, fill out a truth table that shows all the possible results of the expression, then label the gates on the circuit below with AND/OR/etc. so that it produces the same results.

(A or B) and ((not B) xor C)

ANSWER:



A	B	C	Result
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

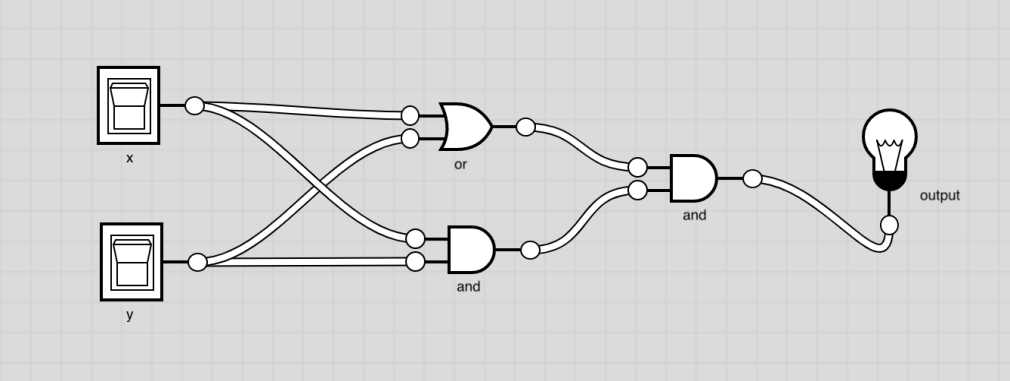
2. Recall that in lecture we built a simple addition machine called a Full Adder. Clearly name and describe the purpose of the input(s) and output(s) of this machine.

ANSWER:
Inputs: X and Y are single digits of the numbers being added. C_in is the number carried from the previous addition.
Outputs: Sum is the right digit of the result; C_out is the left digit, which will be carried to the next addition.

3. What is the main difference between a half adder and a full adder?

ANSWER:
A half adder can only add two digits, while a full adder can add three digits.

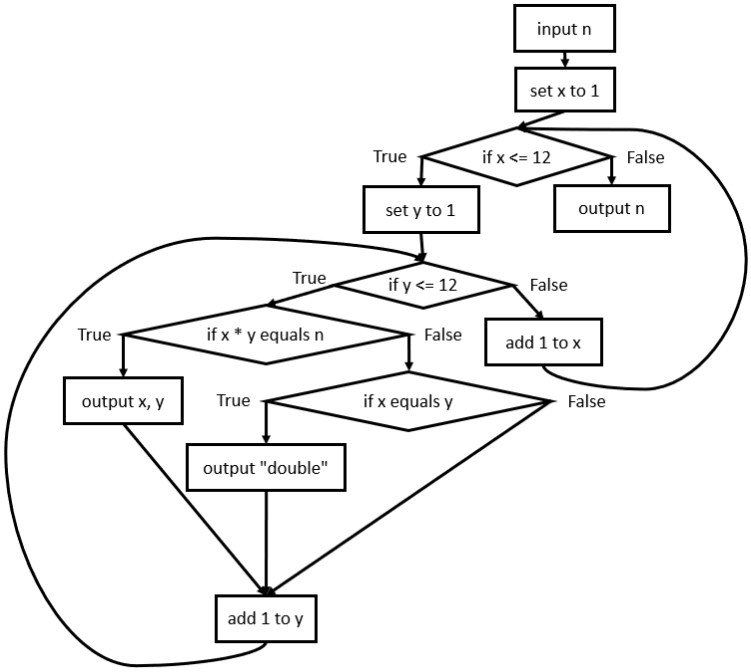
4. What boolean operation does the following logic circuit behave like?



ANSWER:
X and Y

While Loops

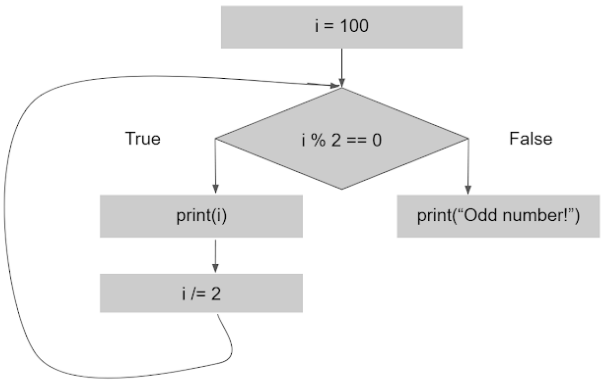
1. Write a function `cw1(n)` that is algorithmically identical to the control flow chart shown below. The function should take an integer n as a parameter and **print** output as specified, returning nothing.



ANSWER:

```
def cw1(n):
    x = 1
    while x <= 12:
        y = 1
        while y <= 12:
            if x * y == n:
                print(x, y)
            elif x == y:
                print("double")
            y = y + 1
        x = x + 1
    print(n)
```

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



ANSWER:

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

3. Use while loop to write the 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, and False otherwise.

ANSWER:

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

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

ANSWER:

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

For Loops

1. Explain when you would use a while loop versus a for loop over a range. Can you always convert a for loop to a while loop? Can you always convert a while loop to a for loop?

ANSWER:
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.

2. Write a function `numberOfFactors(n)` which takes in a positive integer and returns the number of factors it has.

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

3. 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.

ANSWER:
```
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)
```

4. Using a for loop, write the function `sumAllEven(n)` that finds the sum of all even numbers less than or equal to n.

ANSWER:

```
def sumAllEven(n):
      sum = 0
      for i in range(n+1):
            if i % 2 == 0:
                  sum = sum + i
      return sum
```

Looping over Strings

1. Read through the following block of code, and write what it will output..

```
s = "Computer Science"
t = "GO-1-TEN"

print("A:", s[4])
print("B:", t[len(t)-2])
print("C:", s[6:12])

for i in range(2, 10, 4):
    print(s[i] + t[i])
```

ANSWER

```
A: u
B: E
C: er Sci
m-
eE
```

2. Write a function `whileSmile(s)` that takes a string as input and uses a **while loop** to count the number of times the two-character string ":)" occurs in s. You should return the count. For example: `whileSmile("Hello :):):)")` should return 3.

ANSWER:

```
def whileSmile(s):
     i = 0
     count = 0
     while i < len(s)-1:
          if s[i] == ":" and s[i+1] == ")":
               count = count + 1
          i = i + 1
     return count
```

3. Write a function `reverseString(s)` that returns a reversed version of the string s.

ANSWER:

```
def reverseString(s):
      return s[::-1]
```

or

```
def reverseString(s):
      reversed = ""
      for i in range(len(s)):
            reversed = s[i] + reversed
      return reversed
```