These problems were generated by TAs and instructors in previous semesters. They may or may not match the actual difficulty of problems on Exam1.

Algorithms and Abstraction

1. As you learned in lecture, computer science boils down to two main ideas: algorithms and abstraction.
   1) Identify/Define the difference between algorithms and abstraction, and provide an example for each.
   2) What are the 3 key components of a good algorithm?

Answer:

1) Algorithms: algorithms are a set of instructions to solve a given task or problem. An example of an algorithm is a food recipe or a “how-to” manual

   Abstraction: technique used to make complex systems manageable by changing how much detail is used to represent/interact with the system

   An example of abstraction is your internet browser. Instead of needing to know the exact destination of the information you want to see on the internet, you can search websites by their url or by keywords relating to it.

2) Key components:

   Input: Should have a specified input needed in the beginning

   Steps: Should have a clearly-described process for how the algorithm runs

   Output: Should have a specified output produced at the end

Programming Basics

1. For each of the following Python expressions, identify the data type of the value it will evaluate to.
    1) 3
    2) 3.0
    3) “3”
    4) 3 < 3
    5) “3” + “3”
    6) 3.0 * 3.0
    7) 3 / 3

Answer:

1) Integer (int)
2) Floating point number (float)
3) String (str)
4) Boolean (bool)
5) String (str)
6) Floating point number (float)
7) Floating point number (float)

2. Suppose you are creating a few variables to describe yourself or another person. Name the data type that would **best** fit each variable description:

1) Name
2) Age
3) Height (inches)
4) Ability to vote
5) Address
6) Zip code
7) Weight
8) School you attend
9) Ability to speak more than one language

Answer:

1) Str
2) Int (float could work but is less fitting)
3) Float
4) Bool (look below)
5) String
6) Int or String
7) Int or Float
8) String
9) Boolean

Data Representation

1. Answer the following questions regarding number systems:
   1) What is a number system?
   2) How many values can a specific digit have in a decimal number? In a binary number?
   3) Let's say that we are representing length in imperial units. You are given inches, feet, and yards. How could you represent 10,000 inches in this system (show your work)? (hint: 12 inches=foot, 3 feet=yard 1760 feet=mile)

Answer:

1) A way of representing numbers using symbols
2) Decimal: 10 (0,1,2,3,4,5,6,7,8,9)
   Binary: 2 (0,1)
3) 10000 [inches] // 36 [inches] = 277 [yards]
   10000 [inches] % 36 [inches] = 28 [inches]
   28 [inches] // 12 [inches] = 2 [feet]
   28 [inches] % 12 [inches] = 4 [inches]

Yards	Feet	Inches
277	2	4

277 yards and 2 feet and 4 inches is equivalent to 10,000 inches
(277yards + 2 feet + 4 inches) == (10000 inches)

2. Complete the following conversions and show appropriate work:
   1) Convert 10111011 from binary to decimal
   2) Convert 11111111 from binary to decimal
   3) Convert 01101110 from binary to decimal
   4) Convert 57 from decimal to 8-bit binary
   5) Convert 63 from decimal to 8-bit binary
   6) Convert 88 from decimal to 8-bit binary

Answer:

1) 187 (1+2+8+16+32+128=187)
2) 255 (1+2+4+8+16+32+64+128=255 OR note that the next bit would be 256, so we know making all 8 bits 1 will be 255)
3) 110 (2+4+8+32+64=110)
4) 00111001 (57=32+16+8+1)
5) 00111111 (63=32+16+8+4+2+1)
6) 01011000 (88=64+16+8)

3. The following questions relate to interpreting binary numbers as abstracted types
    1) Answer the following questions relating to abstracting colors:
        a) How many bytes are needed to represent an RGB value?
        b) What are the highest and lowest values for any color?
        c) Convert the color purple to an RGB value. Provide the values in decimal and binary.
        d) Convert 11010101 11110000 00110011 to a color (You can use this link to determine what color is appears to be: https://www.w3schools.com/colors/colors_rgb.asp)
    2) Answer the following questions relating to abstracting text (ASCII table: http://www.asciitable.com/):
        a) Convert 00111010 00101001 to ASCII characters.
        b) Convert 'CMU cmu' into 8-bit binary (neglect the quotation marks).

Answers:

1) Colors:
    a) 3 bytes (24 bits, 3 8-bit values)
    b) 255 is the highest value, 0 is the lowest
    c) Purple is (R=255,G=0,B=255) or 11111111 00000000 11111111
    d) (213, 240, 51)->yellowish
2) Text:
    a) :) (00111010=58, 00100101=41)
    b) 01000011 01001101 01010101 00100000 01100011 01101101 01110101
    'C'=67 'M'=77 'U'=85 ' '=32 (space!)
    'c'=99 'm'=109 'u'=117

Functions (Calls and Definitions)

1. Evaluate the print statements and state what the value of y and x are at the end of the function.

```
x = 15+1

def f():
    y = 12
    print(y * 2)
    print(y)
    y = "apple"
    print("bananas, 15-110, and",y)
    print(x+3)

f()
```

Answer:
Statements printed:
24
12
"bananas, 15-110, and apple"
19

The value of y at the end of the code is "apple". The value of x is 16.

2. Consider the following function:

```
def helloThere(name):
    print("Hello there,", name, "!")
```

1) What will you see in the shell if you call print(helloThere("Stella")) ?
2) Does this function have side effect(s)? If so, state the side effect(s).
3) Does this function have a returned value? If so, state the returned valued.
4) What do we call the statement helloThere("Stella")? What do we call the string "Stella" in that statement?

Answer:

1) Shell output:
   Hello there, Stella !
   None
2) Yes, the print statement in the function is the side effect
3) Yes, the returned value is None. (all functions return None by default if another value is not specified)
4) helloThere("Stella") -> function call
   "Stella" -> input or argument are both acceptable (parameter is incorrect, the variable *name* is the parameter)

3. Consider the following algorithm for a function:

Algorithm:

1. Input a number n
2. If n is greater than or equal to 0 do the following, otherwise skip to step 3:
    a. If n is divisible by 10 then print out the value of n followed by " is cool number", otherwise skip to step 2b
    b. If step 2a did not occur, check if n is divisible by 4, and if so print out the value of n followed by " is a strange number", otherwise skip to step 2c
    c. Skip to step 4
3. Print the value of n followed by " is a negative number!!"
4. Print the string "ALL DONE"

Identify the argument(s), returned value, and side effect(s) of this algorithm. Then write the corresponding Python function called divisionCheck.

Answer:
The argument to the function is n. The returned value will be None, since we don't return anything in the function. The side effects are the possible print statements.

```
def divisionCheck(n):
    if (n >= 0):
        if ((n % 10) == 0):
            print(n, "is a cool number")
        elif ((n%4) == 0):
            print(n, "is a strange number")
    else:
        print(n, "is a negative number!!")
    print("ALL DONE")
```

4. The function fruitCalculator has parameters percentApples, totalFruit, and farm. fruitCalculator calculates the number of apples harvested by the farm, prints the number of apples and the number of other fruit, and returns a string announcing the harvest of apples. For example, fruitCalculator(0.75, 400, “Carnegie Farms”) would return “Carnegie Farms harvested 300 apples this year”, and the following would be printed in the console.

300 apples were harvested!
100 other fruits were also harvested!

Write the function fruitCalculator according to the description above.

Answer:

```
def fruitCalculator(percentApples, totalFruit, farm):
    numApples = percentApples * totalFruit
    numOther = totalFruit - numApples
    print(numApples, "apples were harvested!")
    print(numOther, "other fruits were also harvested!")
    return farm + " harvested " + str(numApples) + " apples this year"
```

5. Suppose we had the following segment of code:

```
course = "15110"
grade = 95.0
def randomFunction(x, y):
      sum = x + y
      difference = x - y
      print(course)
      return (sum + difference) / 2
```

Give the scopes, either local or global, of the following variables: course, grade, sum, x, y, and difference.

Answer:
Global variables: course, grade
Local variables: x, y, sum, difference

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.

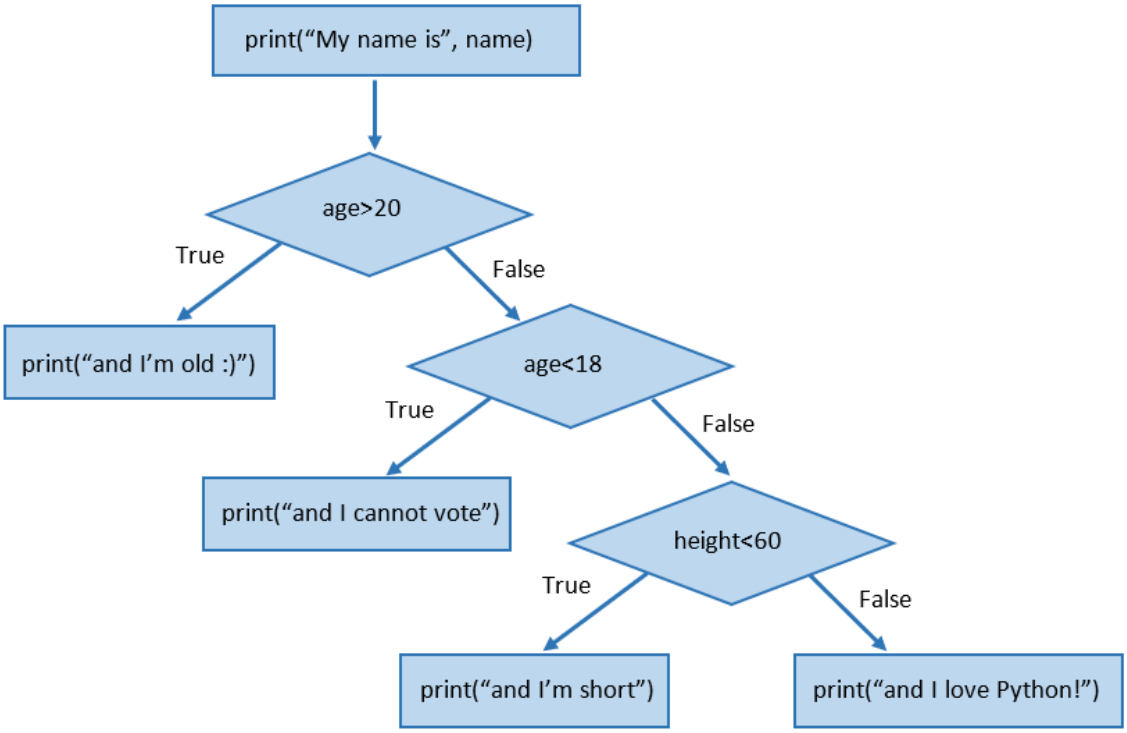
ANSWER:

```
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

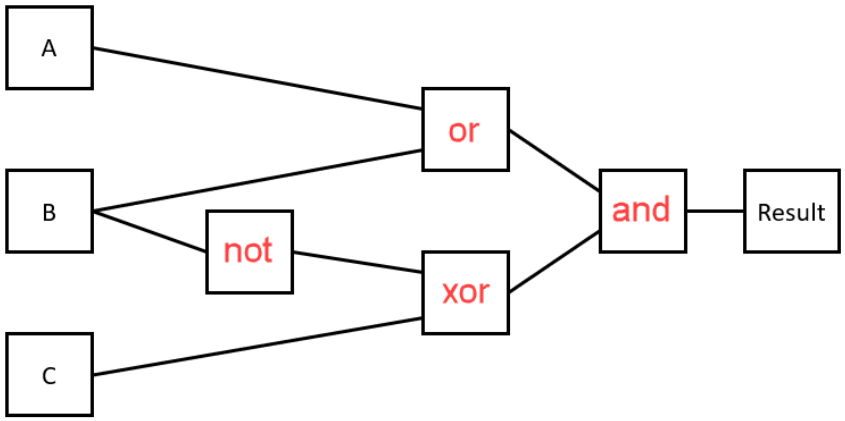
2) The syntax error on line 4 causes the error. Even though there are runtime errors occurring in an earlier line, syntax errors are always recognized first. We know this because syntax errors are caused by issues with tokenizing or parsing, but runtime errors occur later when bytecode is being executed.

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

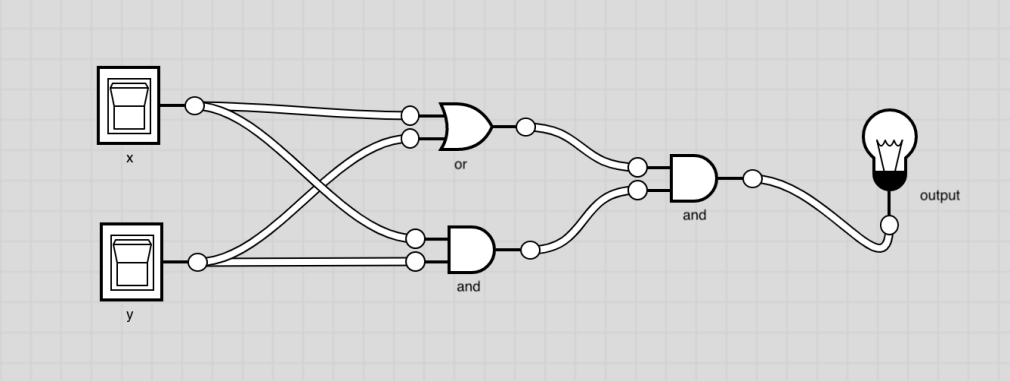
2. Recall that in lecture we built a simple addition machine called a Full Adder. Clearly name and describe the purpose of the input(s) and output(s) of this machine.

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.

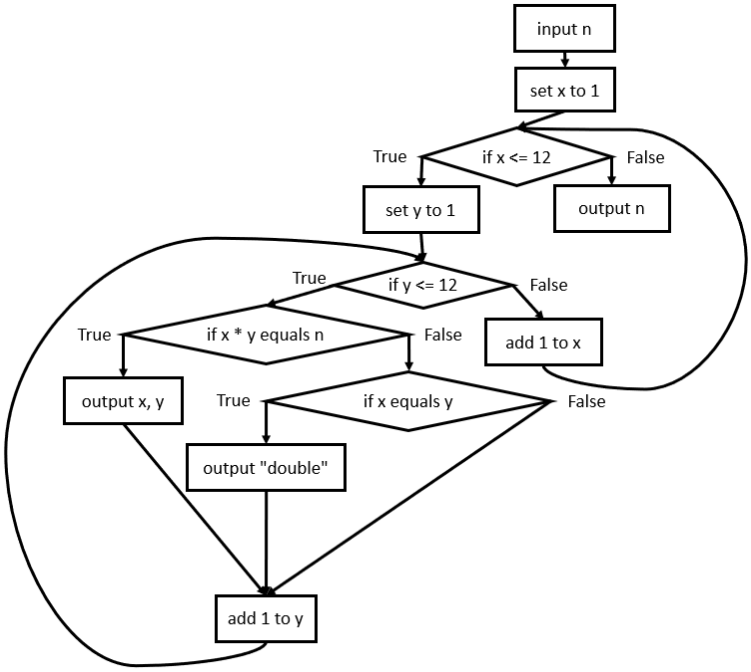
4. What boolean operation does the following logic circuit behave like?



ANSWER:
X and Y

While Loops

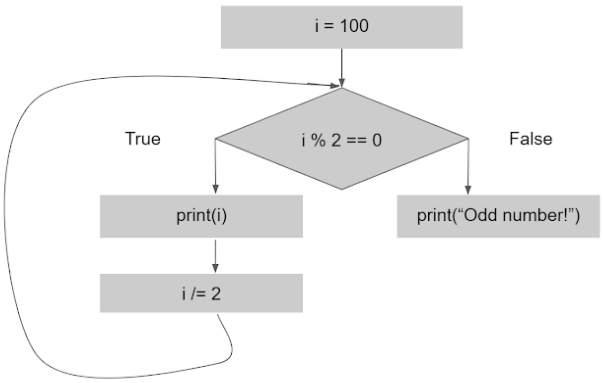
1. Write a function `cw1(n)` that is algorithmically identical to the control flow chart shown below. The function should take an integer n as a parameter and **print** output as specified, returning nothing.



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
```

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
```