***Remember**: These are just example practice problems written by TAs. They do not reflect the true exams or topics covered by them.*

Abstraction and Algorithms - definitions

1. A diamond in a flow chart represents:

    a. A variable assignment
    b. A while loop
    c. A test of true or false
    d. A value true or false

    Answer: C

2. True or False: Abstraction allows us to think about a problem at different levels.

    Answer: True.

Binary

1. Conversion to/from Decimal

    a. Represent 110 as an 8-bit binary number.

        Answer: 01101110

    b. What's the biggest number you can represent with 8-bit unsigned binary?

        255

2. Addition

    a. 10101010 + 01001100 =

        Answer: 011110110 (170 + 76 = 246)

    b. 01101101 + 01101010 =

        Answer: 11010111(109 + 106 = 215)

Data abstraction - pixels, ascii

1. How many bytes are in a pixel? What do they represent?

   Answer: 3 bytes, 1 for the level of red, 1 for blue, and 1 for green

2. If the letter A is the integer 65, write the binary to represent G (Hint: convert G to int and then to binary).

   Answer: G is 71, in binary 01000111

Bytecode

Literal Table:

Addr	Literal
1	15
2	110

Variable Table:

Addr	Variable
0	x
1	y
2	z
3	product

What is the corresponding Python code for this byte code?

LOAD_CONST 1
STORE_FAST 0
LOAD_CONST 2
STORE_FAST 1
LOAD_FAST 1
LOAD_FAST 0
BINARY_ADD
STORE_FAST 2

Answer:

```
x = 15
y = 110
z = y + x
```

What is the number that is stored at address 2?

Answer: 110 + 15 = 125

Given that to multiply numbers you would use BINARY_MULTIPLY, what would you add to the bytecode if the following line is added to they python code?

product = z * x

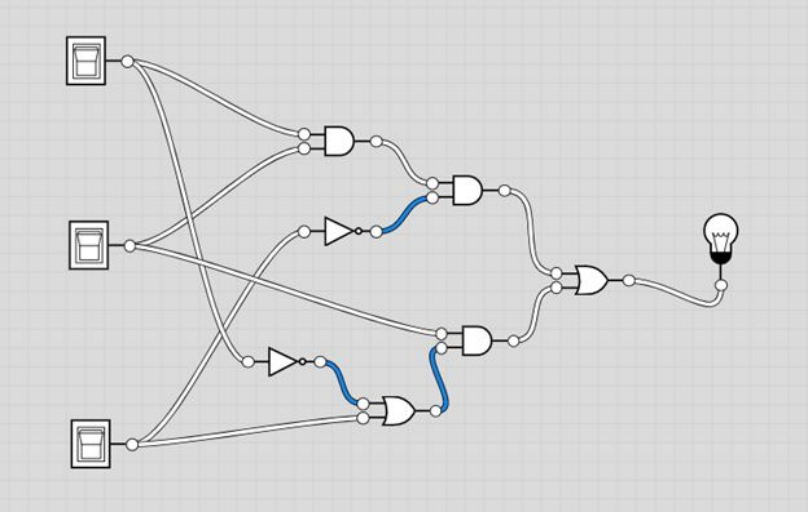
Answer:

```
LOAD_FAST 2
LOAD_FAST 0
BINARY_MULTIPLY
STORE_FAST 3
```

Logical operations

1. Logic gates, Boolean expressions, truth tables, interconversions

    a. Given the following circuit diagram, translate it into a boolean expression. On an exam, you can assume you will be given a legend of the names of the gates.



    b. Assume that X is at the top, Y is in the middle, and Z is at the bottom.

Answer: `(X AND Y AND (NOT Z)) OR (((NOT X) OR Z) AND Y)`

c. Using the circuit diagram and your answer to part A, generate a truth table. Be sure to include all combinations of inputs (should have 8 rows in total!) and be very clear about what your columns represent!

Answer:

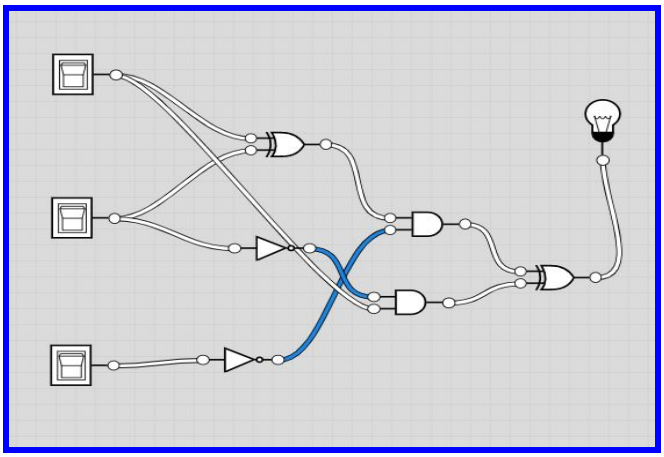
X	Y	Z	Output
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	1
0	0	1	0
0	0	0	0

d. Given the following truth table, generate a Boolean expression for it in terms of its inputs:

X	Y	Z	Output
1	1	1	0
1	1	0	0
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	1
0	0	1	0
0	0	0	0

Answer: `((X XOR Y) AND (NOT Z)) XOR (NOT Y AND X)`

e. Draw the circuit for your answer to part C



2. Half Adder, Full Adder

   a. Consider adding two bits, X and Y. Complete the table for adding those bits together in binary. The first row is done for you:

X	Y	X + Y
1	1	10
1	0	01
0	1	01
0	0	00

   b. Consider the output to be separated into two bits, carry and output. For example, if X = 1 and Y = 1 and X + Y = 10, then the carry bit would be 1 and the output would be 0. With that in mind, what logical operation would you use to get carry bit for X + Y? Write a truth table to represent only the carry bit, then determine the logical operator.

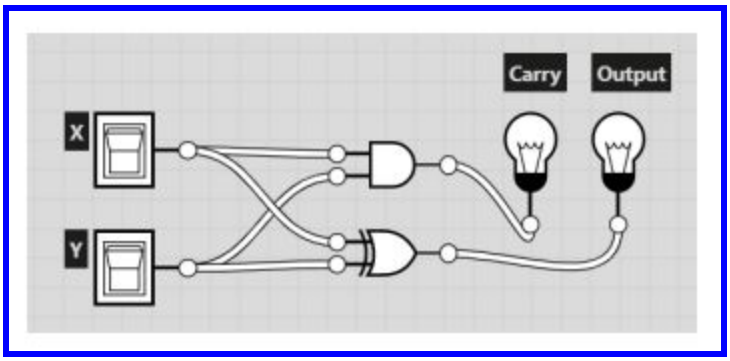
   Answer: Truth table is the left bit in X+Y column of a. X AND Y for the carry.

   Do the same for the output bit.

   Answer: Truth table is the right bit in X+Y column of a. X XOR Y for the output

c. Using these two logical operations, draw out the circuit for adding two bits X and Y. Keep in mind that you have TWO outputs, a carry bit and an output bit!

Answer:



d. We can consider a Full Adder to add three bits together: X, Y, and the carry bit from a previous addition, which we will call C_{in}. Fill out the following table for the Full Adder. The first two rows are completed for you.

C_{in}	X	Y	C_{in} + X + Y	C_{out}	Output
1	1	1	11	1	1
1	1	0	10	1	0
1	0	1	10	1	0
1	0	0	01	0	1
0	1	1	10	1	0
0	1	0	01	0	1
0	0	1	01	0	1
0	0	0	00	0	0

e. What logical operation on the inputs X, Y, and C_{in} would give the C_{out} column of the table? The output column of the table?

Answer: ((X XOR Y) AND C in) OR (X AND Y) for C out, (X XOR Y) XOR C in for output column.

Conditionals - code tracing, code writing

1. Write a function f(x) that prints “foo” if the value x is a multiple of 2 and 3, print “bar” if x is a multiple of 2 and NOT a multiple of 3, and “boop” if x is a multiple of 5 only. Otherwise return None.

```
def f(x):
      if x % 2 == 0:
            if x % 3 == 0:
                  print(“foo”)
            else:
                  print(“bar”)
      elif x % 5 == 0:
            print(“boop”)
      else:
            return None
```

2. Trace the following code using the parameters below:

```
def codeTrace(a, b, c, d):
     if a < 12:
          if a < 0:
               if len(b) > 5:
                    return b[5:] + c
               elif len(b) > 2:
                    return b[2:]
               else:
                    return c
          elif a > 0:
               if d == 0:
                    return b
               else:
                    return a + d
          else:
               if len(c) > 3 and len(c) < 8:
                    Return len(c) + d
               else:
                    return len(c) - d
     else:
          if len(b) == 0:
               if len(c) > 5:
                    return c[:5]
               else:
                    return c + str(d)
          else:
               return str(a) + str(d)
```

a. codeTrace(20, “hello”, “goodbye”, 10)

Answer: “2010”

b. codeTrace(5, “hello”, “goodbye”, 10)

Answer: 15

c. codeTrace(-6, “hi”, “foo”, 3)

Answer: “foo”

d. codeTrace(0, “hi”, “15110”, 4)

Answer: 9

Errors

1. Identify whether errors exist in the following versions of the function isPerfectSquare. This function should return True or False based on whether x is a perfect square, where a perfect square is any number that can be made by squaring a whole number (for example: 25 is a perfect square since 5^2=25, but 12 is not a perfect square because no whole number can be multiplied by itself to equal 12)

   If there is no error, write no error. If there is an error, identify it in the code (circle/ underline/ describe), label the type of error and edit the code to correct the error. (only receive credit for naming the type of error if you correctly identify the error)

```
# Perfect squares include: 0, 1, 4, 9, 16, 25, ... 100, ...


def isPerfectSquare1(x): #check if x is a perfect square
      if x**0.5 == integer(x**0.5):
            return True
      return False
```

Answer: runtime error (NameError: 'integer' not defined)

```
      #if x**0.5 == int(x**0.5):
```

```
def isPerfectSquare2(x): #check if x is a perfect square
      if (x**0.5)/1 == (x**0.5)//1):
            return True
      else:
            return False
```

Answer: syntax error

Unmatched parenthesis after //1 - either delete it or add matching open parenthesis in front

```
def isPerfectSquare3(x): #check if x is a perfect square
      return (x**0.5)%1 == 1
```

Answer: logical error

Want remainder to be zero after finding square root

```
      #return (x**0.5)%1 == 0
```

2. Answer the following T/F questions about errors, and provide a few words / one sentence explanation of your reasoning
    a. Unmatched parentheses are an example of a syntax error

       Answer: True - incorrect syntax that interpreter could not properly tokenize

    b. Division by zero is an example of a logical error since we should know it's impossible

       Answer: False - a runtime error because python cannot compute the result

    c. Runtime errors can occur when the interpreter cannot figure out how to parse your python code

       Answer: False - this can describe syntax errors

    d. Trying to get the integer value of a string by using int("hi"), for example, is a logical error

       Answer: False - runtime error (performing operation of wrong type)

    e. If your code contains both a runtime and a syntax error, your interpreter will show you only whichever error occurs first

       Answer: False - will always show syntax error first (even if syntax error is on line 15 and runtime error is on line 2)

    f. Test functions that the professors provide in homework starter files check for logical errors

       Answer: True - only catch logical errors

Loops

1. Write a function triangle that takes in a string and prints out the first letter, then the second two.. Etc and then back down in a triangle form. (using a for loop)

   Example: triangle('Bye'):

   B
   By
   Bye
   By
   B

   Answer:

```
def triangle(s):
      for i in range(len(s)):
            print(s[0:i])
      for i in range(len(s), 0, -1):
            print(s[:i])
```

2. Write a function square that takes in an integer n and returns a list that includes the squares of every number less than and not including n using a while loop.

Answer:

```
def square(n):
      squares = []
      i = 1
      while i < n:
            squares += [i**2]
            i += 1
      return squares
```

3. Write a function longestElem(L) that determines the longest word in a list of strings and returns it. There could be multiple -- in this case, return a list of all of those words

Answer:

```
def longestElem(L):
      longest = ['']
      longestLength = 0
      for word in L:
            if len(word) > longestLength:
                  longest = [word]
                  longestLength = len(word)
            else if len(word) == longestLength:
                  longest.append(word)
      if len(longest) > 1:
            return longest
      else:
            return longest[0]
```

Strings - code writing

1. Write the function vowelAtOddIndexCount(s) that takes a string s and returns an integer count of all the vowels that are at an odd index in s. For example, vowelAtOddIndexCount('pumpkin') would return 2, because there are two vowels at odd indexes in the string (1 and 5, specifically).

   Answer:

```
def vowelAtOddIndexCount(s):
      count = 0
      for i in range(len(s)):
            if i % 2 == 1 and s[i] in 'aeiou':
                  count += 1
      return count
```

2. Write the function lastNamesOnly(nameString), where nameString is a comma separated string of first names and last names (separated by a space), and the function returns a list of the last names.

   For example:
   - Suppose nameString == 'John Doe,Jane Doe,Johnny Johnson,Sandy Sanderson'
   - (Assume that there are no spaces after the commas)
   - lastNamesOnly(nameString) would return ['Doe', 'Doe', 'Johnson', 'Sanderson']

   Hint: Notice that spaces separate the first and last names

```
def lastNamesOnly(nameString):
      lastNameList = [ ]
      for name in s.split(','):
            spaceIndex = name.find(' ')
            lastName = name[spaceIndex + 1:]
            lastNameList.append(lastName)
      return lastNameList
```

Input from shell and files - code tracing

1. Trace the following code and write two possible values of `guess` that would give different printouts. What does it print? Write EVERYTHING it prints until the next return or input line, except the first "Guess a number between 1-10" that you respond to with a guess.

```
def guessinggame(guesses):
      num = 2
      for i in range(guesses):
            guess = input("Guess a number between 1-10")
            if int(guess) == num:
                  print("You got it!")
                  return
            else:
                  print("Try again")
      return
```

guess = _____

What does it print?

Answer:
guess = "2"
Print:
You got it

guess = _____

What does it print?

Answer:
guess = "1" or string with a number not "2"
Print:
Try again!
Guess a number between 1-10

Note: You must put the guess in a string. Input always returns a string. We would accept a printout with strings but not required.

2. Trace the following code and write 20 words or less summarizing what it returns. Do NOT write what each line is doing. Reminder, the sorted(list) function returns a new sorted version of list.

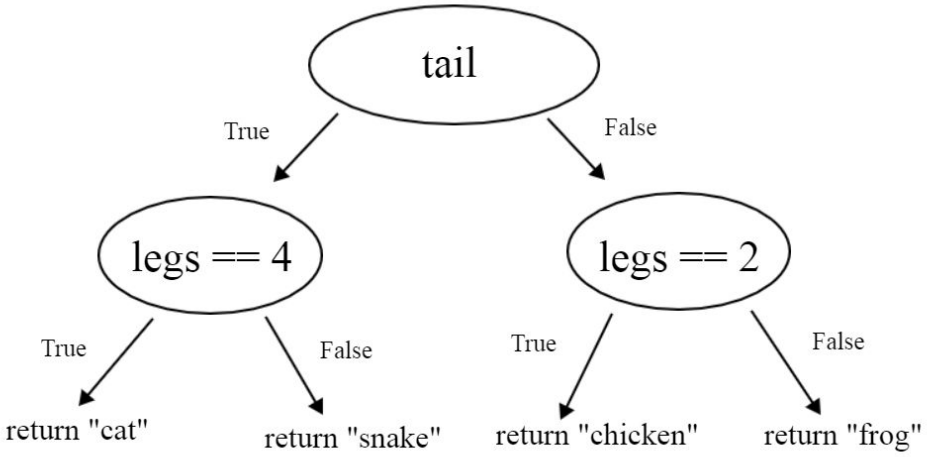
```
def fileread(filename):
      f = open(filename, "r")
      text = f.read()
      lines = text.split("\n")
      vals = []
      for line in lines:
            vals.append(int(line))
      sortvals = sorted(vals)
      if len(sortvals) == 0:
            return None
      else:
            return sortvals[len(sortvals)//2]
```

Answer: It returns the median element if the list is not empty and otherwise returns None.

More code writing

1. Using the control flow chart below, write a function classifyAnimal(tail, legs) that takes in an integer representing the number of legs the animal has and a boolean representing if it has a tail and returns a string of the animal name.

   Note: In this image, the ovals are the same as diamonds we learned in class flow charts.



Answer:

```
def classifyAnimal(legs, tail):
      if tail:
            if legs == 4:
                  return “cat”
            else:
                  return “snake”
      else:
            if legs == 2:
                  return “chicken”
            else:
                  return “frog”
```

2. Write a function doubleLetters(s) that takes in a string and returns the number of pairs of letters in the string. For example, doubleLetters(“aabbcc”) returns 3 and doubleLetters(“ababcc”) returns 1. There will be no more than two of the same letter in a row.

Answer:

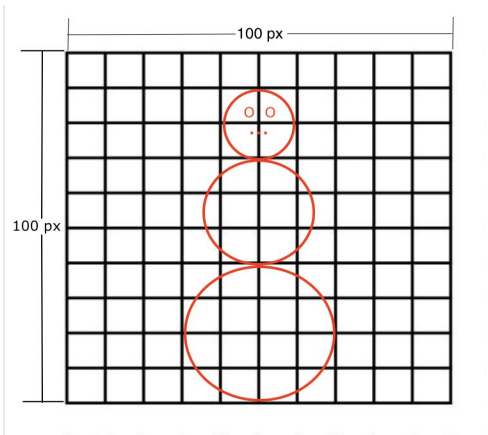
```
def doubleLetters(s):
    count = 0
    for i in range(len(s) - 1):
        if (s[i] == s[i + 1]):
            count = count + 1
    return count
```

Graphics - code tracing

1. Assume that we set up a canvas appropriately, then ran the following lines of code. Draw the resulting graphics in the box below. The box has dimensions 100x100 pixels.

   canvas.create_oval(40, 10, 60, 30)
   canvas.create_oval(35, 30, 65, 60)
   canvas.create_oval(30, 60, 70, 100)
   canvas.create_text(50, 20, text = “O O”, anchor = “s”)
   canvas.create_text(50, 20, text = “. . .”, anchor = “n”)

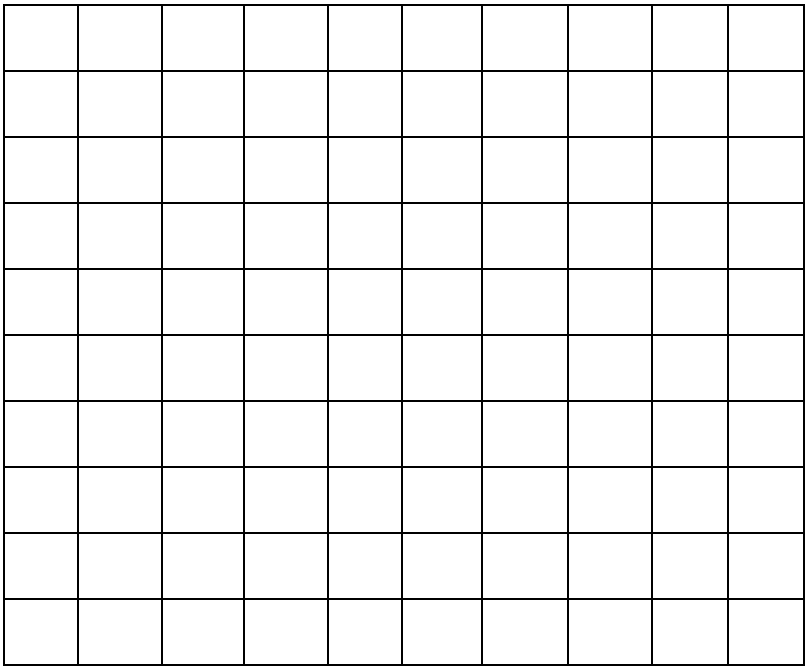
   Each square is 10 long and 10 high...

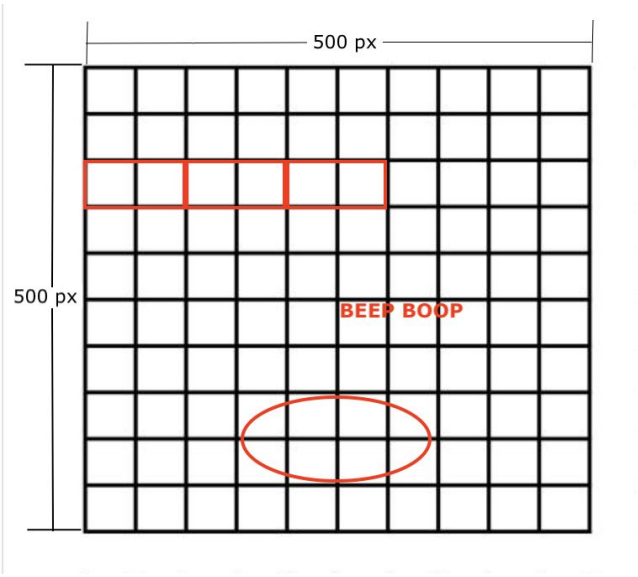


2. Assume that we set up a canvas appropriately, then ran the following function. Draw the resulting graphics in the box below. The box has dimensions 500x500 pixels.

```
def mysteryDraw(canvas):
    size = 100
    for i in range(3):
        canvas.create_rectangle(size*i, 100, size*(i + 1), 150)
    canvas.create_text(250, 250, text = “BEEP BOOP”, anchor = “nw”)
    canvas.create_oval(150, 350, 350, 450)
```

Each square is 50 long and 50 high...




500 px
500 px
BEEP BOOP

Lists - code writing

1. Write a function to convert a list of multiple integers to a single number. Assume all elements of the list will be an integer.

   For example, listToNum([1,2,4,23]) = 12423 and listToNum([7,15,10]) = 71510

```
def listToNum(lst):
    finalNumber = ''
    for item in lst:
        finalNumber += str(item)
    return int(finalNumber)
```

2. Write a function to rotate a list by a certain amount to the right. Ex. rotate([1,2,3,4], 1) = [4,1,2,3] and rotate([1,2,3,4,5], 3) = [3,4,5,1,2]

```
def rotateList(lst, n):
    new = n % len(lst)
    return lst[len(lst)-new:] + lst[:len(lst)-new]
```

3. Write a function to return a list of duplicates from a given list lst.

```
def remove_dups(lst):
      dup_items = []
      uniq_items = []
      for x in lst:
            if x not in uniq_items:
                  uniq_items.append(x)
            else:
                  dup_items.add(x)
      return dup_items
```

4. Write a function that returns the second largest element in a list. There may be duplicates, but the second largest should not equal the first largest (this is the hardest part). If there are fewer than 2 elements or if there are fewer than 2 unique elements, it should return None.

   One way:

```
def second_largest(numbers):
      dup_items = []
      uniq_items = []
      for x in lst:
            if x not in uniq_items:
                  uniq_items.append(x)
            else:
                  dup_items.add(x)
      uniq_items.sort()
      if len(uniq_items) < 2:
            return None
      return uniq_items[len(uniq_items)-2]
```

   Another way:

```
def second_largest(numbers):
      if (len(numbers)<2):
            return None
      maxlist = []
      numberscopy = numbers+[] #so the func is not destructive
      while len(maxlist) < 2 and len(numberscopy) > 0:
            maxval = max(numberscopy)
            if maxval not in maxlist:
                  maxlist.append(maxval)
            else:
                  numberscopy.remove(maxval)
      if len(maxlist) < 2:
            return None
      return maxlist[1]
```

2D lists - code tracing

A. Trace the following function. What is the 2D list that is printed?

```
def hardmystery(a ,b):
      r = []
      for c in range(1, a, 2):
            d = [c]*c
            r.append(d)
      for e in range(b, 1, -1):
            for f in r:
                  if (len(f) > e):
                        f[e] = '1'
      return r
print(hardmystery(6, 3))
```

Answer: `[[1], [3, 3, '1'], [5, 5, '1', '1', 5]]`

B. Trace the following function. What is printed in the function? What is printed after the function returns?

```
def hardmystery2(lst):
      result = []
      for i in range(len(lst)-1,-1,-1):
            n = lst[i]
            for e in n:
                  if e.isupper() or e.isdigit():
                        result += [e]
      print(result)

print(hardmystery2([['i', 'love', '15', '110'], ['i', 'love',
'ice', 'cream'], ['I', 'AM', 'STUDYING', 'rn']]))
```

In function Answer: ['I', 'AM', 'STUDYING', '15', '110']

Outside function Answer: None

Destructive vs non-destructive methods

1. A. Write a function that destructively reverses the order of the elements in a 1D list, but doesn't use another temporary list nor the reverse function. Return None. Hint: At any given point, L must contain all the elements within the original list L. Think about how you could swap elements in the list. Which would you swap to reverse the list?

```
def reverseFlip1(L):
      for i in range(len(L)/2):
            temp = L[len(L) - i]
            L[len(L) - i] = L[i]
            L[i] = temp
      return None
```

   B. Then, write a function that NON-destructively reverses the order of the elements in a 1D list, and returns the new reversed list.

```
def reverseFlip2(L):
      M = []
      for i in range(len(L)-1,-1,-1):
            M.append(L[i])
      return M
```

Aliasing (at a simple level) - code tracing

1. What does the following code print?

```
def ct1(L):
      A = [ ]
      C = L
      C[0] = 5
      B = C
      B[3] = 12
      L[0] = 32
      print("A =" + str(A))
      print("B =" + str(B))
      print("C =" + str(C))
      print("L =" + str(L))
ct1([23,4,5,745,643,33])
```

Answer:

```
A = []
B = [32, 4, 5, 12, 643, 33]
C = [32, 4, 5, 12, 643, 33]
L = [32, 4, 5, 12, 643, 33]
```

2. What does this function print?

```
def ct2(L):
    print("L =" + L)
    L = [0, 4, 5]
    A = L
    A[1] = 1
    print("A =" + A)
    print("L =" + L)
ct2([0,0,11,15,110,8])
```

Answer:

```
L = [0,0,11,15,110,8]
A = [0,1,5]
L = [0,1,5]
```

Mutability - short answer

1. a) List 2 data types that are mutable and 2 data types that are immutable.

   b) What does it mean for a data type to be mutable in terms of their memory location?

   Answer
   a) Mutable: list, dictionary
      Immutable: integer, string
   b) If a data type is mutable like list, then changing the value of the list does not necessarily change its memory location.

2. What are the limits on data types for keys and values in a dictionary? Why are there limitations?

   Answer: Lists cannot be keys for dictionary, because they are mutable and hashing a list key that has been changed would result in a different number. Values can be any data type.

Recursion

1. Tracing: what does this function print for each of the following lists L?

```
def recursion1(L):
      if len(L) == 0:
            return 1
      else:
            a = L[0]
            b = L[1:]
            if a % 2 == 1:
                  return a * recursion1(b)
            else:
                  return recursion1(b)

print(recursion1([1,6,4,2,9])) __________

print(recursion1([2,3,4,5,6,7])) __________

print(recursion1([2,4,6,8,10,12])) _________
```

      Answer: 9, 105, 1

2. Tracing: what does this function print for each of the following inputs?

```
def recursion2(L, n):
      if len(L) == 0:
            return [n]
      else:
            r = []
            a = L[0]
            b = L[1:]
            r += recursion2(b, n)
            r += recursion2(b, n+a)
            r = list(set(r))
            r = sorted(r)
            return r

print(recursion2([1, 2, 3], 0))
print(recursion2([0], 3))
print(recursion2([2, 4, 6], 1))
```

Answer: [0, 1, 2, 3, 4, 5, 6], [3], [1, 3, 5, 7, 9, 11, 13]

3. Tracing: what does this function print?

```
def recursion3(x, y):
    if (x == y):
        return 0
    else:
       if x % 2 == 0:
          return x + recursion3(x+1, y)
        return x - recursion3(x+1, y)

print(recursion3(2, 5))
```

Answer: 1

4. Tracing: what does this function print?

```
def recursion4(s):
    if (len(s) < 2):
        return s
    else:
        mid = len(s)//2
        return recursion4(s[mid:]) + recursion4(s[:mid])

print(recursion4("abcd"))
```

Answer: "dcba"

5. Write a recursive function getIndices(s, c) that returns a list of all indices i of s such that s[i] equals the character c. E.g. getIndices(“hello world”, “l”) = [2, 3, 9].

```
def getIndices(s, c):
      if len(s) == 0: return []
      currIndex = len(s) - 1
      if s[currIndex] == c:
            return [currIndex] + getIndices(s[:-1], c)
      return [] + getIndices(s[:-1], c)
```

6. Write a recursive function countEven(n) to count the number of even digits in an integer n. E.g. countEven(5334) = 1, countEven(2468) = 4, countEven(0) = 1.

```
def countEven(n):
      num = str(n)
      if len(n) == 0:
            return (int(n) % 2 == 0)
      else:
            currDigit = int(n[0])
            if (currDigit % 2 == 0):
                  even = 1
            else:
                  even = 0
            return even + countEven(n[1:])
```

7. Write a recursive function printEvenLeaves(T) that prints all of the leaves of the tree T that have an even value.

```
def printEvenLeaves(T):
     if T['children'] == []: # we're at a leaf
          if T['value'] % 2 == 0:
               print(T['value'])
     else:
          for child in T['children']:
               printEvenLeaves(child)
```

8. Write a recursive function index(L, e) that finds the index of the first instance of the element e in the list L. If the element doesn't exist, return -1. Hint: If the function finds the element in a smaller list, how should you update the index in the recursive case before returning?

```
def index(L, e):
      if len(L) == 0:
            return -1
      else:
            if L[0] == e:
                  return 0
            elif (index(L[1:],e) == -1):
                  return -1
            else:
                  return 1 + index(L[1:],e)
```

Runtime and Big-O Notation

1. Simplify this Big O expression: O(((10N)(50N))/6 + N + N/2)

   Answer: O(N^2)

2. Write the Big-O runtime of this program in terms of N. Make sure your Big O is simplified and assume that all named functions have the runtimes mentioned in class.

```
def whatIsMyBigO(s): # len(s) == N
      result = []
      count = 0
      for char in s:
            value = ord(char)
            If linearSearch(result, value) == False:
                  result.append(value)
            If value == 42:
                  count = count + 1
      return result
```

   Answer: O(N^2)

Search Algorithms

1. Write, in order, the elements checked to find 6 in the following list for both linear and binary search: L = [1,5,6,10,11,12]. (For binary search,assume middle element is extracted with len(L)//2)

Answer: Linear search -> 1,5,6, Binary search -> 10, 5, 6

2. If you chose to sort a list and then search it, what is the fastest runtime you could achieve?

Answer: O(nlogn). You would need to sort it using merge sort O(nlogn) and then look for it using binary search O(logn) or linear search O(n).

Sorting Algorithms (Insertion sort, Selection sort, Mergesort)

1. In less than 5 sentences, describe the bigO and behavior of the following sorting algorithms when it is applied to an unsorted list of size n.

    a. Insertion sort

    Answer: O(n^2) Iterate through each element in the input list and insert them in the correct place in the output list

    b. Selection sort

    Answer: O(n^2) Find the smallest (or largest) element in the input list and swap it with the first (or last element)

    c. Mergesort

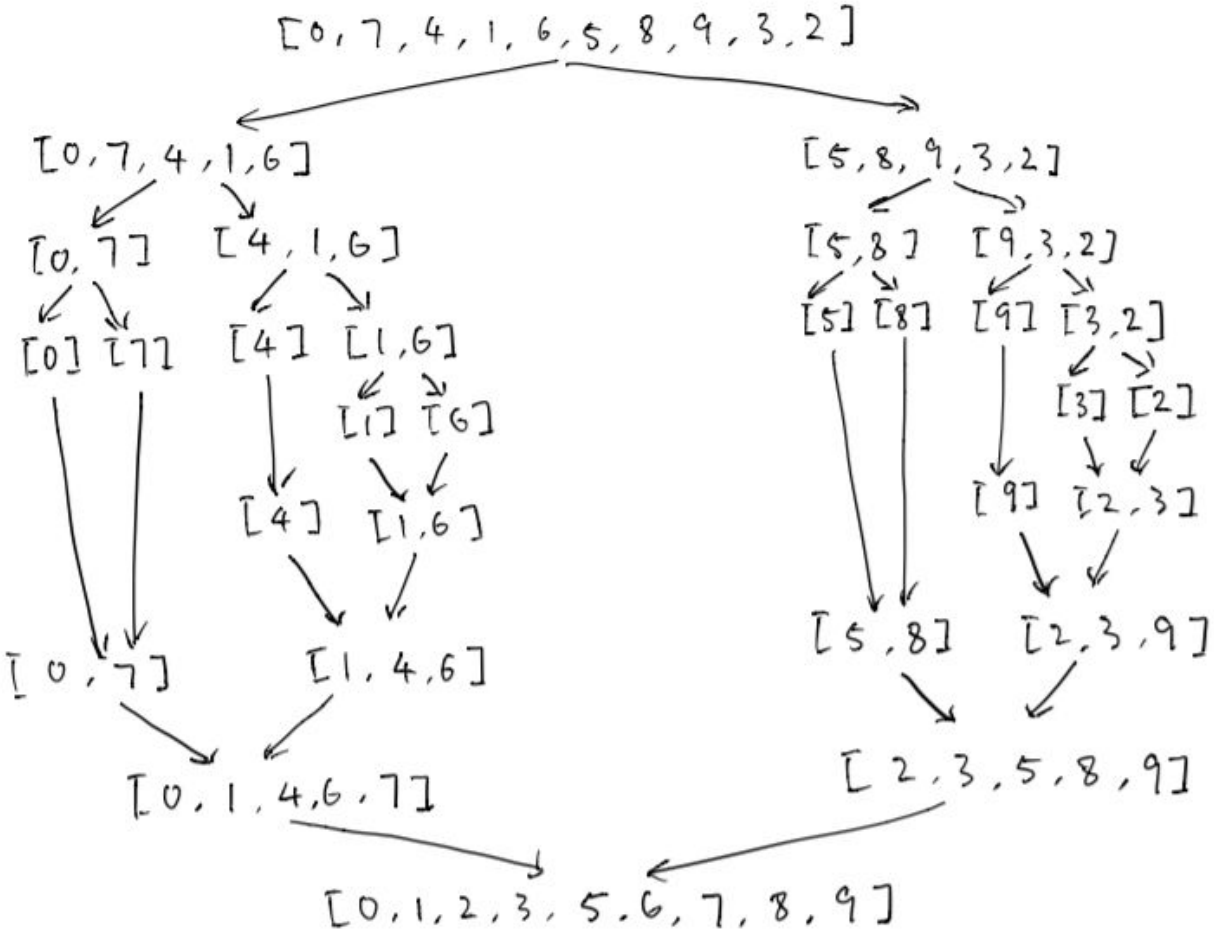
    Answer: O(log n) Recursively split the input list and join it back together in order using a merge function

2. Draw a diagram of how the list [0, 7, 4, 1, 6, 5, 8, 9, 3, 2] would be split and merged using mergeSort.

   IMPLEMENTATION OF MERGESORT IF NEEDED.

```
def mergeSort(L):
    # Base case: empty and one-element lists are already sorted
    if len(L) == 0 or len(L) == 1:
        return
    # Split the list in half, recursively sort each half
    mid = len(L) // 2
    left = L[:mid]
    right = L[mid:]
    mergeSort(left)
    mergeSort(right)

    # Merge the two sorted lists back together
    leftIndex = 0
    rightIndex = 0
    i = 0
    # Only need to compare the first unsorted elements,
    # because it's sorted!
    while leftIndex < len(left) and rightIndex < len(right):
        if left[leftIndex] < right[rightIndex]:
            L[i] = left[leftIndex]
            leftIndex = leftIndex + 1
        else:
            L[i] = right[rightIndex]
            rightIndex = rightIndex + 1
        i = i + 1
    # Add any remaining elements to the end of the list
    while leftIndex < len(left):
        L[i] = left[leftIndex]
        leftIndex = leftIndex + 1
        i = i + 1
    while rightIndex < len(right):
        L[i] = right[rightIndex]
        rightIndex = rightIndex + 1
        i = i + 1
```


[0, 7, 4, 1, 6, 5, 8, 9, 3, 2]
[0, 7, 4, 1, 6]
[5, 8, 9, 3, 2]
[0, 7]
[4, 1, 6]
[5, 8]
[9, 3, 2]
[0] [7]
[4] [1, 6]
[5] [8]
[9] [3, 2]
[1] [6]
[3] [2]
[4]
[1, 6]
[9]
[2, 3]
[0, 7]
[1, 4, 6]
[5, 8]
[2, 3, 9]
[0, 1, 4, 6, 7]
[2, 3, 5, 8, 9]
[0, 1, 2, 3, 5, 6, 7, 8, 9]

Hash Tables/hash functions - short answer

1. In **one** sentence each: (a) What is the underlying data structure of a hash table? (b) What does a hash table store? (c) How are the locations for data chosen? (d) How can we check if an element is in a hash table?

   Answer: *A hash table is a table (or list). It stores one or more elements at each index. The index for each element is obtained by hashing the element and modding the result by the length of the table. We can check if an element is in the hash table by hashing it and modding the result by the length of the table, then looking at that index in the table.*

2. Describe one concrete situation in which a hash table would **not** be an appropriate choice of data structure. Explain why a hash table would not be appropriate in that situation. (Bonus: What data structure would you use instead?)

   Answer: Any answer in which the value to store is a mutable data structure. Mutable data structures would hash to different values if they change.

3. We will use the following table (of length 12) as a hashtable. We want to store the names of students in this table. We use the following hash function: hash(student) = birthday month

   The students and their birthdates are as follows:

   Alice: August

   Bob: September

   Charlie: December

   Diana: July

   Eleanor: September

   Frank: July

   Gary: January

   Fill in the hashtable. (You may use the student's initial to represent their whole name, e.g., A for Alice.)

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec

Answer

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
G						D F	A	B E			C

Dictionaries

1. **Code Writing -**
    a. Write the function invert_dict(D) that takes a dictionary in which the values can be strings, integers, or lists. The function should return a dictionary that has the keys and values of the original dictionary swapped, but the new values are contained in lists. However, if the value of the original dictionary was mutable, we should not swap them and instead insert them as key and a list of that value (e.g., “class”:[“110”,”112”] would be inserted as “class”:[[“110”,”112”]]. If two new keys overlap, append the new value to the existing list. Note that in order to check if a value is a list, you can use type(val) == list.

Example: invert_dict({ “profs” : [“Rivers”, “Rosenthal”], “best ta” : “Trevor”, 110 : “profs”}) == { “profs” : [[“Rivers”, “Rosenthal”], 110] , “Trevor” : [“best ta”] }

```
def invert_dict(D):
    newdict = dict()
    for key in D:
        new_key = D[key]
        new_val = key
        if type(new_key) == list:
            new_key = key
            new_val = D[key]
        if new_key in newdict:
            newdict[new_key].append(new_val)
        else:
            newdict[new_key] = [new_val]
    return newdict
```

b. What would change in the solution to the problem above if we change the prompt as follows: Any lists only contain strings and/or integers. Change the handling of lists as values in the original dictionary to be such that the values of the lists now act as their own value of the initial key.

Example: invert_dict({ “profs” : [“Rivers”, “Rosenthal”], “best ta” : “Trevor”, 110 : “Rivers”}) == { “Rivers” : [“profs”, 110], “Rosenthal” : [“profs”], “Trevor” : [“best ta”]}

Answer: We would need to carry out the operation of reversal for each element in the lists, so we can perform a nested for loop with the lists. New code in bold.

```
def invert_dict(D):
    newdict = dict()
    for key in D:
        new_key = D[key]
        new_val = key
        if type(new_key) == list:
            for item in new_key:
                if item in newdict:
                    newdict[item].append(new_val)
                else:
                    newdict[item] = [new_val]
        else:
            if new_key in newdict:
                newdict[new_key].append(new_val)
            else:
                newdict[new_key] = [new_val]
    return newdict
```

2. **Code Writing** - Write the function max_freq(s) that takes a string s and returns the number of occurrences of the character which appears the most times. The function must have complexity O(N). For example, max_freq("trevor is the best") = 3 because there are three e's and three t's.

```
def max_freq(s):
    D = dict()
    for c in s:
        if c in D:
            D[c] += 1
        else:
            D[c] = 1
    max_count = 0
    for c in D:
        if D[c] > max_count:
            max_count = D[c]
    return max_count
```

3. **Code Writing** - Write a function inOrder(d) that takes in a dictionary mapping strings to strings (i.e. d1 = { “trevor”:”russell”, “connor”:”clancy”}) and returns a list of the values sorted by their key in alphabetical order.

   Examples:

   inOrder(d1) == [“clancy”, “russell”]

   inOrder({ “b”:”y”, “a”:”z”, “c”:”x”}) == [“z”, “y”, “x”]

   inOrder({“alpha”:”b”, “beta”, “b”}) == [“b”, “b”]

```
def inOrder(d):
    keys = []
    vals = []
    for k in d:
        keys.append(k)
    keys.sort()
    for k in keys:
        vals.append(d[k])
    return vals
```

Note: dictionaries are not sorted and we cannot assume any order of the keys

4. **Code Writing** - Write a function combineDicts that takes two dictionaries d1 and d2 that both have string keys and int values and returns a new dictionary that has all of the keys and values from both d1 and d2, if d1 and d2 share a key then the value of that key in the new dictionary should be the sum of the values in d1 and d2.

   Examples:

   D1 = {“a”: 1, “b”: 0, “c”: 5}

   D2 = {“c”: 10, “d”:1, “e”:8}

   D3 = {“a”: -2, “b”: 100, “f”: -19}

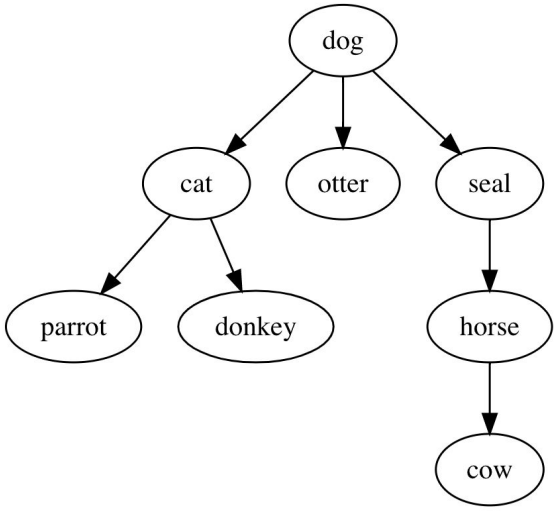
   combineDicts(D1, D2) == {“a”:1, “b”:0, “c”:15, “d”:1, “e”:8}

   combineDicts(D2, D3) == {“a”:-2, “b”:100, “c”:10, “d”:1, “e”:8, “f”:-19}

   combineDicts(D1,D3) == {“a”:-1, “b”:100, “c”:5, “f”:-19}

```
def combineDicts(d1, d2):
    d = {}
    for k1 in d1:
        d[k1] = d1[k1]
    for k2 in d2:
        if k2 in d:
            d[k2] = d[k2] + d2[k2]
        else:
            d[k2] = d2[k2]
    return d
```

Trees



1. Use the above graph to answer the following questions:

List all the nodes in the graph.

Answer: dog, cat, parrot, donkey, otter, seal, horse, cow

Which node is the root?

Answer: dog

Which nodes are leaves?

Answer: parrot, donkey, otter, cow

Which nodes are parents of cat?

Answer: dog (in a tree, nodes only have one parent)

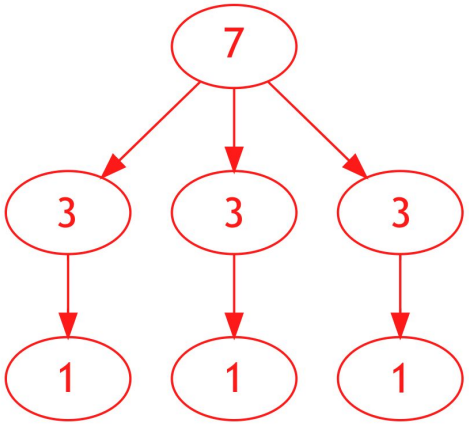
Which nodes are children of dog?

Answer: cat, otter, seal

2. Consider the following mystery function:

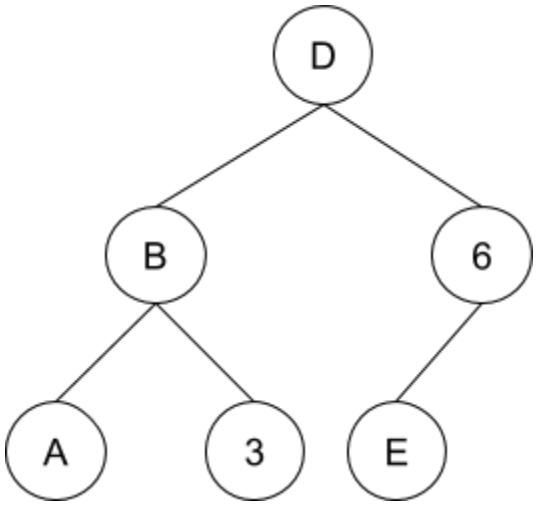
```
def mystery(num):
    if num <= 1:
        return {"value": num, "children": []}
    t = {"value": num, "children": []}
    next_num = num // 2
    for i in range(next_num):
        t["children"].append(mystery(next_num))
    Return
```

Draw the tree that's created if we call `mystery(7)`

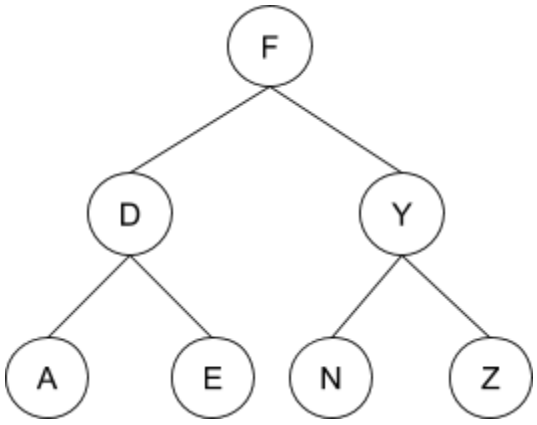


Advanced Trees

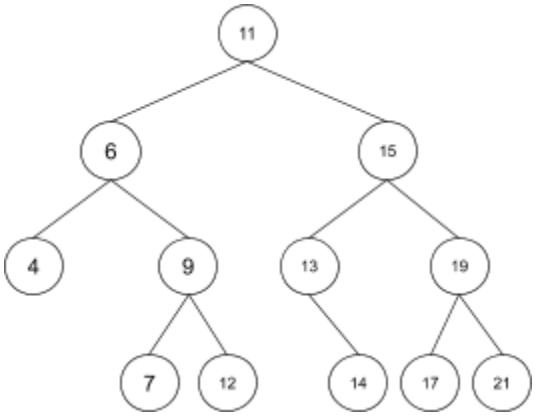
1. For each of the following tree diagrams, write **all** tree structures (trees, binary trees, and binary search trees) that match the diagram.



Answer: Tree

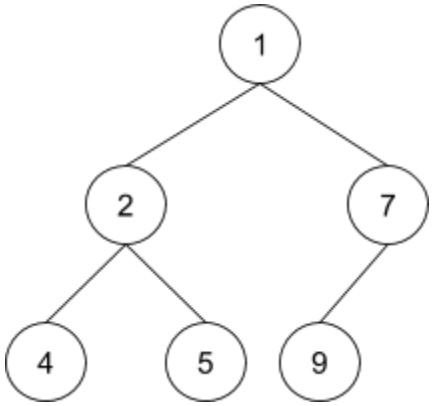


Answer: Tree, Binary Tree, Binary Search Tree (BST is true because in ASCII A is less than D, E is greater than E, etc)



Answer: Tree, Binary Tree (12 is out of place for a BST)

2. In this problem, you will translate the algorithm below into a recursive Python function preOrder(tree) that will compute the preorder traversal of an input binary search tree. A preorder traversal is a type of tree traversal is where we first print the nodes value and then recursively run the function on each child. In this case, instead of printing, we will append to a string and return it. For example, given the following tree, the function would return "124579".



    1. Base Case: If the visited node is a leaf, convert the value of the current node from an int into a string and return it.
    2. Recursive Case:
        a. Convert the value of the current node from an int into a string and save it in a variable *nodeVal*.
        b. If there is a left subtree, add the value of calling your function on the left subtree to the variable *nodeVal*.
        c. If there is a right subtree, add the value of calling your function on the right subtree to the variable *nodeVal*.
        d. Return *nodeVal*.

```
def preOrder(tree):
     if tree["left"] == None and tree["right"] == None:
          return str(tree["value"])
     else:
          nodeVal = str(tree["value"])
          if tree["left"] != None:
               nodeVal += preOrder(tree["left"])
          if tree["right"] != None:
               nodeVal += preOrder(tree["right"])
          return nodeVal
```

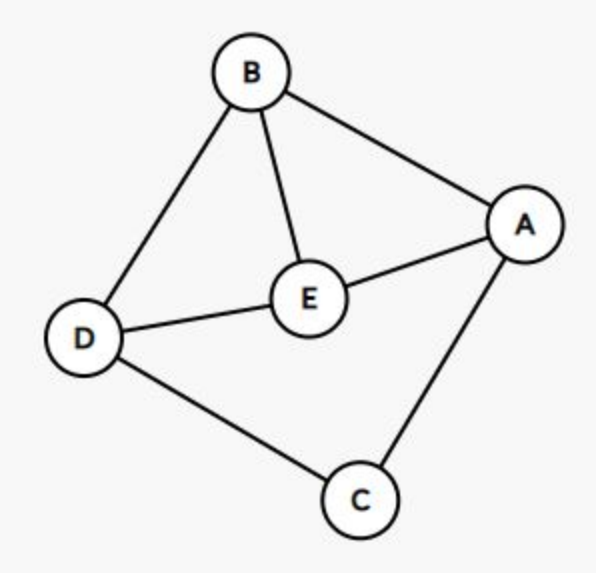
Graphs

1. Given the following code, draw the **undirected** graph.

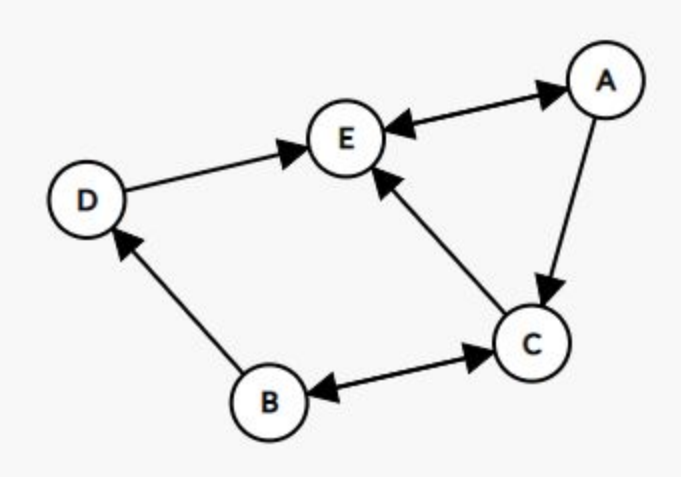
   graph = {"A": ["B" ,"C", "E"], "B": ["A" , "D", "E"], "C": ["A", "D"], "D": ["B", "C", "E"],

   "E": ["A" , "B", "D"] }

   Answer:



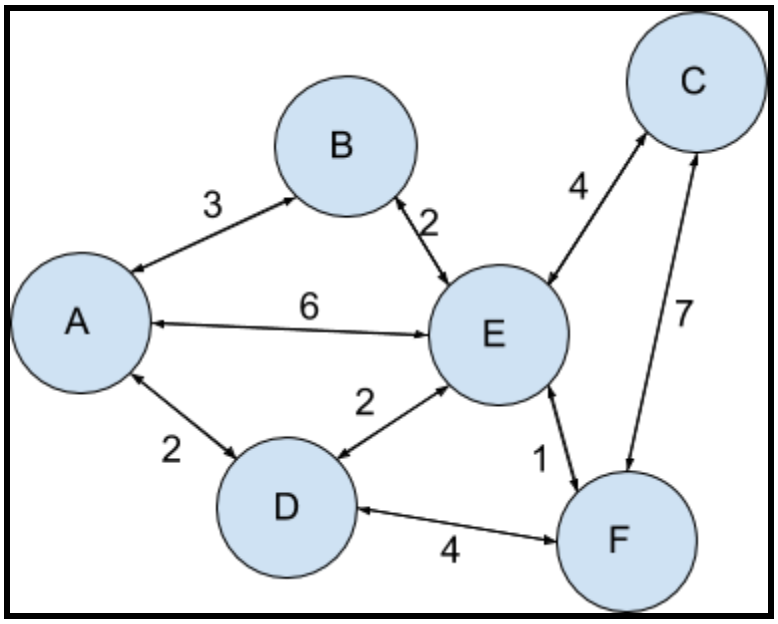
2. Write the dictionary representation of the following graph



Answer:

```
graph = {A: ["C", "E" ],
      B: ["C" , "D"],
      C: ["B", "E"],
      D: ["E"],
      E: ["A"] }
```

3. Fill in the adjacency matrix below for the given graph.

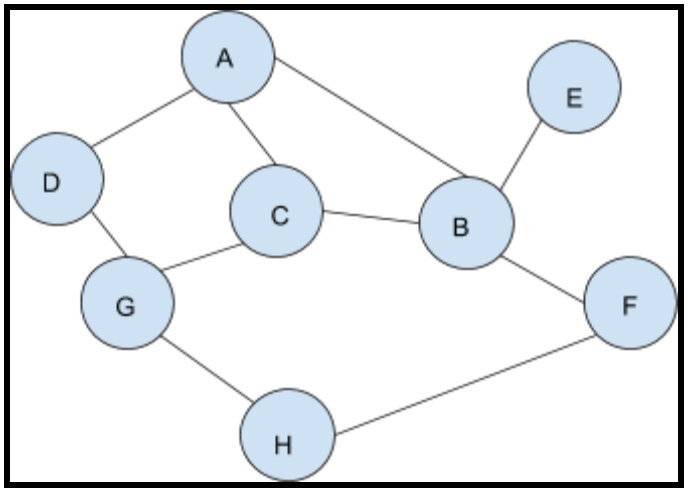



```
graph =
   # A   B   C   D   E   F
[  [    ,   ,   ,   ,   ],   # A
   [    ,   ,   ,   ,   ],   # B
   [    ,   ,   ,   ,   ],   # C
   [    ,   ,   ,   ,   ],   # D
   [    ,   ,   ,   ,   ],   # E
   [    ,   ,   ,   ,   ] ]  # F
```

Answer:

```
(N short for None)
graph =
   # A   B   C   D   E   F
[  [  N,  3,  N,  2,  6,  N],   # A
   [  3,  N,  N,  N,  2,  N],   # B
   [  N,  N,  N,  N,  4,  7],   # C
   [  2,  N,  N,  N,  2,  4],   # D
   [  6,  2,  4,  2,  N,  1],   # E
   [  N,  N,  7,  4,  1,  N] ]  # F
```

4. Use the following graph for the problem:



A. What nodes will we visit (and in what order) if we conduct BFS to find ‘H’ on the following graph? Start at node ‘A’. **Answer: ABCDEFGH**

B. What if we used DFS? **Answer: ABEFH**

Tractability

1. In one sentence, what does the algorithm for SubsetSum compute?

   Answer: SubsetSum checks if any combination of elements of a list sum to a particular value.

2. Is the problem it solves in P, NP, or NP-Complete?

   Answer: NP-Complete

3. Briefly explain your answer.

   Answer: It is in NP because searching all possible combinations of elements of a list is not possible in polynomial time (it is 2^n where n is the number of elements in the list). It is in NP-Complete, beacuse if you can solve SubsetSum, you could solve Boolean Satisfiability - the presence of the elements in the subset is equivalent to the 1's in a circuit. Therefore it is in NP-Complete. We don't know if it is in P, since we don't know if P = NP.

4. Assume that the problem that SubsetSum is solvable is in P. Can we then solve the boolean satisfiability problem in polynomial time?

Answer: Yes. SubsetSum is NP-Complete. We can reduce the boolean satisfiability to it in polynomial time. If we can reduce Boolean Satisfiability to SubsetSum in polynomial time and solve SubsetSum in polynomial time, then Boolean Satisfiability would be solvable in polynomial+polynomial = polynomial time.