CSC 110 EXAM 2 REVIEW GUIDE

NOTE: Individual exam October 19th and Group exam October 21st
Solutions Start on Page 18

TOPIC LIST:

1. Functions
2. Random
3. Graphics
4. Scope and Constants
5. Lists
6. For-loops
7. Topics from previous exams
8. Anything covered in class, the PAs, or the readings up until the exam

WHERE TO STUDY:

- Slides
- Textbook
- Prep problems
- This study guide

QUESTIONS?

- Online Office Hours
- Email your TA or instructor

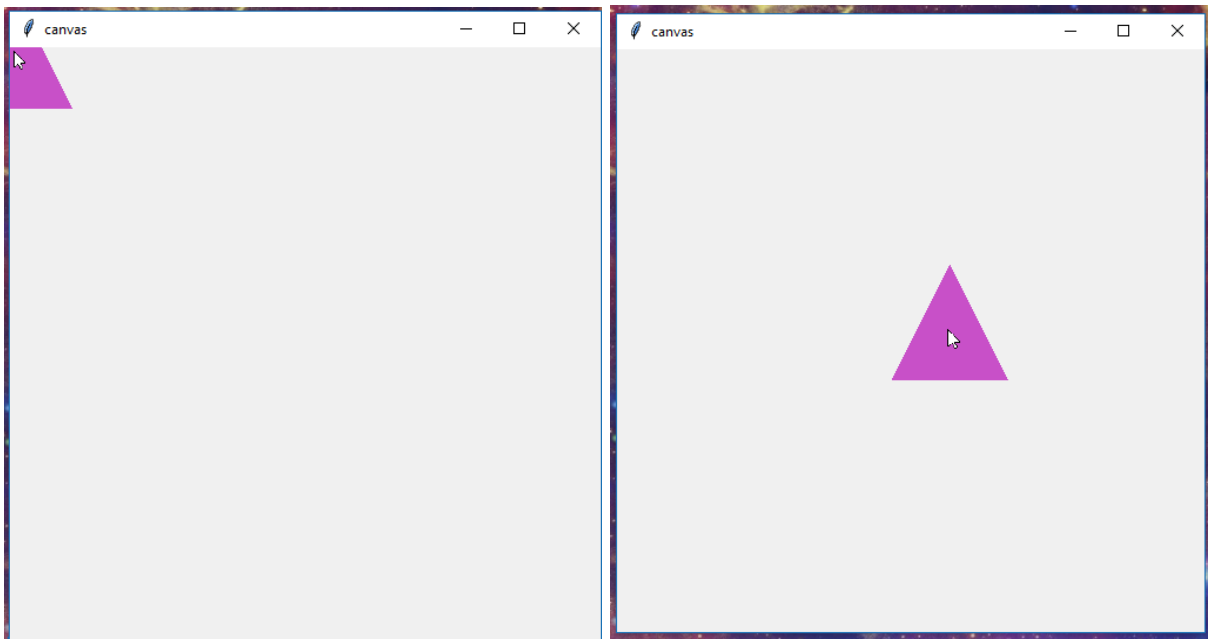
REVIEW PROBLEMS:

Note: these are not necessarily the questions that will be on your test. These questions were written collectively by the Section Leaders (Old TA's who haven't seen the test yet) and resemble what we think you need practice with for the test. Do not use this as your only resource for studying.

That being said, feel free to jump questions, and do the ones you think will help you the most. Ask questions on Discord, and attend a review session for additional clarification.

1) Write code and use graphics to draw a purple triangle that follows the mouse.
    - Center the mouse in the middle of the purple triangle.
    - The canvas should be 500x500.
    - Use `get_color_string()` with rgb to get the purple color.
    - Set `update_frame()` to 30.

    Example of output in the start and when mouse is in the middle of canvas:



2) Write a function that takes in an `x` and `y` coordinate and draws three squares (remember squares are defined by equal sides) of 3 different colors connected by their sides to the right with the coordinates denoting the leftmost squares top left corner.

3) Draw the output of the following code (be sure to label the colors):

```
from graphics import graphics

def main():
	gui = graphics(400, 600, "Art")

	gui.line(0, 0, 400, 600, "black")
	gui.line(0, 600, 400, 0, "black")

	for i in range(0, 300, 50):
		gui.ellipse(i + 50, i + 100, 25, 25, "green")

	gui.triangle(50, 600, 350, 600, 200, 350, "blue")
	gui.triangle(100, 600, 300, 600, 200, 400, "purple")

main()
```

4) Draw the output of the following code:

```
from graphics import graphics

def main():
	pos = [250, 50, 125, 250, 375, 250, 250, 50, 125, 250, 375, 250]
	screen = graphics(500, 300, “Problem 4”)
	while True:
		screen.clear()
		for i in range(0, len(pos), 4):
			screen.line(pos[i], pos[i+1], pos[i+2], pos[i+3])
		screen.update_frame(60)

main()
```

5) Given this code:

```
def squared():
      power = 2
      return val ** power

val = int(input('Enter a number: '))
val_squared = squared()
print(str(val) + ' squared is ' + str(val_squared))
```

   a) What is the output when the user inputs 5?
   b) How many global variables?
   c) How many local variables?

6) List all local variables, global variables, and global constants in the following piece of code. List them all in a way that differentiates them from each other (i.e. “x is a local variable”). List also the value of the above at the end of main of the global variables for the code that runs.

```
THE_BEST = True
best_amount = 3

def main():
      youre = True
      if youre == THE_BEST:
            is_best(5)
      else:
            not_best(.01)

def is_best(factor):
      global best_amount
      best_amount = factor * best_amount

def not_best(factor):
      global best_amount
      worst_amount = factor * best_amount
```

7) List some values from the following code that could be defined as constants. What would you name the constant, what would its value be, and where could you use it in the code?

```
mins = int(input("How long did you watch Netflix for? "))
shows = int(input("How many episodes/movies did you watch? "))

# avg number of minutes for each show
mins_per_show = mins / shows
# how many times would this fit in 24 hours?
shows_per_day = 24 * 60 / mins_per_show

if mins_per_show < 30:
      print("You were watching a sitcom!")
      print("You could watch", shows_per_day, \
            "episodes in a day.")
elif mins_per_show < 60:
      print("You were watching an hour-long drama!")
      print("You could watch", shows_per_day, \
            "episodes in a day.")
elif mins_per_show < 120:
      print("You were watching movies!")
      print("You could watch", shows_per_day, "movies in a day.")
else:
      print("You were watching Lord of the Rings!")
      print("You could watch", shows_per_day, "movies in a day.")
```

8) What is the minimum and maximum value that rand can evaluate to?

```
rand = random.randint(1, 99)
rand = rand * 3
```

9) What are the main differences between a for loop and a while loop? When would you want to use a for loop instead of a while loop?
10) Given the following scenarios indicate whether a for loop or while loop is **more** appropriate.
    a) Counting all the numbers between 1 and 5000
    b) Iterate over a list of integers until the value is not odd
    c) Removing all 5s from a list of integers
    d) Printing all letters in a string
    e) Running exactly 1 mile
    f) Running until you get tired

11) Write two different types of for loops that behave the same as the given while loop:

```
i = 0
lst = ['a', 'b', 'c', 'd']
while i < len(lst):
      print(lst[i])
      i += 1
```

12) Which of the following types in Python are mutable? Select all that apply
    (a): lists
    (b): strings
    (c): integers
    (d): floats
    (e): booleans

13) What would be the output of the following code? (Be careful)

```
input = [10, 50, 83, 20, 98, 105, 102]
for i in input:
      if i > 50:
            input.remove(i)
print(input)
```

14) What will the following code produce?

```
def function1(lst):
    i = 0
    new_list = []
    while (i < len(lst)):
        if (lst[i] % 2 == 1):
            new_list.append(lst[i])
        i += 1
    return new_list

def function2(a, b):
    i = 0
    new_list = []
    while (i < len(a)):
        if (a[i] % b == 0):
            new_list.append(a[i])
        i += 1
    return new_list

def main():
    lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    lst = function1(lst)
    lst = function2(lst, 3)
    print(lst)

main()
```

15) Write a function called `numbers(lst)` that will take in a list of integers and returns all the numbers that are divisible by 3 and 5. For example:

```
lst = [1, 14, 17, 15, 45, 19, 21]
numbers(lst) → [15, 45]
```

16) What is the output of the following program?

```
def change(a, b):
      for i in range (len(b)):
            b[i] = b[i] + str(a[i])

def main():
      a=[1, 2, 3, 4]
      b=['goldfish', 'dog', 'cat', 'hamster']
      change(a, b)
      print(b)

main()
```

17) What will this code output?

```
def main():
      a_list = ["Hello World", "5", "What's up?", "3", "7", "28",
      "To be or not to be"]
      new_list = create_new_list(a_list)
      print(new_list)

def create_new_list(list_to_change):
      new = []
      for i in range(len(list_to_change)):
            if (list_to_change[i].isnumeric() == False):
                  phrase = list_to_change[i]
                  new.append(phrase[5])
      return new

main()
```

18) Write a function `is_even` that takes `lst` as a parameter and returns `True` if every value in the list is an even value or `False` if there exists an odd number. For example:

`is_even([6, 22, 56, 92])` → `True`

`is_even([34, 52, 45, 96])` → `False`

19) Write a function `remove_odd(lst)` that takes `lst` as a parameter and remove all the odd index from the list and return the new list with all odd index removed from previous list.

20) Create a function, called `count_vowels`, that takes as parameters a list containing vowels, called `vowels`, and a list of strings, called `str_list`. This function should return a list the same length as `str_list` that holds the amount of vowels found in the word at the same index as it is in in `str_list`. First, try to get the problem working while using the **in** keyword. Then try and solve the problem without using the **in** keyword (Using the **in** keyword for your loop IS acceptable, such as for i **in** range). Which implementation is better? (Using the **in** keyword for your loop IS acceptable, such as for i **in** range. For example:
`count_vowels(vowels, ["Hello World", "What's up?", "3", ”” , "To be or not to be"]) → [3, 2, 0, 0, 6]`

21) Write another function called `sum_list(some_list)` that takes in a list as a parameter and returns all elements summed together. Note they do not have to be integers. For example:
`sum_list([1,2,3]) → 6`
`sum_list(['1','2','3']) → '123'`
`sum_list([True, True]) → 2`

22) Write a function called `list_even_lengths(string)` that takes a string as a parameter and counts how many times each word with an even number of characters appears in the original string. All characters in each word count toward the length (including punctuation), not just letters. You may use as many lists as you would like for auxiliary storage. You should *only* use for loops, not while loops. Your program should print each word and its count. For example, if the input string is:

`“Hello, how are you? Hello, I am doing well. I am too!”`

Output:
`Hello, : 2`

```
you? : 1
am : 2
too! : 1
```

23) Write a function called `letter_count()` that takes a list of strings and a letter as parameters. The function should count the amount of times that letter appears in all strings of the list. It then should print at the end: "`There were <count> <letter>'s.`". Where `<count>` is the count of letters found and `<letter>` is the argument letter passed in. For example:

```
letter_count(['abcd', 'tree', 'tech'], 'e') → "There were 3 e's."
letter_count(['the', 'world', 'is', 'beautiful'], 'u') → "There were
2 u's."
letter_count(['CSC', '110'], 'z') → "There were 0 z's."
```

24) Write a function `random_output(items)` that takes in a list and prints out the contents of that list in a random order each time it is run. For example, if the list `['1', '2', '3']` was passed in as an argument, the output could be any of the following:

```
random_output(['1', '2', '3']) → 1 3 2
random_output(['1', '2', '3']) → 2 3 1
random_output(['1', '2', '3']) → 3 2 1
random_output(['1', '2', '3']) → 1 2 3
```

Note that every element should be printed once and only once.

Hint: remove the element after you have printed it.

25) Write a method `longest_substring()` that takes a string as a parameter and returns the length of the longest substring without repeating characters. For example:

```
longest_substring( "abcabcbb") → 3
```

In the string “abcabcbb” which is passed as a parameter “abc” is the longest substring without any repetition of characters.

26) Write a function score_summaries(scores) that takes a list of strings with score information for the results of various sporting events and their respective team/city information and returns a list of strings, each of which notates which team won the game, what the margin of victory was, or if the game resulted in a tie. For example

```
inputs = [“STL 8 - NYY 2”, “CLE 5 - DET 3”, “MIA 2 - HOU 10”]
inputs2 = [“LAL 101 - “BOS 106“, “ARI 6 - SEA 6”, “PHI 41 - NE 33” ]

results = score_summaries(inputs)
results2 = score_summaries(inputs2)

print(results)
print(results2)
```

Output:

```
[“STL wins by 6 points”, “CLE wins by 2 points”, “HOU wins by 8 points”]
[“BOS wins by 5 points”, “ARI vs SEA ended in a tie”, “PHI wins by 8 points”]
```

27) Write a function only_alphabetical_order that takes in a string parameter, called sentence, and splits it into a list of its component words (if there is just one word, just that word is in the list). Your function should print out a list of only the words that come later in the alphabet than the one before it. This means that if the first letter of a word comes earlier in the alphabet than the the word before it, it will not appear in the final list. You are only allowed to use one list and a while loop. Assume all words in the input start with a letter. For example:

```
only_alphabetical_order(“I am a computer science major.”)  →  [“I”, “am”, “computer”, “science”]
```

```
only_alphabetical_order(“How much has he improved today?”) →
[[“How”, “much”, “improved”, “today?”]
```

28) What output will we get from the following code?

```
def main():
    a = "hello"
    b = "world"

    c, d = funct_1(a, b)
    print(c, d)

    b, a = funct_2(c, 35)
    print(a, b)

def funct_1(b, a):
    i = 0
    c = ""

    while (i < len(b)):
        c += b[i : len(b)]
        return c, a[2 : 4]
        i += 1

def funct_2(a, b):
    i = 0
    j = 2
    b = 100

    string = ""
     # string =
    while (i < len(a) - 1):
        string += a[i : j]
```

```
            i += 1
            j += 1

        return b, string

    main()
```

29) At a sale, each miniature bridge costs $1.
    Each customer will only buy one miniature bridge and will pay with dollar bills. You must provide the correct change to each customer, so that the net transaction is that the customer pays $1.
    Note that you don't have any change in hand at first.

    Write a function called `bridge_sale` that accepts a list, called `bills` that represent the customers who are standing in line and order one at a time(in the order of `bills`) with the number being how many dollar bills the customers pay with. Return `True` if and only if you can provide every customer with correct change. Otherwise, return `False`. For example:

    `bridge_sale([1, 1, 1, 2, 4]) → True`

    `bridge_sale([1, 1, 2]) → True`

    `bridge_sale([2, 2]) → False`

    `bridge_sale([1, 1, 4]) → False`

30) Write a function called `remove_duplicate(num_list)` that can remove all of the duplicates in the `num_list` and return the edited `num_list`. For example:

    `remove_duplicate([1,2,4,6,3,1,3,5,6,7,5]) → [1, 2, 4, 6, 3, 5, 7]`
    `remove_duplicate(["the","the","the","the","the","the"]) →['the']`

31) Write a function named `hottest_month()` that takes as a parameter a string of months and numbers each separated by whitespace. An example string would look like

this: “`November 23 December 20 January 25`”. Each month is followed by a number that represents the average temperature of that month. The function should first create a list containing the months and temperatures (use `split()`), then iterate through the list to determine which month was the coldest month (lowest temp) and which month was the hottest month (highest temp). The function should then print out the lowest and hottest month. For example:

```
hottest_month(“January 25 February 27 March 26 April 30 May 34 June 40 July 44”)
```

This should print out:

```
Coldest month: January, 25
Hottest month: July, 44
```

Note: two months will never have the same average temperature.

32) Write a function, `remove_letter` that takes a list of strings as a parameter as well as a letter between a and z. The function should return a version of the list without the strings that start with the letter provided as a parameter.

33) Write the output of the following program:

```
def main():
      a = “you”
      b = “us”
      c = “me”
      calculus(c, b, a)
      b = “five”
      a = “two”
      calculus(a, b, c)
      c = “seven”
      calculus(b, c, a)

def calculus(a, b, c):
      print(c, “plus”, a, “equals”, b)
```

```
main()
```

34) Find the output for the following code:

```
def calculate(n, other):
      if n < other:
            n += 3
            other -= 4
      if other < n:
            n -= 2
            other = n * 2
      else:
            other -= 1
      print(n, other)
```

a) `calculate(6, 3)`
b) `calculate(17, 19)`
c) `calculate(4, 4)`
d) `calculate(36, 2)`

35) Write the code for a function `currency_converter` that takes two floats as parameters, one representing the amount of one currency, and a second representing the exchange rate of the first currency into a second currency (i.e. the price of the first currency in terms of the second currency). The function should also take two string parameters representing the type of currency passed in as well as the type of currency which the first is being exchanged to (e.g. 'USD', 'CAD', 'AUD', 'EUR'). Your function should first make sure that the currency amount entered is non negative and that the exchange rate entered is also non negative. The function should then print out the representation of the first currency in terms of the second currency.

Example:

```
currency_converter(3.2, 0.87, 'USD', 'EUR')
3.2 USD is equivalent to approximately 2.78 EUR.

currency_converter(4.0, 82.39, 'CAD', 'JPY')
4.0 CAD is equivalent to approximately 329.56 JPY.
```

```
currency_converter(100, 0.040, 'MXN', 'GBP')
100 MXN is equivalent to approximately 4.00 GBP.

currency_converter(40, -50.23, 'AUD', 'USD')
Please enter a non-negative exchange rate.

currency_converter(-200, 30838.59, 'BTC', 'SEK')
Currency entered must NOT be a negative amount.
```

36) Write a function `my_isnumeric(s)` that behaves similarly to the `isnumeric()` function we have used in class. Your function should take a string as a parameter and determine if all characters of the string are digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). If they are all digits, your function should print the message "`[string] is numeric.`" For any non-digit character encountered, your function should print "`[character] is not a number.`" For example:

```
my_isnumeric(“87365”) prints:
87365 is numeric.

my_isnumeric(“hi”) prints:
h is not a number.
i is not a number.

my_isnumeric(“h3110 w0r1d”) prints:
h is not a number.
w is not a number.
  is not a number.
r is not a number.
d is not a number.
```

37) Write the code for function `convert_time(time)` that takes in a string of military time (ex: "16:42") as a parameter and prints out "`The time is`" followed by the 12 hour form of the given time with AM or PM. Keep in mind that midnight is portrayed as "00" in military time.

38) Write a function `exist_midpoint` that accepts three integers as a parameter and prints "`True`" if one of the integers is the midpoint between the other two integers.If midpoint does not exist you should print "`False`".
Example:

```
exist_midpoint(4,6,8)      ->    “True”
exist_midpoint(7,7,7)      ->    “True”

exist_midpoint(3,1,3)      ->    “False”
```

```
exist_midpoint(2,18,19)    ->    “False”
```

39) Write a function `is_even` that accepts an integer as parameter and prints “`Even`” if the integer is even and “`Odd`” if the integer is not even i.e the integer is odd.
Example:

```
is_even(8)       ->    “Even”
is_even(64)      ->    “Even”
is_even(9)       ->    “Odd”
is_even(7)       ->    “Odd”
```

SOLUTIONS

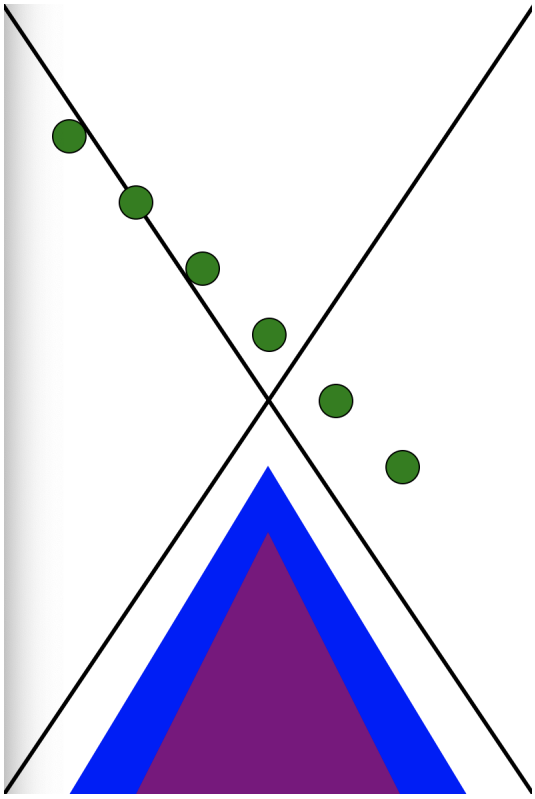
1)

```
gui = graphics(500, 500, "canvas")
red = 220
green = 110
blue = 210
color = gui.get_color_string(red, green, blue)
while True:
    x = gui.mouse_x
    y = gui.mouse_y
    gui.clear()
    # not perfectly in middle but very close
    gui.triangle(x - 25, y + 25, x + 25, y + 25, x, y - 25, color)
    gui.update_frame(30)
```

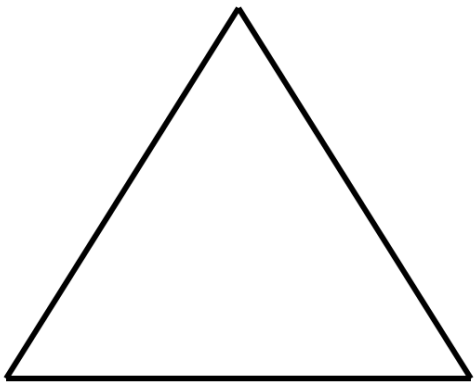
2)

```
def three_squares(xcoord, ycoord):
    gui = graphics(500, 500, "Three Squares")
    gui.rectangle(xcoord, ycoord, 50, 50, "green")
    gui.rectangle(xcoord + 50, ycoord, 50, 50, "yellow")
    gui.rectangle(xcoord + 100, ycoord, 50, 50, "red")
```

3)



4)



5)

a) 5 squared is 25

b) Two global variables, val and val_squared
c) One local variable

6)

Local variables: youre, worst_amount, factor
Global variables: best_amount
Global contants: THE_BEST

AFTER MAIN RUNS VALUE OF GLOBAL VARIABLES:
best_amount = 15

7)
'Mins' could be a global constant, could name it "MINUTES", 'shows' could also be a constant and we could call it NUM_SHOWS.

You also if you wanted, could make the hardcoded values in the if statements (30, 60, 120), they could be named 'SHORT', 'MEDIUM', or 'LONG'.

8)

The minimum value would b 1 * 3 which would be 3, and the maximum value would be if 99 is chosen, which would be 99 * 3, or 297.

9)

A while loop is a good loop to use if you are going to be going through a list and deleting or adding elements. Because if you delete something from your list, you can simply not increment i so that nothing gets skipped. If you try and remove and add things in a for loop, things can get very tricky!

There are two types of for loops and they each have their own merits. The first one uses a range of numbers, such as for i in range(0,10), which will start i at 0 and go up to 9 because the 10 is

exclusive. This loop works really well when you are doing some comparisons on a list and need to grab multiple elements from the list at the same time. Because the other type of for loop does not allow you to do that. The last type of for loop is a loop such as
for item in list. This is a good loop to use when you don't really care about indexing and you want to perform the same task or checks on each item in the list. This list though has no indexing, so 'item' in the for loop represents an actual item from the list.

10)

a) For loop, it can increment i for you all the way up to 5000 so you don't have to increment i yourself, saving time.
b) While loop, look at the keyword until, that is essentially giving you a condition to stop at, which should hint a while loop.
c) While loop, removing items from a list is best done with a while loop, since you can control if i gets incremented or decremented correctly.
d) For loop, quickest way to accomplish this.
e) For loop, its running an exact amount of time/distance, so no need to declare your indexes in a while loop.
f) Opposite to answer e, this would be a while loop because you want to do something until a condition is met, in this case, running UNTIL you are tired.

11)

```
# First type
for i in range(len(lst)):
    print(lst[i])

# Second type
for letter in lst:
    print(letter)
```

12) a

13)

[10, 50, 20, 105]

14)

[3, 9]

15)

One possible answer

```
def numbers(lst):
    new_list = []
    for number in lst:
        if number % 3 == 0 and number % 5 == 0:
            new_list.append(number)

    return new_list
```

16)

[“goldfish1”, “dog2”, “cat3”, “hamster4”]

17)

[“ “, “s”, “ “]

18)

```
# one possible solution
def is_even(lst):
    for number in lst:
        if number % 2 == 1:
            return False
    return True
```

19)

One possible Solution

```
def remove_odd(lst):
    new_list = []
```

```
    for i in range(len(lst)):
        if i % 2 == 0:
            new_list.append(lst[i])
    return new_list
```

20)

```
# Solution WITH using in keyword
def count_vowels(vowels, str_list):
    temp_list = [0] * len(str_list)
    for i in range(len(str_list)):
        word = str_list[i]
        for j in range(len(word)):
            if word[j] in vowels:
                temp_list[i] += 1
    return temp_list
```

```
# Solution WITHOUT using in keyword
def count_vowels(vowels, str_list):
    temp_list = [0] * len(str_list)
    for i in range(len(str_list)):
        word = str_list[i]
        # grab each letter in each word
        for j in range(len(word)):
            letter = word[j]
            # compare each vowel in vowel list to letter
            for k in range(len(str_list)):
                if vowels[k] == letter:
                    temp_list[i] += 1
    return temp_list
```

21)

More efficient solution:
```
def sum_list(some_list):
    return sum(some_list)
```

Alternate option:
```
def sum_list(some_list):
    # To get the type to add or we could get errors
    first = some_list[0]
    for i in range(1, len(some_list)):
        first += some_list[i]
    return first
```

22)

One possible solution
```
def list_even_lengths(string):
    word_count = []
    word_list = string.split()
    for word in word_list:
        if len(word) % 2 == 0:
            if word not in word_count:
                word_count.append(word)
                word_count.append(0)
            index = word_count.index(word)
            word_count[index + 1] += 1
       for i in range(0,len(word_count) - 1, 2):
           print(word_count[i], ":", word_count[i + 1])
```

23)

One possible solution
```
def letter_count(lst, letter):
    total = 0
    for word in lst:
        for curr_letter in word:
            if curr_letter == letter:
                total += 1
    print("There were", total, letter + "'s.")
```

24)

One possible solution

```
def random_output(items):
    while len(items) > 0:
        num = random.randint(0, len(items) - 1)
        current = items.pop(num)
        print(current + " ", end="")
```

25)

One possible solution

```
def longest_substring(string):
    longest = 0
    seen = ""
    for i in range(len(string)):
        # if we've seen the letter before
        if string[i] in seen:
            if len(seen) > longest:
                longest = len(seen)
            seen = string[i]
        else:
            seen += string[i]
    return longest
```

26)

One possible solution

```
def score_summaries(scores):
    temp = []
    for line in scores:
        # split each string in the list into a sublist.
        info = line.split(" - ")
        # split each sublist on whitespace to access both teams info
        first = info[0].split()
        second = info[1].split()
        name_1 = first[0]
        name_2 = second[0]
        points_1 = int(first[1])
        points_2 = int(second[1])
        if points_1 > points_2:
```

```
            string = name_1 + " wins by " + str(points_1 - points_2) + " points"
        elif points_1 < points_2:
            string = name_2 + " wins by " + str(points_2 - points_1) + " points"
        else:
            string = name_1 + " vs " + name_2 + " ended in a tie"
        temp.append(string)
    return temp
```

27)

```
# One possible solution
def only_alphabetical_order(sentence):
    word_list = sentence.split()
    i = 1
    while i < len(word_list):
        if word_list[i][0] <= word_list[i-1][0]:
            word_list.pop(i)
        else:
            i += 1

    print(word_list)
```

28)

hello rl

heellllo 100

29)

```
# One possible solution
def bridge_sale(bills):
```

```
    current_change = 0
    for i in range(len(bills)):
        amount = bills[i]
        if amount - current_change > 1:
            return False
        current_change += amount
    return True
```

30)

```
# One possible solution
def remove_duplicate(num_list):
    seen = []
    i = 0
    while i < len(num_list):
        current = num_list[i]
        if current in seen:
            num_list.pop(i)
        else:
            i += 1
        seen.append(current)
    return num_list
```

31)

```
# One possible solution
def hottest_month(string):
    info = string.split()
    cold_month = info[0]
    cold_temp = info[1]
    hot_month = info[0]
    hot_temp = info[1]
    for i in range(2, len(info), 2):
        curr_month = info[i]
        curr_temp = info[i + 1]
        if curr_temp < cold_temp:
            cold_month = curr_month
            cold_temp = curr_temp
```

```
        elif curr_temp > hot_temp:
            hot_month = curr_month
            hot_temp = curr_temp
    print("Coldest month: " + cold_month + ", " + cold_temp)
    print("Hottest month: " + hot_month + ", " + hot_temp)
```

32)

One possible solution

```
def remove_letter(lst, letter):
    new_list = []
    for word in lst:
        if word[0] != letter:
            new_list.append(word)
    return new_list
```

33)
you plus me equals us
me plus two equals five
two plus five equals seven

34)

a) 4 8
b) 18 36
c) 4 3
d) 34 68

37) ONE POSSIBLE SOLUTION

```
def currency_converter(currency_1_amount, exchange_rate, currency_type1, currency_type2):
  if (exchange_rate < 0):
    print ("Please enter a non-negative exchange rate.")
  elif (currency_1_amount < 0):
    print ("Currency entered must NOT be a negative amount.")
  else:
    currency_2_amount = currency_1_amount * exchange_rate
    ans = "{current_1} {type_1} is equivalent to approximately {current_2:.2f} {type_2} ."
    print (ans.format(current_1=currency_1_amount, type_1=currency_type1,
      current_2=currency_2_amount, type_2=currency_type2))
```

36) ONE POSSIBLE SOLUTION

```
def my_isnumeric(str):
    if str.isnumeric():
        print (str,'is numeric.')
    else:
        i = 0
        while i < len(str):
            if not str[i].isnumeric():
                print (str[i], "is not a number.")
            i += 1
```

37) ONE POSSIBLE SOLUTION

```
def convert_time(time):
    if (int(time[:2])<=12):
        print ('The time is',time,'AM')
    else:
        print ('The time is',"{}:{}".format(int(time[:2])-12,int(time[3:])),'PM')
```

38) ONE POSSIBLE SOLUTION

```
def exist_midpoint(num1, num2, num3):
    if (num1+num3)/2 == num2 or (num1+num2)/2 == num3 or (num2+num3)/2 == num1:
        print (True)
    else:
        print (False)
```

39) ONE POSSIBLE SOLUTION

```
def is_even(num):
    if (num % 2) == 0:
        print ('Even')
    else:
        print ('Odd')
```