Question 2 Code:

```
def print_abacus(num):
  '''
    Print_Abacus takes in a string of a number
    Then it goes through and adjust the default abacus
    Finally it prints the new abacus
  '''
  print(num + " is represent by: ")
  sizeOf = len(num)
  bottomRowEmpty = [5,6,7,8]
  abacus = [['o','o','o','o','o','o','o','o','o','o','o','o','o'],
            ['o','o','o','o','o','o','o','o','o','o','o','o','o'],
            [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '],
            [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '],
            ['_','_','_','_','_','_','_','_','_','_','_','_','_'],
            [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '],
            [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '],
            [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '],
            [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '],
            ['o','o','o','o','o','o','o','o','o','o','o','o','o'],
            ['o','o','o','o','o','o','o','o','o','o','o','o','o'],
            ['o','o','o','o','o','o','o','o','o','o','o','o','o'],
            ['o','o','o','o','o','o','o','o','o','o','o','o','o']]
  for i in range(sizeOf):
    colIndex = (13-sizeOf)+i
    numSpacesToMove = int(num[i])
    if (numSpacesToMove >= 5):
      # If we have more than 5, adjust the top row first
      abacus[3][colIndex] = 'o'
      abacus[1][colIndex] = ' '
      numSpacesToMove=numSpacesToMove-5
      # Then we change the bottom row
      # To keep track of how many bottom rows we've changes
      bottomPlaced = 0
      while (numSpacesToMove > 0):
        # Parse the next empty index from our list of bottom row
        rowIndex = bottomRowEmpty[bottomPlaced]
        abacus[rowIndex][colIndex] = 'o'
        bottomPlaced+=1
        numSpacesToMove-=1
      ''' After we place the beads in the right spot, we put empty spots
      in the used beads spots '''
      for x in range(9,9+bottomPlaced):
        abacus[x][colIndex] = ' '
    else:
      # If we have less than 5, we only adjust the bottom row
```

```
      # To keep track of how many bottom rows we've changes
      bottomPlaced = 0
      while (numSpacesToMove > 0):
        # Parse the next empty index from our list of bottom row
        rowIndex = bottomRowEmpty[bottomPlaced]
        abacus[rowIndex][colIndex] = 'o'
        bottomPlaced+=1
        numSpacesToMove-=1
      ''' After we place the beads in the right spot, we put empty spots
      in the used beads spots '''
      for x in range(9,9+bottomPlaced):
        abacus[x][colIndex] = ' '
  # Once we finish editing the abacus, we print out the abacus
  for a in range(len(abacus)):
    row = str(abacus[a])
    print(row)


def add_abacus(num1, num2):
  print_abacus(str(num1))
  print('')
  print_abacus(str(num2))
  print('')
  summation = num1 + num2
  print_abacus(str(summation))


# 2a
print('QUESTION 2A: ')
add_abacus(54321, 90678)
# 2b
print('QUESTION 2B: ')
add_abacus(559876543210, 27623428724)
# 2c
print('QUESTION 2C: ')
add_abacus(127002343627, 23412876241)
```

Question 2 Output:

```
(base) CJs-MacBook-Pro:~ cj_hess510$ python -u "/Users/cj_hess510/Desktop/MST Computer Science/CS 2200/HW1/hw1-abacus.py"
QUESTION 2A:
54321 is represent by:
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ']
['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', 'o', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', ' ', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', ' ', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', 'o', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', 'o', 'o']

90678 is represent by:
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', 'o', 'o', 'o']
['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', ' ', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', ' ', ' ', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', 'o', ' ', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', 'o', 'o', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', 'o', 'o', 'o']

144999 is represent by:
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', 'o']
['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', 'o', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', 'o', 'o', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', ' ', ' ', ' ', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', ' ', ' ', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', ' ', ' ', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', ' ', ' ', ' ']
```

```
QUESTION 2B:
559876543210 is represent by:
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
['o', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', ' ', ' ', ' ']
['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
[' ', ' ', ' ', 'o', 'o', 'o', 'o', ' ', 'o', 'o', 'o', 'o', ' ']
[' ', ' ', ' ', 'o', 'o', 'o', ' ', ' ', 'o', 'o', 'o', ' ', ' ']
[' ', ' ', ' ', 'o', 'o', ' ', ' ', ' ', 'o', 'o', ' ', ' ', ' ']
[' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ']
['o', 'o', 'o', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ', 'o']
['o', 'o', 'o', ' ', ' ', ' ', 'o', 'o', ' ', ' ', ' ', 'o', 'o']
['o', 'o', 'o', ' ', ' ', 'o', 'o', 'o', ' ', ' ', 'o', 'o', 'o']
['o', 'o', 'o', ' ', 'o', 'o', 'o', 'o', ' ', 'o', 'o', 'o', 'o']

27623428724 is represent by:
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
['o', 'o', 'o', ' ', ' ', 'o', 'o', 'o', 'o', ' ', ' ', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', 'o', 'o', ' ', ' ', ' ', ' ', 'o', 'o', ' ', ' ']
['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
[' ', ' ', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
[' ', ' ', 'o', 'o', ' ', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', ' ', 'o', ' ', ' ', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ', 'o']
['o', 'o', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
['o', 'o', ' ', ' ', 'o', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
['o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', 'o', ' ', 'o', 'o', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', 'o', 'o', 'o', ' ']

587499971934 is represent by:
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
['o', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ', 'o', ' ', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', 'o', 'o', 'o', ' ', 'o', 'o', 'o', 'o', ' ', 'o', ' ', ' ']
['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
[' ', ' ', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
[' ', ' ', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', 'o', 'o']
[' ', ' ', 'o', ' ', 'o', 'o', 'o', 'o', ' ', ' ', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', 'o', 'o', 'o', 'o', ' ', ' ', 'o', ' ', 'o']
['o', 'o', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
['o', 'o', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ']
['o', 'o', ' ', 'o', ' ', ' ', ' ', ' ', 'o', 'o', ' ', ' ', ' ']
['o', 'o', 'o', 'o', ' ', ' ', ' ', ' ', 'o', 'o', ' ', 'o', ' ']
```

```
QUESTION 2C:
127002343627 is represent by:
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
['o', 'o', 'o', ' ', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', 'o']
['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
[' ', 'o', 'o', 'o', ' ', ' ', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
[' ', ' ', 'o', 'o', ' ', ' ', 'o', 'o', 'o', 'o', ' ', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', 'o', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ']
['o', ' ', ' ', ' ', 'o', 'o', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
['o', 'o', ' ', ' ', 'o', 'o', ' ', ' ', ' ', ' ', 'o', ' ', ' ']
['o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', ' ', 'o', 'o', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o', 'o', 'o', 'o']

23412876241 is represent by:
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
['o', 'o', 'o', 'o', 'o', 'o', 'o', ' ', ' ', ' ', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o', 'o', 'o', ' ', ' ', ' ']
['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
[' ', ' ', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
[' ', ' ', 'o', 'o', 'o', ' ', 'o', 'o', 'o', ' ', 'o', 'o', ' ']
[' ', ' ', ' ', 'o', 'o', ' ', ' ', 'o', ' ', ' ', ' ', 'o', ' ']
[' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' ']
['o', 'o', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
['o', 'o', ' ', ' ', ' ', 'o', ' ', ' ', ' ', 'o', ' ', ' ', 'o']
['o', 'o', 'o', ' ', ' ', 'o', 'o', ' ', 'o', 'o', 'o', ' ', 'o']
['o', 'o', 'o', 'o', ' ', 'o', 'o', 'o', 'o', 'o', 'o', ' ', 'o']

150415219868 is represent by:
['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o']
['o', 'o', ' ', 'o', 'o', 'o', ' ', 'o', 'o', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', 'o', ' ', ' ', ' ', 'o', ' ', ' ', 'o', 'o', 'o', 'o']
['_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
[' ', 'o', ' ', ' ', 'o', 'o', ' ', 'o', 'o', 'o', 'o', 'o', 'o']
[' ', ' ', ' ', ' ', 'o', ' ', ' ', 'o', ' ', 'o', 'o', ' ', 'o']
[' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ', 'o', 'o', ' ', 'o']
[' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' ']
['o', ' ', 'o', 'o', ' ', ' ', 'o', ' ', ' ', ' ', ' ', ' ', ' ']
['o', 'o', 'o', 'o', ' ', 'o', 'o', ' ', 'o', ' ', ' ', 'o', ' ']
['o', 'o', 'o', 'o', ' ', 'o', 'o', 'o', 'o', ' ', ' ', 'o', ' ']
['o', 'o', 'o', 'o', ' ', 'o', 'o', 'o', 'o', ' ', 'o', 'o', 'o']
```

Question 3 Code:

```python
def uncompact_subtractives(num):
  '''
    This function is used for uncompacting subtractives by converting the
    roman numeral word into an integer and then converting that integer into
    roman numerals without subtractives
    Returning a roman numeral word without subtractives
  '''
  rom_val = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
  int_val = 0
  # Convert the roman numerals to a number
  for i in range(len(num)):
    if i > 0 and rom_val[num[i]] > rom_val[num[i-1]]:
      # Subtractives
      int_val += rom_val[num[i]] - 2 * rom_val[num[i-1]]
    else:
      int_val += rom_val[num[i]]

  # Break down what symbols need without subtractives
  symbolsNeeded = {'I': 0, 'V': 0, 'X': 0, 'L': 0, 'C': 0, 'D': 0, 'M': 0}
  while int_val > 1000:
```

```
    symbolsNeeded['M']+=1
    int_val-=1000
  while int_val > 500:
    symbolsNeeded['D']+=1
    int_val-=500
  while int_val > 100:
    symbolsNeeded['C']+=1
    int_val-=100
  while int_val > 50:
    symbolsNeeded['L']+=1
    int_val-=50
  while int_val > 10:
    symbolsNeeded['X']+=1
    int_val-=10
  while int_val > 5:
    symbolsNeeded['V']+=1
    int_val-=5
  while int_val > 0:
    symbolsNeeded['I']+=1
    int_val-=1

  # Write the new roman numeral word sorted from largest to smallest
  newRomanWord = ""
  while symbolsNeeded['M'] > 0:
    newRomanWord+='M'
    symbolsNeeded['M']-=1
  while symbolsNeeded['D'] > 0:
    newRomanWord+='D'
    symbolsNeeded['D']-=1
  while symbolsNeeded['C'] > 0:
    newRomanWord+='C'
    symbolsNeeded['C']-=1
  while symbolsNeeded['L'] > 0:
    newRomanWord+='L'
    symbolsNeeded['L']-=1
  while symbolsNeeded['X'] > 0:
    newRomanWord+='X'
    symbolsNeeded['X']-=1
  while symbolsNeeded['V'] > 0:
    newRomanWord+='V'
    symbolsNeeded['V']-=1
  while symbolsNeeded['I'] > 0:
    newRomanWord+='I'
    symbolsNeeded['I']-=1

  return newRomanWord
```

```python
def add_romanNums(num1, num2):
  # Take out the subtractives
  num1 = uncompact_subtractives(num1)
  num2 = uncompact_subtractives(num2)
  # Concatenate
  combined = num1 + num2
  combined = uncompact_subtractives(combined)

  return combined

def subtract_romanNums(num1, num2):
  # Take out the subtractives
  num1 = uncompact_subtractives(num1)
  num2 = uncompact_subtractives(num2)

  num1_dict = {'I': 0, 'V': 0, 'X': 0, 'L': 0, 'C': 0, 'D': 0, 'M': 0}
  num2_dict = {'I': 0, 'V': 0, 'X': 0, 'L': 0, 'C': 0, 'D': 0, 'M': 0}


  for i in num1:
    num1_dict[i]+=1
  for j in num2:
    num2_dict[j]+=1

  newNum = ''

  for x in ['M','D','C','L','X','V','I']:
    diff = num1_dict[x] - num2_dict[x]
    if (diff > 0):
      newNum+=(diff*x)

  return newNum



# 3a
print('QUESTION 3A: ')
print(add_romanNums(add_romanNums('CXCV','XXXI'), 'LXXXVIII'))
# 3b
print('QUESTION 3B: ')
print(subtract_romanNums(subtract_romanNums('CCCXCV','CXV'),'LXX'))
# 3c
print('QUESTION 3C: ')
print(add_romanNums('CCMCCCII', 'MLCLCIII'))
```

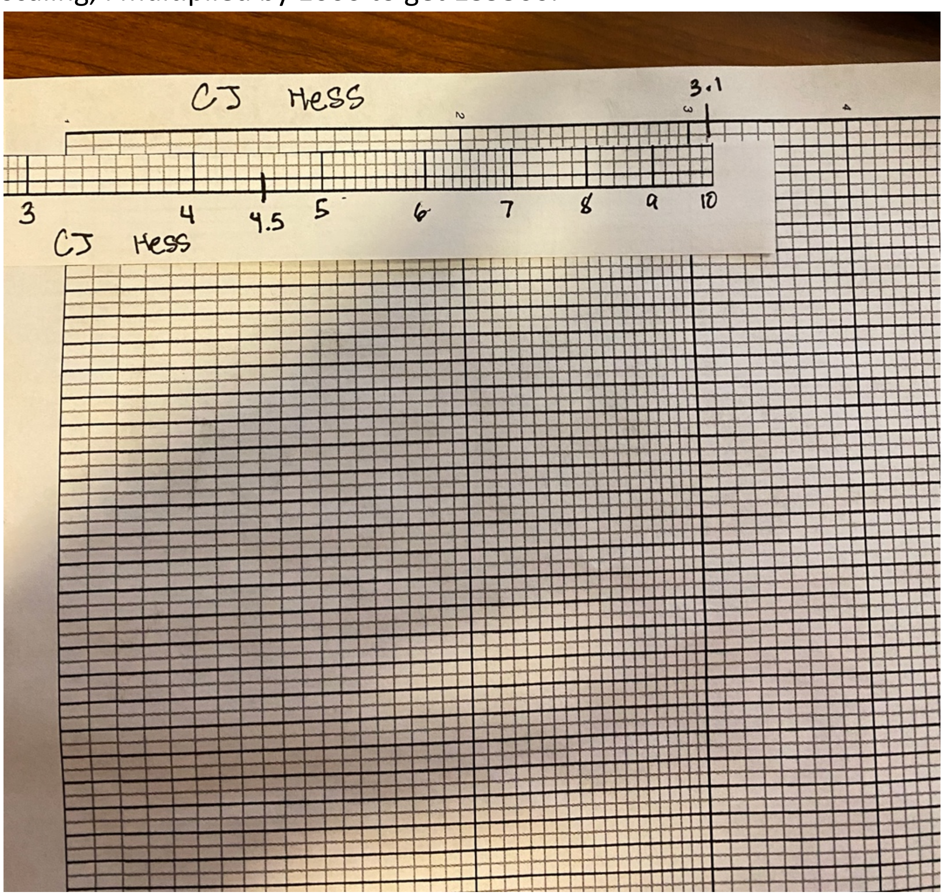
Question 3 Output:

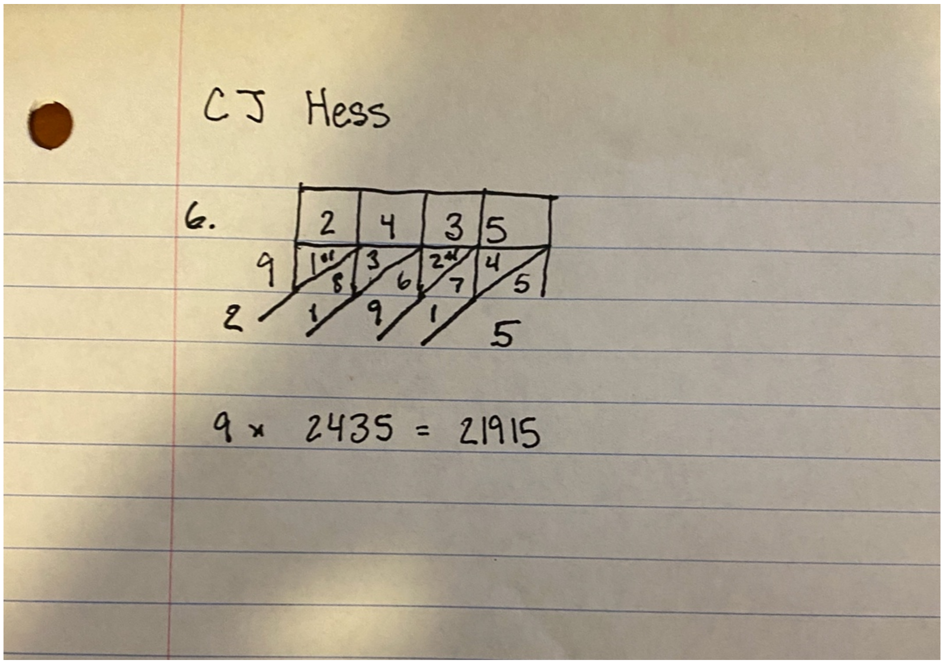
```
TERMINAL   PROBLEMS   OUTPUT   DEBUG CONSOLE

(base) CJs-MacBook-Pro:~ cj_hess510$ python -u "/Users/cj_hess510/Desktop/MST Computer Science/CS 2200/HW1/hw1-romans.py"
QUESTION 3A:
CCCVIIIII
QUESTION 3B:
CCX
QUESTION 3C:
MMCCCCIII
(base) CJs-MacBook-Pro:~ cj_hess510$
```

Question 4a-c ^

Question 4d: My initial thought for approaching multiplying two big roman numeral numbers was of confusion. The multiplication method that we use unfortunately could not work for roman numerals and therefore it could not be solved efficiently. The processing time of adding that many roman numerals x number of times would exponentially increase as the amount of digits increased. I do not believe that the Romans could have had a great scientific revolution with the use of only Roman Numerals to work with.

5. I changed the scales by some magnitude of 10 to make them fit on the sheet. So instead of 44.98, I used 4.5 by rounding and dividing by 10. For 3127, I used 3.1 by dividing by 1000 and rounding. With how the slide aligned, I did 4.5 * 3.1*10 = 1.395. Then accounting for the scaling, I multiplied by 1000 to get 139500.





7a. The ENIAC was very big and was able to be walked through. It was great for watching processes run and lights execute.
7b. 20 words
7c. Subroutines were not common at all in ENIAC programming due to the extremely limited storage
7d. Ballistic Research Laboratories
7e. The cards would soak up moisture, so they needed to be able to dry out.