**Student Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*This exam is double-sided. Be sure to pay attention to all instructions.*

Q1. [**32 pts.**] Compute the value of these expressions assuming the following variable values.

vals = [7, 3, 1, 3, 5, 2]  
x = 5  
s = 'Year 2014'  
y = 3

|  |  |  |
| --- | --- | --- |
|  | Expression | Expression Value |
| a) | s.split() | ['Year','2014'] |
| b) | vals[1:2]+vals[4:5] | [3,5] |
| c) | vals[x]\*y | 6 |
| d) | int(s[6:]) | 14 |
| e) | vals.index(y,x-y) | 3 |

|  |  |  |
| --- | --- | --- |
|  | After executing… | Then vals is equal to … |
| f) | del vals[:y] | [3,5,2] |
|  | After executing… | Then x is equal to … |
| g) | x -= vals[x] + vals[y] | 0 |
|  | After executing… | Then vals is equal to … |
| h) | for i in range(y,x):  vals[i] = vals[i–1] – vals[i–2] | [7,3,1,–2,–3,2] |

For the following expressions, write the appropriate value after each individual statement executes. **For each question, start with the variables defined above**.

**Student Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*This exam is double-sided. Be sure to pay attention to all instructions.*

Q1. [**32 pts.**] Compute the value of these expressions assuming the following variable values.

vals = [9, 6, 1, 3, 9, 2]  
x = 5  
s = 'Year 2014'  
y = 3

|  |  |  |
| --- | --- | --- |
|  | Expression | Expression Value |
| a) | vals[1:2]+vals[4:5] | [6,9] |
| b) | s.split() | ['Year','2014'] |
| c) | int(s[6:]) | 14 |
| d) | vals.index(y,x-y) | 3 |
| e) | vals[x]\*y | 6 |

|  |  |  |
| --- | --- | --- |
|  | After executing… | Then vals is equal to … |
| f) | del vals[:y] | [3,9,2] |
|  | After executing… | Then x is equal to … |
| g) | x -= vals[x] \* vals[y] | -1 |
|  | After executing… | Then vals is equal to … |
| h) | for i in range(y,x):  vals[i] = vals[i–1] – vals[i–2] | [9,6,1,-5,-6,2] |

For the following expressions, write the appropriate value after each individual statement executes. **For each question, start with the variables defined above**.

|  |
| --- |
| Q4 Expected Output |
| >>> checksum([1,2,3,6])  **True** >>> checksum([2,2,6,12]) **False** >>> checksum([3,3])  **True** |

Q2. [**16 pts**.] Write a function, checksum(values), that **returns** True if the final integer in the values list equals the sum total of all previous integers in values; otherwise it **returns** False. Assume that values contains at least two integers.

def checksum(list): +4 header +2 docString  
 """Return True if last element equals sum of others"""

sum = 0 +1 variable to store sum  
for idx in range(len(list)-1): +2 for loop +1 one less +1 :  
 sum += list[idx] +2 increase sum with value

return sum == list[-1] +1 return +2 accurate

|  |
| --- |
| Q2 Sample Output |
| >>> countGaps([10,11,12])  **0** >>> countGaps([1,4,6,8])  **4**  >>> countGaps([1,2,5,7,10])  **5** |

Q3. [**16 pts.**] Write a function countGaps(values) that takes a values list containing at least two integers in increasing order. This function **returns** a count of the missing integers in a sequence starting with the first integer in values and ending with the last integer in values.

def countGaps(values): **+4 function header**""" **+2 documentation**Count how many missing numbers exist in sequence from 1st element to last  
""" **+2 variable to keep track**   
count = 0 **+2 variable for loop   
 +1 index iteration**  
for idx in range(len(values)-1): **+1 len-1**  
 count += values[idx+1] - values[idx] - 1 **+2 computation**  
return (count) **+2 return proper count**

Q4. [**18 pts.]** In this question you are to demonstrate that you can hand-execute Python code. Below start by executing main() and then report on the value of list printed at three places.

|  |
| --- |
| **def** trial(list):  **1. What is the first value printed?**  **[4, 4, 6, 1]**  **2. What is the second value printed?**  **[4, 6, 1]**  **3. What is the third value printed?**  **[2, 5]**  idx = 1  **while** idx **in** list:  **del** list[idx]  idx += 1  **def** main():  list = [4, 2, 6, 1]  **for** idx in range(1,2):  list[idx] += list[idx] \* idx  **print**(list)  trial(list)  **print**(list)  list = [2, 1, 5, 4]  trial(list)  **print**(list) |

Q4. [**18 pts.]** In this question you are to demonstrate that you can hand-execute Python code. Below start by executing main() and then report on the value of list **printed** at three places.

|  |
| --- |
| **def** trial(list):  **1. What is the first value printed?**  **[1, 8, 8, 9]**  **2. What is the second value printed?**  **[1, 8, 9]**  **3. What is the third value printed?**  **[7, 2]**  idx = 1  **while** idx **in** list:  **del** list[idx]  idx += 1    **def** main():  list = [1, 7, 8, 9]  **for** idx in range(1,2):  list[idx] = list[idx] + idx  **print**(list)  trial(list)  **print**(list)  list = [7, 1, 2, 4]  trial(list)  **print**(list) |

Q5. [**18 pts.**] You are provided with the following function (I only show you the header and its documentation). **NOTE: You do not have to write the info function! It is provided to you.**

**def** info(element):  
 """  
 For a string representing an atomic symbol, return a list of   
 three string literals representing (in this order) that   
 element’s atomic number, full name, and atomic mass.  
 """

|  |
| --- |
| Q5 Sample Output |
| >>> info('C') **['6', 'Carbon', '12.011']**  >>> info('H') **['1', 'Hydrogen', '1.008']**  >>> mass('CHHHH')  **16.043** >>> mass('CC') **24.022** |

Using this info function, write a function mass(molecule) that takes a string description of the atoms in a molecule and **returns** its total mass when adding together the masses of the individual atoms. Note that the mass of *methane* (CHHHH) is 16.043 because 16.043 = 12.011 + 4\*1.008.

**Write the mass(molecule) function below:**

def mass(molecule): **+4 function header**""" **+2 documentation**Sum up the total mass of all atoms in molecule  
""" **+1 variable to keep track**   
mass = 0 **+2 variable for loop over s**   
**for** e in molecule: **+2 for retrieving val[2]** val = info(e) **+2 for calling info** mass += **int**(val[2]) **+2 for summing** **+1 for convert to int**  
**return** (mass) **+2 return proper count**