Student ID: 730091091
ECM1421DA Exam

1.
   a.

```python
def q1a(list1, list2):
    '''Returns a list that contains elements of two (non-empty) lists of integers,
with duplicates removed, sorted in ascending order.'''
    # Combining the two lists
    combined_list = list1 + list2
    # Removing the duplicates
    unique_list = list(dict.fromkeys(combined_list))
    # Sorting the list
    sorted_list = unique_list.sort()
    return sorted_list
```

   b.

```python
def q1b(numbers):
    '''Returns the median of integers in a non-empty list'''
    # Sorting the list
    sorted = numbers.sort()
    # Finding the median position
    pos = (len(sorted)+1)/2
    # Returning the median
    if int(pos) == pos:
        return sorted[pos-1]
    else:
        return (sorted[pos-1.5]+sorted[pos-0.5]) / 2
```

   c. Python is applying BIDMAS rules where it is first diving my_third by 3 and then adding my_first and my_second. This can be corrected by enclosing my_first + my_second + my_third in brackets.
   d.

```python
print(counties[8:13])
```

   e. Since the length of regions is 4, the while loop is going to iterate from i=0 to i=4. When it reaches i=4, it is going to try to print the value in regions with index 4, but there are only 4 items, with the 1st item having an index of 1 and the 4th item having an index of 3. This will cause a list index out of range error. This can be fixed by changing the <= to a <, as this will only iterated up to i=3.
   f. At the end of each while loop iteration, i is not being changed and remains at 0, unlike in the previous question where there is an i+= 1 within the while loop. This means that the while condition will always be true and program will continuously print out treeslist[0], or in other words, the

program will constantly print out “oak”. This can be fixed by adding i+=1 inside of the while loop after the print statement.

2.

a.

i.

```python
def calc_dist(i_vel, acc, time_s):
    '''Returns the distance travelled by a body, given its initial velocity and 
rate of acceleration'''
    return (i_vel * time_s) + time_s**2 * acc/2

i_vel = 10 #initial velocity
acc = 2 #acceleration
time_s = 10 #time in seconds

print(calc_dist(i_vel, acc, time_s))
```

ii. Docstrings play an important role in conveying functions' purpose, inputs, outputs and behaviour. They enhance the code’s readability and assist other developers in understanding the code. By describing a function’s purpose, expected inputs, and outputs, docstrings make your code easier for you and others to understand. Well-structured docstrings allow developers to quickly grasp the essence of a function without delving into the implementation details. Therefore, including docstrings is essential for maintaining clear, well-documented code.

iii.

```python
assert time_s < 0, 'Time less than zero.'
```

iv. One reason why it may be beneficial to convert lines of Python code to functions is code reuse. Functions allow you to define a piece of functionality once and reuse it multiple times throughout your program. By encapsulating specific tasks within functions, you avoid duplicating code and keep your program concise and readable. Another reason is readability and conciseness. Well-named functions make your code more understandable. Instead of having lengthy code blocks, you can call a function with a descriptive name and a docstring, making it clear what that part of the code does. This improves readability and maintainability, especially when collaborating with other developers.

b.

i.

```
try:
    my_div(a, b)
except:
    print('Division Error')
```

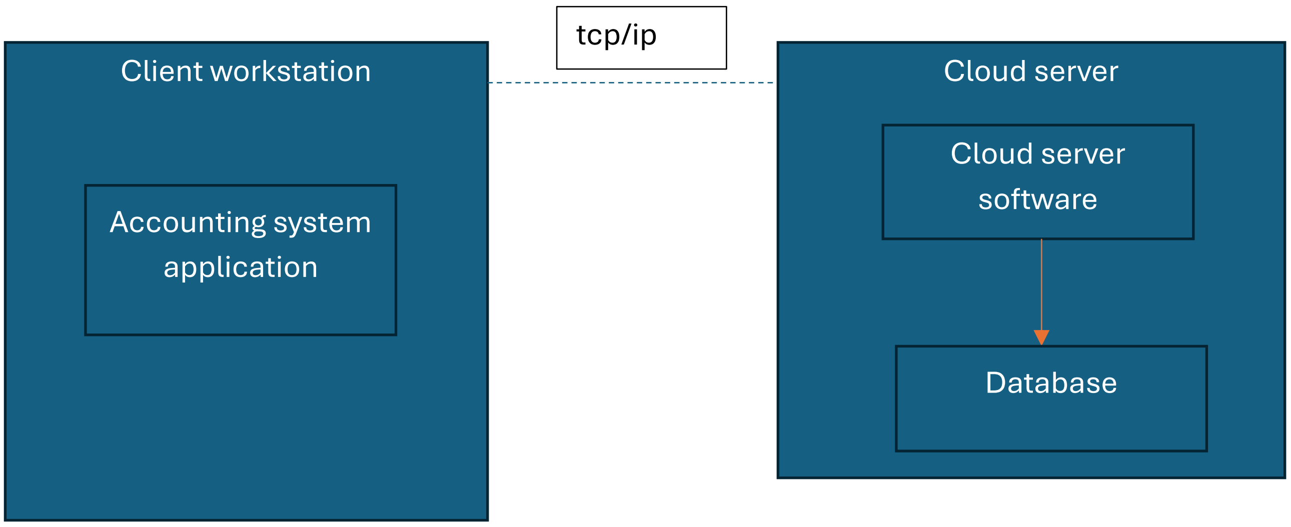
ii. One reason why it is important to handle program exceptions correctly is program reliability. By handling exceptions properly, you can prevent your program from crashing or producing incorrect results due to unexpected errors or input. Another reason is simplified error handling. Exception handling allows you to separate error-handling code from the main program logic, making it easier to read and maintain your code.

3.

a.

i. A 'thick-client' is a software application or system architecture where most of the processing and functionality resides on the client side (i.e., the user's device). The user's workstation must have a certain amount of memory and disk space available for the program to run.

ii.



iii. This implementation may require a complex installation and requires the use of objects native to the client operating system. Additionally, the software provider will need to provide support and upgrades. There may also be security and management implications involved.

b.

i. In a linear development model, analysis is started and completed in one go at the start of the project. Once analysis is completed, it is not repeated until a new system feature is identified. In this ay, the omission of the user stories would not be reviewed or identified until the development of the current requirements is complete. This way would allow the team to deliver a working system for warehouse workers and the accounts team quickly, but the purchasing section would need to be analysed, designed, coded and tested once these sections had been completed. In an iterative development model, the requirements are constantly evaluated, reviewed and modified as appropriate. Projects using this method often have brief requirements at the start which evolve over time, with the intention that these evolve into the required system. The omission of these user stories would be picked up quickly and analysis of this section can take place whilst the other stories are being worked on. Once some requirements have been set, no matter how brief, development can start on the purchasing section. This may cause a delay in the full delivery of the system, but once the system is delivered, all requirements should have been met.

ii. At the start of a project, it is often the case that users are not clear of the full, detailed requirements of the system. It can be difficult to visualise what the system may look like and how it will feel to use. Therefore, many of the required features may not have been specified in the intial requirements. This means the initial deployment of the software will often lack many of the features that are wanted. As per Boehm’s Spiral Model, the development process occurs multiple times in a spiral fashion, with multiple prototypes being delivered before the final mature software with all of the features is deployed.