1. This code is an example of the Adapter Design Pattern using abstraction. The legacy system provides song data in a list format, while the music app expects it in a dictionary format. The ListToDictAdapter acts as a bridge by converting the list into a dictionary before passing it to the app. This allows the old system and new application to work together without changing their original code. Implement a **class-based** adapter. **[6]**

```
from abc import ABC, abstractmethod

# ----- Target Interface (Abstract Class) -----
class MusicData(ABC):
    @abstractmethod
    def get_song_data(self):
        """Return song data in dictionary format"""
        pass

# ----- Adaptee/external service  -----
class LegacyMusicList:
    def get_song_data_list(self):
        # Format: [title, artist/band, duration]
        return ["Dhushor Somoy", "Artcell", 203]

# ----- Adapter (Implements Target Interface) -----
class ListToDictAdapter( MusicData, LegacyMusicList ):
    def __init__(self, legacy_list):
        # no need to write a constructor


    def get_song_data(self):
        song_data_list = self.get_song_data_list()
        return {"title":song_data_list[0], "artist":song_data_list[1], "duration":song_data_list[2]}




# ----- Tester Code -----
legacy_music = LegacyMusicList()
adapter = ListToDictAdapter(          Nothing to write here          )
song = adapter.get_song_data()
print(f"Now Playing: '{song['title']}' by {song['artist']} [{song['duration']}s]")
```

2. Now, write the singleton design pattern code in the **ListToDictAdapter** class. **Do not rewrite** the init() and get_song_data() methods. **[4]**

```
class ListToDictAdapter:
    __instance = None

    def __new__(cls):
      if cls.__instance is None:
        print("Creating the instance")
        cls.__instance = super().__new__(cls)
      return cls.__instance

obj1 = ListToDictAdapter()
obj2 = ListToDictAdapter()
print(obj1 is obj2)
```

3. Write a single note to define Load testing, Stress testing, Alpha testing, Beta testing and Regression testing. **[5]**

a. **Load testing:**

b. **Stress testing:**

c. **Alpha testing:**

d. Beta testing:

e. Regression testing: