1. (a) Consider the following Java method which takes a student's class performance, mid-term examination and final examination marks as parameter. The method then calculates and returns the student's cumulative letter grade.

String evaluateGrade(double ct, double mid, double final) returns totalGrade;

Here are some requirements regarding how this method should operate- [3 × 4 = 12]

- Marks of each section (ct, mid, final) must be non negative. If any input parameter is less than zero the method throws *InvalidSectionScore* exception.
- The sum of all three sections cannot be greater than 90, in which cases the method throws *ScoreOverflow* exception
- Some test cases are listed on the following table -

Case No.	Input	Expected Output
1	(20, 30, 40)	A
2	(10, 26, 36)	B+
3	(-30, 26, 35)	Exception[InvalidSectionScore]
4	(20, 36, 39)	Exception[ScoreOverflow]
5	(0, 15, 0)	F

Now devise executable test cases for **any four out of the above five**(including both error cases) specifications for this method in the *JUnit* notation.

Case no. 01:

```
@Test

Public void testEvaluateGrade_Normal(){

    Student stu = new Student("Mr. X","011011506");

    double ct= 20, mid= 30, final= 40;

    String grade = stu.evaluateGrade(ct, mid, final);

    String expectedGrade = "A";

    AssertEquals(grade, expectedGrade);

}
```

Case no. 02:

```
@Test
Public void testEvaluateGrade_Normal(){
    Student student = new Student("Mr. Y","011012506");
    double ct= 10, mid= 26, final= 36;
    String gpa = student.evaluateGrade(ct, mid, final);
    String expectedGrade = "B+";
    AssertEquals(gpa, expectedGrade);
}
```

Case no. 03:

```
@Test
Public void testEvaluateGrade_NegativeInvalid(){
    Student Student = new Student("Mr. Loser","011000000");
    double ct= -30, mid= 26, final=35;
    Throwable exception = AssertThrows(InvalidSectionScore.class, () ->
{Student.evaluateGrade(ct, mid, final);});
    AssertEquals = ("The value of ct, mid and final can't be negative",
exception.getMessage());

}
```

Case no. 04:

```
@Test

Public void testEvaluateGrade_Overflow(){

    Student Student = new Student("Mr. Advance","011100100");

    double ct= 20, mid= 36, final=39;

    Throwable exception = AssertThrows(ScoreOverflow.class, () ->
{Student.evaluateGrade(ct, mid, final);});

    AssertEquals = ("The total sum of ct, mid and final can't be larger
than 90", exception.getMessage());


}
```

Case no. 05:

```
@Test

Public void testEvaluateGrade_FailCase(){

    Student s = new Student("Mr. Z","011012569");

    double ct= 0;

    double mid= 15;

    double final= 0;

    String finalGrade = s.evaluateGrade(ct, mid, final);

    String expectedGrade = "F";

    AssertEquals(finalGrade, expectedGrade);

}
```

(b) Derive the output when the following test suite is executed. [4]

```
@BeforeAll
public void init(){
    Sytem.out.println("Starting method Testing");
}
@AfterEach
public void cleanup(){
    System.Out.println("Finished Testing");
}
@Test
public void testMethod_normal() {
// Setup
    Student s = new Student();
    s.setName("MrSQA");
    s.setID("1100927001290");
    s.setGPA(3.55)
// Test Steps
    try{                MnSQA
        assertThat(s.getName(), both(containsString("Mr")).and(containsString("SQA"));
        System.out.Println("Name Test Successful");
         assertAll("student",
            () -> assertEquals("1100927001290", s.getID()),
            () -> assertEquals("3.75", s.getGPA()));
         System.out.Println("Id and cgpa test successful");
    }catch(Exception e){
        fail("failed after the previous step");
    }
}
```

@Outputs

1. Starting method Testing.

2. Name Test Successful.

3. failed after the previous step.

4. Finished Testing.