[redacted]

Question - 1
git describe

`git describe <commit-id>` describes the commit against

- ○ the HEAD.
- ○ the nearest branch.
- ○ the nearest tag.
- ○ the latest commit of the main branch.

Question - 2
git squash

Which command(s) will squash the last 3 commits into one commit? Pick all that apply.

- ○ git rebase -i HEAD~3 ; In the screen that opens up, change “pick” to “squash” for the 2 commits before the latest commit.
- ○ git rebase -i HEAD~3 ; In the screen that opens up, change “pick” to “fixup” for the 2 commits before the latest commit.
- ○ git reset --soft HEAD~3 && git commit
- ○ git squash 3

Question - 3
git status

Based on the output of the ‘git status’ command that is shown, which of the following statements is true?

```
git status
On branch newbranch
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:  README.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:  README.md
```

- There are 2 files with the name README.md.
- `git reset --soft HEAD~` was executed and the last commit added the REAMDE.md file.
- `git reset --mixed HEAD~` was executed and the last commit added the REAMDE.md file.
- `git reset --hard HEAD~` was executed and the last commit added the REAMDE.md file.
- `git add README.md` was executed and then another change was made to the README.md file.

Question - 4
Clone a Git Repo

Which command(s) will clone a Git repository with submodules? Select all that apply.

- git clone --recurse-submodules https://github.com/cameronmcnz/surface.git
- git clone --recursive https://github.com/cameronmcnz/surface.git
- git clone https://github.com/cameronmcnz/surface.git ; git submodule init; git submodule update --recursive
- git clone https://github.com/cameronmcnz/surface.git; git fetch --all --recursive.
- git clone https://github.com/cameronmcnz/surface.git ;git submodule update --init --recursive

Question - 5
How Many Branches?

You have just cloned a remote repository to your local disk. The remote repository has 5 branches. How many branches does your local repository have?

- 5
- 6
- 10
- 15

Question - 6
Conditional Coverage

Unit tests for a WeatherForecast class are being enhanced. This includes a conditional operation depending on the temperature range. A test case is needed that improves code coverage and ensures that branches in the forecastDescription method are tested.

Consider the following Swift snippet:

```
class WeatherForecast {
    func forecastDescription(temperature: Double) -> String {
        if temperature < 0 {
            return "Freezing conditions expected."
        } else if temperature >= 0 && temperature < 15 {
            return "Chilly weather ahead."
        } else if temperature >= 15 && temperature < 25 {
            return "Mild temperatures for the day."
        } else {
            return "Heatwave conditions imminent."
        }
    }
}
```

The current test suite has one test case for the "Mild temperatures for the day." scenario. Which of the following unit test snippets correctly increases the code coverage by testing the untested branches?

○

```
func testForecastForFreezing() {
    let forecast = WeatherForecast()
    XCTAssertEqual(forecast.forecastDescription(temperature: -5), "Freezing
conditions expected.")
}

func testForecastForHeatwave() {
    let forecast = WeatherForecast()
    XCTAssertEqual(forecast.forecastDescription(temperature: 30), "Heatwave
conditions imminent.")
}
```

○

```
func testForecastRange() {
    let forecast = WeatherForecast()
    let description = forecast.forecastDescription(temperature: -5)
    let containsFreezing = description.contains("Freezing")
    let containsHeatwave = description.contains("Heatwave")

    XCTAssertTrue(containsFreezing || containsHeatwave)
}
```

○

```
func testForecastBelowFreezing() {
    let forecast = WeatherForecast()
    let description = forecast.forecastDescription(temperature: -1)
    XCTAssertEqual(description, "Freezing conditions expected.")
}
func testForecastAboveHeatwave() {
    let forecast = WeatherForecast()
    let description = forecast.forecastDescription(temperature: 25)
    XCTAssertEqual(description, "Heatwave conditions imminent.")
}
```

○

```
func testForecastAtThresholds() {
    let forecast = WeatherForecast()
    let descriptions = [
        forecast.forecastDescription(temperature: -0.1),
        forecast.forecastDescription(temperature: 14.9),
        forecast.forecastDescription(temperature: 24.9)
    ]
    XCTAssertEqual(descriptions, ["Freezing conditions expected.", "Chilly weather
ahead.", "Mild temperatures for the day."])
}
```

Question - 7
Unit Testing

Select one or more advantages of writing *unit tests*:

- Simplifies debugging by uncovering bugs early in development
- Speeds development by simplifying integration
- Improves design as part of test-driven development
- Detects new bugs during regression testing

Question - 8
JUnit Multiple When Usage

Consider the following code block

```
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.util.ArrayList;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

public class JUnitWhenMultipleUsage {

    private List<String> nameList;

    @BeforeEach
    public void init() {
        nameList = mock(ArrayList.class);
    }

    @Test
    public void testWithMockedArrayList() {
        when(nameList.size()).thenReturn(10).thenReturn(20).thenReturn(30).thenReturn(40).thenReturn(41);
        assertEquals(X, nameList.size());
        assertEquals(Y, nameList.size());
        assertEquals(Z, nameList.size());

        verify(nameList, T).size();

    }
}
```

Which of the following *X, Y, Z,* and *T* values, when applied, allow the *testWithMockedArrayList* method to pass the test?

- X = 41
  Y = 41
  Z= 41
  T = atLeastOnce()

-

X = 41
Y = 41
Z= 41
T = times(3)

- ◯ X = 10
  Y = 20
  Z= 30
  T = times(3)

- ◯ X = 41
  Y = 41
  Z= 41
  T = atLeast(3)

Question - 9
Mockito Inject Mock

Which of the following annotations can be used to inject mock attributes into the test object automatically?

- ◯ @Inject
- ◯ @MockInjects
- ◯ @InjectMocks
- ◯ @Mocks

Question - 10
Mockito Solution

DoctorServiceImpl.java

```
@Service
public class DoctorServiceImpl{

    private DoctorRepository doctorRepository;
    private DoctorMapper doctorMapper;

    public DoctorServiceImpl(DoctorRepository doctorRepository,
                             DoctorMapper doctorMapper) {
        this.doctorRepository = doctorRepository;
        this.doctorMapper = doctorMapper;
    }

    public DoctorDto save(String name, String speciality, Double hourlyRate) {
        Doctor doctor = new Doctor();
        doctor.setName(name);
        doctor.setSpeciality(speciality);
        doctor.setHourlyRate(hourlyRate);

        doctorRepository.save(doctor);
        DoctorDto dto = doctorMapper.entityToDto(doctor);
        return dto;
    }
}
```

DoctorServiceTest.java

```
//annotation here
public class DoctorServiceTest {

    @InjectMocks
    private DoctorServiceImpl doctorService;

    @Mock
    private DoctorRepository doctorRepository;

    @Mock
    private DoctorMapper doctorMapper;

    @Test
    void givenDoctor_whenSave_thenCheckIfDoctorSaved(){
        //given
        String name = "doctor bambam";
        String speciality = "bambam";
        Double hourlyRate = Double.MIN_VALUE;
        //when
        doctorService.save(name,speciality,hourlyRate);
        //
        verify(doctorRepository,times(1)).save(any(Doctor.class));
    }
}
```

To ensure DoctorServiceTest successfully runs and the test passes, what annotation should be used?

- @Test
- @ExtendWith(MockitoExtension.class)
- @RunWith(JUnitPlatform.class)
- None of the above

Question - 11
JUnit Test Order

```
//annotation 1 here
public class HackerRankTest {
    private static StringBuilder test = new StringBuilder("");
    @Test
    //annotation 2
    public void hack() {
        test.append("Hack");
    }
    @Test
    //annotation 3
    public void rank() {
        test.append("Rank");
    }
    @Test
    //annotation 4
    public void er() {
        test.append("er");
    }
    @AfterAll
    public static void assertOutput() {
        assertEquals(test.toString(), "HackerRank");
    }
}
```

Which of the following annotation usages would pass the test with success?

- annotation 1 -> @TestMethodOrder(MethodOrderer.Random.class)
  annotation 2 -> @Order(1)
  annotation 3 -> @Order(2)
  annotation 4 -> @Order(3)

- annotation 1 -> @TestMethodOrder(MethodOrderer.OrderAnnotation.class)
  annotation 2 -> @Order(1)
  annotation 3 -> @Order(2)
  annotation 4 -> @Order(3)

- annotation 1 -> @TestMethodOrder(MethodOrderer.OrderAnnotation.class)
  annotation 2 -> @Order(1)
  annotation 3 -> @Order(3)
  annotation 4 -> @Order(2)

- None of the above

Question - 12
Spring Integration Test MockMvc

```
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import static org.hamcrest.Matchers.containsStringIgnoringCase;
import static org.junit.Assert.assertEquals;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.result.MockMvcResultHandlers.print;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.MvcResult;

@SpringBootTest
@AutoConfigureMockMvc
public class HackerRankIntegrationTest {

    @Autowired
    private MockMvc mockMvc;

    @Test
    public void shouldReturnDefaultMessage() throws Exception {
        MvcResult mvcResult = this.mockMvc.perform(get("/user/listUsers"))
                .andDo(print())
                .andExpect(status().isOk())
                .andExpect(content().string(containsStringIgnoringCase("Hacker Rank Rock")))
                .andReturn();

        assertEquals("application/json",mvcResult.getResponse().getContentType());
    }
}
```

Assume all the following responses are from "/hacker/rank". Which of them may pass the test?

- {"data":{ "key":"Hacker", value:" Rank rocks"}}

- ◯ {"data":{ "key":"Hacker Rank", value:"rocks Hacker rocks Rank"}}
- ◯ {"data":{ "key":"Hacker Rank", value:"rocks"}}
- ◯ {"data":{ "key":"hacker Rank Rocks", value:" Rank rocks"}}