1. Does the following code violate OCP? Explain.

```java
abstract class Shape{
    public double computeArea() {
        if(this instanceof Circle) {
            Circle c = (Circle)this;
            return c.radius * c.radius * Math.PI;
        }
        else if(this instanceof Square) {
            Square s = (Square)this;
            return s.side * s.side;
        }
        else
            return 0;
    }
}
```

```java
class Circle extends Shape{
    double radius;
}

class Square extends Shape{
    double side;
}
```

2. Does the following code violate LSP? Explain.

```java
class Fraction{
    int numerator;
    int denominator;

    public Fraction(int numerator, int denominator) {
        this.numerator = numerator;
        this.denominator = denominator;
    }

    public void setNumerator(int numerator) {
        this.numerator = numerator;
    }

    public void setDenominator(int denominator) {
        this.denominator = denominator;
    }

    @Override
    public String toString() {
        return numerator + "/" + denominator;
    }
}
```

```java
//represents fractions whose numerator is 1
//e.g. 1/2, 1/3, etc.
class UnitFraction extends Fraction{

    public UnitFraction(int denominator) {
        super(1, denominator);
    }

}
```

3. Does the following code violate DIP? Explain.

```java
interface Strategy {
    String format(int day, int month, int year);
}

class StrategyMDY implements Strategy{
    public String format(int day, int month, int year) {
        return month + "/" + day + "/" + year;
    }
}
```

```java
class Date {
    int day;
    int month;
    int year;

    public Date(int day, int month, int year) {
        this.day = day;
        this.month = month;
        this.year = year;
    }

    @Override
    public String toString() {
        Strategy strategy = new StrategyMDY();
        return strategy.format(day, month, year);
    }
}
```

4. You are required to refactor the code provided in “Q4 code” to address the issue of needless redundancy associated with the **order** method in classes **BurgerStore** and **PizzaStore**.

5. The code provided in “Q5 Code” violates one or more SOLID principles. You are required to refactor it so that it conforms to all five principles.

6. Which design pattern is used for the following code?

```java
class CPU{
    void process(){
    }
}

class Memory{
    void read(){
    }
    void write() {
    }
}
```

```java
class Computer{
    CPU c;
    Memory m;
    void compute() {
        m.read();
        c.process();
        m.write();
    }
}
```

7. Which design pattern is used for the following code?

```java
class Student{
    String name;
    Course course;
    public Student(String name, Course course) {
        this.name = name;
        this.course = course;
        this.course.add(this);
    }
    public void displayExamDate() {
        System.out.println(course.examDate);
    }
}
```

```java
class Course{
    List<Student> students;
    Date examDate;
    public Course() {
        students = new ArrayList<Student>();
    }
    public void addStudent(Student student) {
        students.add(student);
    }
    public void scheduleExam(Date examDate) {
        this.examDate = examDate;
        for(Student s:students)
            s.displayExamDate();
    }
}
```

8. Which design pattern is used for the following code?

```java
class Date {
    int day;
    int month;
    int year;
    DateFormatter formatter;

    public Date(int day, int month, int year) {
        this.day = day;
        this.month = month;
        this.year = year;
    }

    public void setFormatter(DateFormatter formatter){
        this.formatter = formatter;
    }

    @Override
    public String toString() {
        return formatter.format(day, month, year);
    }
}
```

```java
interface DateFormatter{
    String format(int day, int month, int year);
}

class MDYDateFormatter implements DateFormatter{
    public String format(int day, int month, int year) {
        return month + "/" + day + "/" + year;
    }
}
```

9. Modify the code of “Programming Exercise 2” of Problem Set 1 using the appropriate design pattern(s) as follows:
   a. Define class **Administration** such that:
      i. It should not be possible to have more than one instance of this class
      ii. It has three fields whose types ensure that no duplicates are allowed and that the elements are stored in the same order they are added
         - **professors** (a collection of all the instantiated professors)
         - **students** (a collection of all the instantiated students)
         - **courses** (a collection of all the instantiated courses)
      iii. It has three methods of type **void**:
         - **addProfessor** that takes an object of type **Professor** as an argument and adds it to the collection of professors
         - **addStudent** that takes an object of type **Student** as an argument and adds it to the collection of students
         - **addCourse** that takes an object of type **Course** as an argument and adds it to the collection of courses
      iv. Every time an object of type **Professor**, **Student**, or **Course** is instantiated in the code, it should be added to the appropriate collection. Note that you might need to modify other classes to achieve that.
   b. The problem with method **isEligible** in class **Course** is that it is does not account for courses other than CSCB07. Even if it is to be used solely for CSCB07, modifying the eligibility conditions later on would require modifying class **Course**. As such, you are required to make the necessary changes so that any future modifications to the eligibility criteria could be done without modifying class **Course**.

10. The code provided in “Q10 Code” includes three classes (DataAnalyis, BubbleSort, MergeSort) and one interface (SortingService). The DataAnalysis class uses a SortingService reference to help compute the median. Using the appropriate design pattern, you are required to add the necessary code so that the sorting services of BubbleSort and MergeSort could be used. For example, a user would be able to utilize your code to write a driver program that instantiates DataAnalysis and computes the median using bubble sort or merge sort. Take note of the following:
    a. You should not modify DataAnalyis, BubbleSort, MergeSort, or SortingService in any way.
    b. Your implementation should be based on one of the design patterns discussed in the course.
    c. You can write as many classes/interfaces as you need.
    d. Sorting should be done using BubbleSort or MergeSort only (i.e. you can't add sorting code somewhere else).

11. You are required to develop a program that mimics a shell. The program should continuously read commands entered by the user through the keyboard and execute them. The commands have the form:

    ```
    command arg1 arg2 …
    ```

    We will consider simplified versions of three well-known commands:

    - `cat`: It takes one path argument. If the path represents a file, the program should display the contents of the file. Otherwise, if the path (say *some_path*) represents a directory, the following message should be displayed: “some_path is a directory”

- `ls`: It takes zero or one path argument. As such, three cases should be considered:
    - If no argument is provided, it should display the contents of the default directory whose path is "."
    - If one argument representing a file is provided, the name of the file should be displayed
    - If one argument representing a directory is provided, the names of the files and subdirectories contained in it should be displayed (one name per line).
- `exit`: It takes no arguments and is used to terminate the program. You can use **System.exit(0);** to do that.

Additional requirements:

- Each of the commands we are considering has a minimum and a maximum number of arguments (e.g. `ls` can have between 0 and 1 arguments). If too few arguments are provided to a command, the message "Too few arguments" should be displayed and the user should be notified without exiting the program. The same goes for the following cases:
    - Too many arguments are provided
    - An incorrect path is provided
    - An unsupported command is provided
- The code must be developed in a way that allows new commands to be added with minimal modifications to the existing classes. **Your solution should be based on the appropriate design pattern.**

Sample input/output:

```
ls /users/rawad/my_directory
.DS_Store
f1.txt
f3.txt
f2.txt
cat
Too few arguments provided
cat /users/rawad/my_directory/f1.txt
Line 1
Line 2
exit
```

12. The Dependency Inversion Principle promotes code reusability.

True/False: _______

13. Inheritance promotes code reusability.

True/False: _______

14. Which design pattern is the most suitable for a sports app that allows users to receive live updates of a match?
    a. Singleton
    b. Strategy
    c. Observer
    d. Façade

15. Match each of the following with the corresponding model (Waterfall or Spiral):
    a. Working software is produced late during the life cycle
    b. Used for high-risk projects
    c. Estimating the time to complete the project is more difficult
    d. More suitable for projects where requirements change

16. Match each of the following with the corresponding SDLC phase (Planning, Analysis, Design, Implementation, Maintenance):
    a. Describing the structure of the user interface
    b. Releasing a patch
    c. Drawing a class diagram
    d. Writing pseudo-code

17. For each of the following, indicate whether it is promoted by the Agile approach:
    a. Delivery of working software at short duration intervals
    b. Customer collaboration
    c. Responding to change

18. Which of the following describes Test-Driven Development?
    a. Implement the design, then develop a test suite that achieves high coverage
    b. Develop a test suite, then implement the design
    c. Write tests incrementally and modify the code when needed to make the tests pass
    d. None of the choices is correct

19. In Scrum, which of the following is not the responsibility of the Scrum Master?
    a. Coordinating the meetings
    b. Addressing obstacles
    c. Coaching
    d. Assigning tasks to the team members

20. In Scrum, Planning Poker is used to estimate the difficulty associated with user stories.
    a. True
    b. False

21. In Scrum, the client is responsible for communicating user stories directly to the team.
    a. True
    b. False

22. In Scrum, which of the following is the responsibility of the Product Owner?
    a. Managing the product backlog
    b. Communicating with the clients

    c. Communicating with the development team
    d. All of the choices are correct

23. In Scrum, standup meetings only occur when something urgent happens that might prevent the team from meeting the sprint goal.
    a. True
    b. False

24. What is missing from the following user story? "I should be able to change my password"
    a. Actor/object
    b. Action
    c. Result
    d. Actor/object and result

25. During a planning poker session, the team members (Alex, Bob, Carol, and Dan) select the following values. Briefly explain what happens next.
    Alex: 6
    Bob: 2
    Carol: 7
    Dan: 5

26. Test-Driven Development easily accommodates late changes in requirements.
    True/False: _______

27. Feature-Driven Development is more suitable than Waterfall for small projects.
    True/False: _______

28. Match each of the following with the corresponding property (Automating the build process, Binding activities at runtime, UI Testing, Representing the metadata of the application).

Intent:	____________
Manifest file:	____________
Gradle script:	____________
Espresso:	____________

29. Which of the following is NOT a lifecycle callback method of an activity in Android?
    a. onCreate
    b. onResume
    c. onDestroy
    d. onExit

30. Briefly explain the main advantage of using the **strings.xml** file in Android.

31. Local unit tests are more costly than UI tests.
    True/False: _______

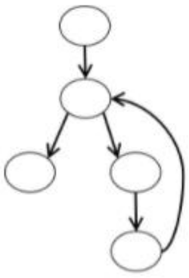
32. Will the following test pass? Explain.

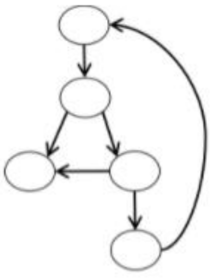
```
public class Presenter{
    private Model model;
    private View view;

    public Presenter(Model model, View view){
        this.model = model;
        this.view = view;
    }

    public void registerStudent() {
        String studentID = view.getStudentID();
        String courseCode = view.getCourseCode();
        if(!model.studentExists(studentID))
            view.displayMessage("student not found");
        else if(!model.courseExists(courseCode))
            view.displayMessage("course not found");
        else if(model.alreadyRegistered(studentID, courseCode))
            view.displayMessage("student already registered");
        else{
            model.register(studentID, courseCode);
            view.displayMessage("student registered successfully");
        }
    }
}
```

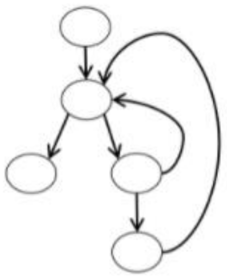
```
@RunWith(MockitoJUnitRunner.class)
public class ExampleUnitTest {
    @Mock
    View view;

    @Mock
    Model model;

    @Test
    public void testPresenter(){
        when(view.getStudentID()).thenReturn("100");
        when(view.getCourseCode()).thenReturn("CSCB07");
        when(model.studentExists("100")).thenReturn(true);
        when(model.courseExists("CSCB07")).thenReturn(true);
        Presenter presenter = new Presenter(model, view);
        presenter.registerStudent();
        presenter.registerStudent();
        verify(view, times(2)).displayMessage("student registered successfully");
    }
}
```

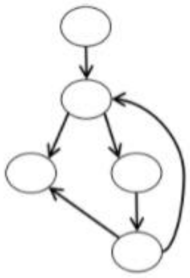
33. Which control flow graph represents the following code?

```
public static int numZero (int [ ] arr){
    int count = 0;
    for (int i = 0; i < arr.length; i++){
        if (arr [ i ] == 0){
            count++;
        }
    }
    return count;
}
```


A


B


C


D