1. .The following scenarios depict different levels of cohesion in a software design
   a) Consider the following pseudocode for an object class which is used for generating monthly reports for a system

```
class Report{
void connectDB();
void generateMonthlySalesReport();
void getCurrentWeather();
void restartServer();
}
```

   - What type of *cohesion* do you think it depicts and why?

   Cohesion: Coincidental cohesion

   Methods are unrelated and serve different purposes

   - What kinds of problems can arise with the type of *cohesion* you identified?

   Hard to maintain, low reuse, changes affect unrelated functionality

   - How can the class be improved upon to reduce/avoid the problems you claimed?

   Split into separate, focused classes

   a) What about the following object class?

```
class Cake{
void addEggs(int num);
void addSugar(double amnt);
void addFat(double amnt);
void beatIt(double woohoo);
}
```

   Cohesion: Functional cohesion
   All methods contribute to one task

2. What type of coupling is this?

   Class `CourseSection` has ***public*** class variables called `minClassSize` and `maxClassSize`. These are changed from time to time by the university administration. Many methods in classes `Student` and `Registration` access these variables.

   How can we improve this?

   Make variables private and access them through getters/setters or a configuration class

3. The following interface for a `Person` object is said to break the *Interface Segregation* principle of OO design. Explain this problem and how it can be corrected

```
public interface Person{
  public String getName();
  public double getSalary();
  public Date getBirthdate();
  public Schedule getClassSchedule();
  public Schedule getWorkSchedule();
  public void setSalary();
}
```

Split into smaller, role-specific interfaces