CSCI3100 Software Engineering

Assignment 1

Due – 12:59:59pm, 7th Feb, 2021 (Sunday)

Please submit the homework online through Blackboard.

Late submission penalty within 24 hours: 50%; after 24 hours: 100%.

Remember to go through Veriguide for Academic Honesty Declaration.

Missing Veriguide report: 50% mark deduction.

Answer the following problems based on lecture Topics 1 – 3 notes, from which you can consider for solutions. Each question is assigned 25 points as its full marks.

1. Classification of Software Qualities

a) Classify each of the qualities discussed in this lecture as *internal*, *external*, *product* or *process* by filling out the following table with "X" or "√" marks. The classifications are not necessarily mutually exclusive.

b) Explain and justify your answers.

Quality	Internal	External	Product	Process
Correctness				
Reliability				
Robustness				
Performance				
User Friendliness				
Verifiability				
Maintainability				
Repairability				
Evolvability				
Reusability				
Portability				
Understandability				
Interoperability				
Productivity				
Timeliness				

Visibility				

2. Relations among Software Qualities

Consider the following statements. Please judge if it is true or false and give your explanation for each of them. (Note: here A contributes to B means improving A brings large chance of improving B.)

a) Repairability, maintainability, evolvability contribute to each other.
b) Correctness contributes to robustness.
c) Productivity contributes to timeliness.
d) Reliability and robustness contribute to each other.
e) Performance contributes to usability.
f) Understandability contributes to interoperability.
g) Verifiability contributes to correctness.
h) Visibility contributes understandability.
i) Usability and Reusability contribute to each other.
j) Portability contributes to reusability.

(For **Question 3** and **Question 4**) Imagine you are required to implement Warcraft, which has the following properties:

a. The leftmost side of the map is the *headquarter* of the red army, while the rightmost side is the *headquarter* of the blue army. There are 30 *castles* between these two *headquarters*.
b. In every timestamp t, each *headquarter*, if there is no *soldier* in this *headquarter*, will randomly produce a *soldier* among:
    i. *Ninja*: Costs 14 *souls*. *HP*: 80; *Force*: 15; *Defense*: 9. If a *soldier* is attacked by a *Ninja*, then its *Defense* will -2. *Defense* can be negative.
    ii. *Knight*: Costs 17 *souls*. *HP*: 110; *Force*: 10; *Defense*: 15. If a *Knight* kills an *soldier*, then it will get half of the *force* of that *soldier*.
    iii. *Demon*: Costs 20 *souls*. *HP*: 100; *Force*: 14; *Defense*: 7. When a *Demon* moves to a next *castle*, its *HP* will -5 and *Force* will +5. *Demons* will not die because of moving.

    Producing *soldiers* need *souls*. All *castles* (including *headquarters*) will generate 10 *souls* in every timestamp t.
c. In every timestamp t, every *soldier* in the map will move to next *castle* unless there is an enemy in the current *castle*, or there is already a comrade in the next *castle*, or it has already arrived the enemy's *headquarter*.
d. If there are both red and blue *soldiers* in a *castle* (excluding *headquarters*), then there will be a fight. If t mod 2 == 0, then red *soldier* attacks on blue *soldier*. If t mod 2 == 1, then blue *soldier* attacks on red *soldier*. The *HP* of the attackee will be deducted (the *force* of the attacker – the *defense* of the attackee). Also, the attackee will counterattack, that is, the *HP* of the attacker will be deducted (0.5 * the *force* of the attackee – the *defense* of the attacker). If the *HP* becomes non-positive, then this *soldier* dies. If one *soldier* dies and another one does not, then another one gets the *souls* of this *castle*.

e. If there are two enemy *soldiers* in a *headquarter*, then this side loses the game and the opposite wins.

3. Modularity and Inheritance

a) Your friend Tony implemented parts of the classes for you and so did another friend Mary. According to Topic 3 slide 12, we treat a class as a module. Please draw the modularity diagram of Tony's code and Mary's code. Then calculate the total cohesion and coupling of the two implementations. Whose design is better? Why?

Tony's code:

```
class Knight {
    public:
        int HP;
        int force;
        int defense;
        void attack(Ninja ninja);
};

class Ninja {
    public:
        int HP;
        int force;
        int defense;
        void attack(Knight knight);
};

void Knight::attack(Ninja ninja) {
    ninja.HP -= this->force - ninja.defense;
    if (ninja.HP <= 0) {
        this->force += 0.5 * ninja.force;
    }
    this->HP -= 0.5 * ninja.force - this->defense;
}

void Ninja::attack(Knight knight) {
    knight.defense -= 1;
    knight.HP -= this->force - knight.defense;
    this->HP -= 0.5 * knight.force - this->defense;
}
```

Mary's code:

```
class Soldier {
    public:
        int type; // Ninja = 1; Knight = 2;
        int HP;
        int force;
        int defense;
        void attack(Soldier soldier);
};

void Soldier::attack(Soldier soldier) {
    if (this->type == 1) soldier.defense -= 1;
    soldier.HP -= this->force - soldier.defense;
    if (this->type == 2 && soldier.HP <= 0) {
        this->force += 0.5 * soldier.force;
    }
    this->HP -= 0.5 * soldier.force - this->defense;
};
```

b) C++, like other modern programming languages, provides a feature called *inheritance*. Imagine you are further required to implement a new soldier called *Dragon*, please explain what problem you may encounter if you do not inherit certain *soldiers* from a common parent class. According to Topic 3, you have learned principles of designing software. Please explain using inheritance aligns with which principles the best (explain at least two of them)? According to Topic 2, using inheritance contributes to which software qualities most (explain at least two of them with your justifications)?

4. Software Development and Software Development Model

a) These are key elements mentioned above: *headquarter, castle, soldier, souls, HP, force, defense, Ninja, Knight, Demon, timestamp*. There are several **events** you need to implement: soldier moving; soldier attack; get souls from castles; soldier producing; game over. Please design several classes to include the features mentioned above. (Note: you do not have to implement a runnable piece of code; instead, you only need to show which functions/variables should be in which classes.)

b) If you are asked to implement new soldiers or even new mechanism (e.g., weapons). Which software development model you learned in CSCI3100 class will you use? Please explain from at least two features. (Note: typical features can include simplicity, cost, risk control, expertise required, user involvement, flexibility, etc.)