CENG242

Homework #5

(Due Date: 25th May, 2013 Saturday 23:55)

In this homework, you are asked to implement some unit classes of a simple board war game. In this game, there are two enemy sides fighting for their lands (NORTH and SOUTH) in the neutral zone which is our game board. Each side has wizards, men, dwarves, dragons, clerics and one battlemage. The game will end when there is no one left in one of the lands or when the maximum number of turns is reached (you don't need to worry about this part because it will be handled by the GameEngine).

Game rules:

- Each type of creature has different sight and act ranges. Men, dwarves, clerics have an act range of 3x3. Men and clerics have a sight range of 5x5. Dwarves have a sight range of 3x3. Wizards, battlemage, dragons have an act and sight range of 7x7. In these ranges consider the creature as standing in the center. For instance; dwarves can act and see just the neighboring 8 squares (east, southeast, etc).
- Each creature can have different initial toughness level and each hitting or spelling action has different power to damage the opponent creature.
- Wizards have various spelling powers; they can cast a freeze spell on the creatures so that the creature cannot make a move in the next turn (just for one turn) but these bewitched creatures cannot be attacked by dragons, dwarves and men. Wizards can also cast a weaken spell which causes the opponent creature's toughness to decrease according to the wizard's power. Moreover, wizards can teleport anywhere in their sight range (if there isn't any other creature).
- Creatures cannot go to a square where there is another creature or there is a wall.
- All creatures except battlemage, clerics and wizards can benefit from a FOOD. So if you move to a square in which there is a food, automatically increase your toughness by 1.
- The order of the creatures according to their importance is as follows: battlemage > wizard > dragon > cleric > dwarf > human. This importance order is important when attacking or moving to a square; the importance order will get clarified in each type of creature's action.
- Dragons can send fires to the squares in their act area, but their fires are limited. They can also paw to a creature in a neighboring 8 squares.
- Dwarves are able to withstand to heat so they do not get affected by dragon fires.
- Clerics can heal creatures; they help to increase human, dragon toughness by 1, dwarf toughness by 2, wizard toughness by 3, cleric and battlemage toughness by their own action power. When they are healing a dragon, the left fire count is also increased by 1. When they are healing a wizard, the mana count is also increased by 2. (All these operations are handled in ++ operator, so you should increase your toughness by these given values in ++ operator implementation. Additionally, if you are a dragon you should increase your left fire count, and if you are a wizard you should increase the mana count)
- Distance between 2 points should be calculated according to manhattan distance. You need this metric when you are trying to move closer to an enemy.

- You can attack or cast spell on only the enemy creatures and can heal only the creatures on your side.
- Note that you can see everyone in your sight range and can attack to anyone in your act range or move to a square in your act range even there is another creature between you and the target square.
- Throughout the game the following coordinate system will be used. You are expected to obey this coordinate system while giving relativeX, relativeY (position to the current creature which is in 0,0) values. The first coordinate is relativeX, the second one is relativeY. So for instance, if you make a move to the left neighbouring square, set relative to -1, relativeY to 0.

-2, 2	-1, 2	0, 2	1, 2	2, 2
-2, 1	-1, 1	0, 1	1, 1	2, 1
-2, 0	-1, 0	0, 0	1, 0	2, 0
-2, -1	-1, -1	0, -1	1, -1	2, -1
-2, -2	-1, -2	0, -2	1, -2	2, -2

The action flow (move() function) for each type of creature is as follows:

- Battlemage: If your toughness is below 3 (not 3), heal yourself (even clerics cannot heal themselves, battlemage can). Else behave like a wizard if you can cast a spell. If you cannot make a spell, behave like a cleric.
- Wizard: If don't have at least one mana, you cannot cast a spell. If there is an enemy which is battlemage or wizard in your act range, cast a freeze spell to the most important one. If there is an enemy (not battlemage or wizard) in your act range, cast a weaken spell to the most important one. Note that, each spell cost you one mana, if you make a spell, decrease your mana. If there isn't any enemy in your act range, move closer to the most important enemy in your sight. If there is no such square, move randomly.
- Cleric: Heal the most important creature on your side which is in your act area (clerics cannot heal themselves). If you cannot heal a creature, then move closer to the most important creature (on your side) in your sight. If there is no such square, move randomly.
- Dragon: If you have an unbewitched enemy in a neighbouring square, paw the most important one. Else if there is an unbewitched enemy in your act range and if there is fire left, fire the most important creature and decrease your left fire count (except dwarves. Don't send fires to dwarves because they don't get affected.) If you cannot attack a creature, then move closer to the most important enemy in your sight. If you cannot move closer to an enemy then go to a square with food. If there is no such square, move randomly.
- Dwarf: If you have an unbewitched enemy in your act range, attack the most important one. If you cannot make an attack then move closer to the most important enemy in your sight. If you cannot move closer to an enemy then go to a square with food. If there is no such square, move randomly.
- Human: If you have an unbewitched enemy in your act range, attack the most important one. If you cannot make an attack then move closer to the most important enemy in your sight. If you cannot move closer to an enemy then go to a square with food. If there is no such square, move randomly.

In the implementation you should have a base class named Creature. Other classes should be derived from this class and implemented by you:

```
enum Side { NORTH, SOUTH };

enum MoveType { WALK, HUMANATTACK, DWARFATTACK, DRAGONPAW, DRAGONFIRE,
HEAL, FREEZESPELL, WEAKENSPELL, NOACT};

enum ObjectType { BATTLEMAGE, WIZARD, CLERIC, HUMAN, DWARF, DRAGON,
FOOD, WALL};

struct Move

{

    MoveType mType;

    int relativeX;

    int relativeY;

};

struct Object

{

    Side teamSide;

    ObjectType objType;

    int relativeX;

    int relativeY;

    bool bewitched;

};

class Creature

{

private:

    Side side;

    int actionPower;

    int toughness;

    bool bewitched;

public:
```

```
    Creature (Side _side, int _power, int _toughness) : side(_side),
actionPower(_power), toughness(_toughness) { bewitched = false;};

    virtual ~Creature () { };

    Side getSide() const { return side; };

    int getActionPower() const { return actionPower; };

    int getToughness() const { return toughness; };

    bool getBewitched() const { return bewitched };

    void setBewitched(bool _bewitched) { bewitched = _bewitched; };

    void loseToughness(int decreasingToughnessAmount) { toughness -=
decreasingToughnessAmount;}

    virtual void operator++() {} /*this function is called when there
is a healing act on you, so increase your toughness according to the
given values mentioned above in the text (in the derived class
implementation), also increase mana or leftFireCount if applicable*/

    virtual Move move(const std::vector<Object>& list) = 0;

 };


class Human : public Creature

{

public:

    Human (Side _side, int _power, int _toughness) : Creature(_side,
_power, _toughness) { }

    void operator++();

    Move move(const std::vector<Object>& list);

};


class Dwarf : public Creature

{

public:

    Dwarf (Side _side, int _power, int _toughness) : Creature(_side,
_power, _toughness) { }

    void operator++();
```

```
    Move move(const std::vector<Object>& list);
};


class Dragon : public Creature
{
private:
    int leftFireCount;
public:
    Dragon (Side _side, int _power, int _toughness, int
_leftFireCount) : Creature(_side, _power, _toughness),
leftFireCount(_leftFireCount) {}
    int getLeftFireCount() const; // return leftFireCount
    void operator++();
    Move move(const std::vector<Object>& list);
};


class Wizard : public virtual Creature
{
private:
    int mana;
public:
    Wizard (Side _side, int _power, int _toughness, int _mana) :
Creature(_side, _power, _toughness), mana(_mana) { }
    int getMana() const; // return leftFireCount
    void operator++();
    Move move(const std::vector<Object>& list);
};


class Cleric : public virtual Creature
{
```

```
public:

    Cleric (Side _side, int _power, int _toughness) : Creature(_side,
_power, _toughness) { }

    void operator++();

    Move move(const std::vector<Object>& list);

};

class BattleMage : public Wizard, public Cleric

{

public:

    BattleMage (Side _side, int _power, int _toughness, int _mana):
Wizard(_side, _power, _toughness, _mana), Clerics(side, _power,
_toughness), Creature(side, _power, _toughness) { }

    void operator++();

    Move move(const std::vector<Object>& list);

};
```

In this homework you will implement unimplemented functions; also you can add new functions and data members. In each turn move functions of the creatures will be called. Don't forget that you are not going to know the whole board; you will see objects which are just in your sight range. Your act range refers to the squares that you can attack or you can move to.

The game will be simulated by a game engine which you will be given in a few days. You will also be provided a sample *main()* function. In main function, each the *move()* function of each creature will be called in a turn based manner. An object list which includes creatures within the called creature's sight range will be passed to the *move()* function. The *move()* function should return its *MoveType*, and the relative indexes of target square which are bundled in *Move Struct.*

Specifications:

1. All the work should be done individually.
2. In evaluation, black box method will be used. So be careful about the name of functions, data structures etc.
3. You will submit your code through Cow system.
4. Your codes should be written in C++ and you should submit a tar file hw5.tar. The tar file should include hw5.h, hw5.cpp. In hw5.h, you should include class declarations, structures, and enumerated types. In hw5.cpp, you should include the definitions. The main.cpp and a sample game class (runs the game) will be given, but you should NOT submit these files.
5. You should test your codes in inek machines by compiling with g++ before submitting.