```
#include <cstring>
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
//Part A) Define classes in C++
class Character {
string Name;
int Health;
std::vector Weapons; //all are instance variables
char Hasparent;
string Level;
public:
Character (); //constructor
Character (Character&);
virtual string speak() = 0;
}
//Copy constructor
Character::Character(Character& original){
Name = original.Name;
Health = original.Health;
Weapons = original.Weapons;
Hasparent = original.Hasparent;
Level = original.Level;
}
class Level{
string Name;
int levelNumber;
public:
Level (Level&);
}
Level::Level(Level& origin){
Name = origin.Name;
levelNumber = origin.levelNumber;
}
class Weapon{
string Name;
double strength;
public:
Weapon (Weapon&);
}
Weapon::Weapon(Weapon& origin){
Name = origin.Name;
strength = origin.strength;
}
```

```
class Monster : public Character{
double angry;
Weakness weakness;
public:
Monster (Monster&);
Monster::Monster(Monster& origin){
angry = origin.angry;
weakness = origin.weakness;
string Speak();
}
}
//Part B) Change Character to abstract method with speak implementation
Monster::Speak(){
return "Booh!"
}
class Weakness{
string b;
string a;
int c;
public:
Weakness (Weakness&);
Weakness::Weakness(Weakness& origin){
b = origin.b;
a = origin.a;
c = origin.c;
}
}
class Wizard: public Character{
double d = magic_level;
public:
string Speak();
}
Wizard::Speak() {
return "I put a spell on you!";
}
//polymorphic function

void everyoneSpeak(std::vector<Character> characters){
for (std::vector<Character>::iterator it = characters.begin(); it !=
characters.end(); ++it){
 cout << it.Speak();
}
}
```