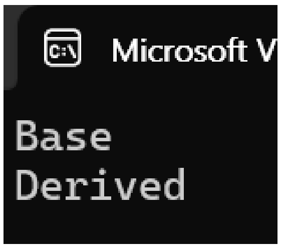
객체지향프로그래밍 LAB #12

<기초문제>

1.

```
#include <iostream>
#include <string>
using namespace std;

class Base {
protected: //Base type
	void print_base() { cout << "Base" << endl; }
};

class Derived : private Base {
public:
	void print_derived() {
		print_base();
		cout << "Derived" << endl;
	}
};

int main() {
	Base base;
	Derived derived;
	derived.print_derived();

	return 0;
}
```



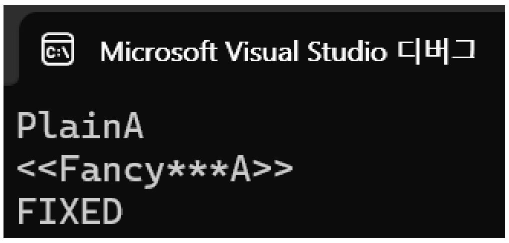
```
Base
Derived
```

2.

```
#include <iostream>
#include <string>
using namespace std;
```

```
class Text {
private:
            string text;
public:
      Text(string _t) : text(_t) {}
      virtual string get() {
               return text;
      }//get()함수 virtual 로 구현
      virtual void append(string _extra) { text += _extra; }
};

class FancyText : public Text {
private:
           // string text;b접근이 안됨, Base Class에서 private
           string left_brac;
           string right_brac;
           string connector;
public:
      // initialization list는 생성자를 호출할 수 있게 해준다.
      FancyText(string _t, string _lb, string _rb, string _con) :
               Text::Text(_t), left_brac(_lb), right_brac(_rb), connector(_con) {}
      string get() override {
               return left_brac + Text::get() + connector + right_brac;
      }
      void append(string e) override { connector += e;}
};

class FixedText : public Text {
public:
      FixedText() : Text::Text("FIXED") {}
      void append(string e) override {}
};

int main() {
            Text t1("Plain");
            t1.append("A");
            cout << t1.get() << endl;

            FancyText t2("Fancy", "<<", ">>", "***");
            t2.append("A");
            cout << t2.get() << endl;

            FixedText t3;
            t3.append("A");
```

```
        cout << t3.get() << endl;
        t1 = t2; // Base <- Derived 가능
        //t2 = t1; // Derived <- Base 불가능

        return 0;
}
```



```
PlainA
<<Fancy***A>>
FIXED
```

3.

(Text.h)

```
#ifndef Text_h
#define Text_h
#include <string>

using namespace std;

class Text {
private:
    string text;
public:
    Text(string _t);
    virtual string get();
    virtual void append(string _extra);
};
#endif
```

(Text.cpp)

```
#include "Text.h"
using namespace std;

Text::Text(string _t) : text(_t) {}
string Text::get() {
    return text;
}
```

```
void Text::append(string _extra) {
	text += _extra;
}
```

(FancyText.h)

```
#ifndef FancyText_h
#define FancyText_h

using namespace std;
#include "Text.h"
#include <string>

class FancyText : public Text {
private:
	string left_brac;
	string right_brac;
	string connector;
public:
	FancyText(string _t, string _lb, string _rb, string _con);
	string get() override;
	void append(string e) override;
};
#endif
```

(FancyText.cpp)

```
#include "FancyText.h"
using namespace std;

FancyText::FancyText(string _t, string _lb, string _rb, string _con) :
				Text::Text(_t), left_brac(_lb), right_brac(_rb), connector(_con) {}
string FancyText::get() {
		return left_brac + Text::get() + connector + right_brac;
}
void FancyText::append(string e) {
	connector += e;
}
```

(FixedText.h)

```
#ifndef FixedText_h
#define FixedText_h
```

```
using namespace std;

#include "Text.h"

class FixedText : public Text {
public:
	FixedText();
	void append(string e) override;
};
#endif
```

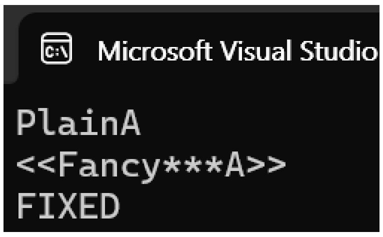
(FixedText.cpp)

```
#include "FixedText.h"
using namespace std;

FixedText::FixedText() : Text::Text("FIXED") {}

void FixedText::append(string e) {}
```

(main.cpp)

```
#include <iostream>
#include <string>
#include "Text.h"
#include "FancyText.h"
#include "FixedText.h"
using namespace std;

int main() {
		Text t1("Plain");
		t1.append("A");
		cout << t1.get() << endl;

		FancyText t2("Fancy", "<<", ">>", "***");
		t2.append("A");
		cout << t2.get() << endl;

		FixedText t3;
		t3.append("A");
		cout << t3.get() << endl;
		t1 = t2; // Base <- Derived 가능
		//t2 = t1; // Derived <- Base 불가능
```

```
	return 0;
}
```

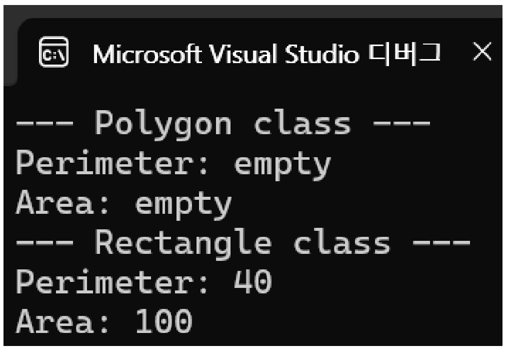

Microsoft Visual Studio

```
PlainA
<<Fancy***A>>
FIXED
```

<응용문제>

1.

```
#include <iostream>
#include <string>
using namespace std;

class Polygon {
public:
	Polygon() {}
	Polygon(int point, float length) {
		mPoint = point;
		mLength = length;
	}
	~Polygon() {}
	virtual void calcPerimeter() {
		cout << "Perimeter: empty" << endl;
	}
	virtual void calcArea() {
		cout << "Area: empty" << endl;
	}
protected:
	int mPoint; // 꼭지점의 갯수
	double mLength; // 한 변의 길이
};

class Rectangle : public Polygon {
public:
	Rectangle() {}
	Rectangle(int point, float length) : Polygon(point, length){}
	~Rectangle() {}
	void calcPerimeter() override {
		cout << "Perimeter: " << (mPoint * mLength) << endl;
	}
	void calcArea() override {
		cout << "Area: " << (mLength * mLength) << endl;
	}
};

int main() {
	Polygon pol;
	Rectangle rec(4, 10);

	cout << "--- Polygon class ---" << endl;
	pol.calcPerimeter();
	pol.calcArea();
	cout << "--- Rectangle class ---" << endl;
	rec.calcPerimeter();
	rec.calcArea();
	return 0;
}
```



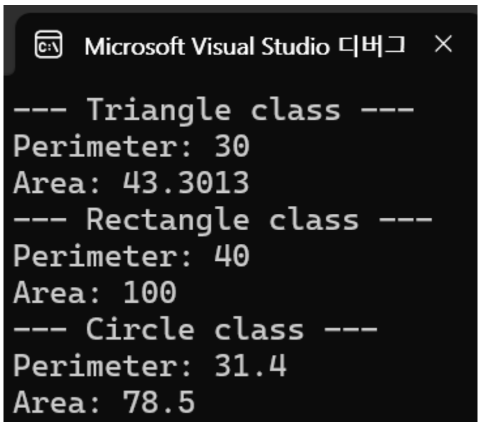
```
--- Polygon class ---
Perimeter: empty
Area: empty
--- Rectangle class ---
Perimeter: 40
Area: 100
```

2.

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

class Polygon {
public:
	Polygon() {}
	Polygon(int point, float length) {
		mPoint = point;
		mLength = length;
	}
	~Polygon() {}
```

```
		virtual void calcPerimeter() {
				cout << "Perimeter: empty" << endl;
		}
		virtual void calcArea() {
				cout << "Area: empty" << endl;
		}
protected:
		int mPoint; // 꼭지점의 갯수
		double mLength; // 한 변의 길이
};

class Triangle : public Polygon {
public:
		Triangle() {}
		Triangle(int point, float length) : Polygon(point, length){}
		~Triangle() {}
		void calcPerimeter() override {
				cout << "Perimeter: " << (mPoint * mLength) << endl;
		}
		void calcArea() override {
				cout << "Area: " << (sqrt(3)/4.0)*(mLength * mLength) << endl;
		}
};
class Rectangle : public Polygon {
public:
		Rectangle() {}
		Rectangle(int point, float length) : Polygon(point, length) {}
		~Rectangle() {}
		void calcPerimeter() override {
				cout << "Perimeter: " << (mPoint * mLength) << endl;
		}
		void calcArea() override {
				cout << "Area: " << (mLength * mLength) << endl;
		}
};
class Circle : public Polygon {
public:
		Circle() {}
		Circle(int point, float length) : Polygon(point, length) {}
		~Circle() {}
		void calcPerimeter() override {
				cout << "Perimeter: " << (2 * 3.14 * mLength) << endl;
		}
		void calcArea() override {
```

```
                cout << "Area: " << 3.14 * (mLength * mLength) << endl;
        }
};

int main() {
        Triangle tri(3, 10);
        Rectangle rec(4, 10);
        Circle cir(0, 5);
        cout << "--- Triangle class ---" << endl;
        tri.calcPerimeter();
        tri.calcArea();
        cout << "--- Rectangle class ---" << endl;
        rec.calcPerimeter();
        rec.calcArea();
        cout << "--- Circle class ---" << endl;
        cir.calcPerimeter();
        cir.calcArea();
        return 0;
}
```

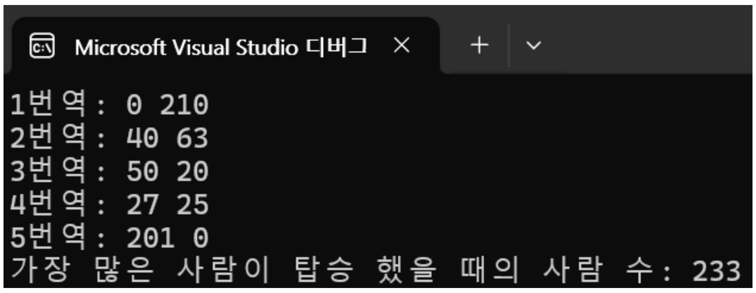


```
--- Triangle class ---
Perimeter: 30
Area: 43.3013
--- Rectangle class ---
Perimeter: 40
Area: 100
--- Circle class ---
Perimeter: 31.4
Area: 78.5
```

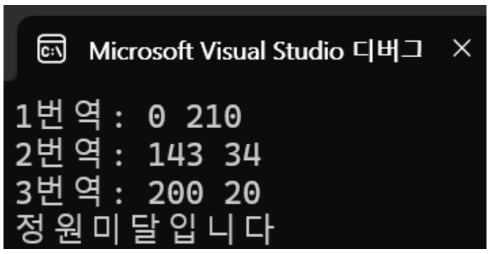
3.

```
#include <iostream>
using namespace std;

class Train {
public:
        Train() {}
        Train(int people) {
                mPeople = people;
        }
        ~Train() {}
```

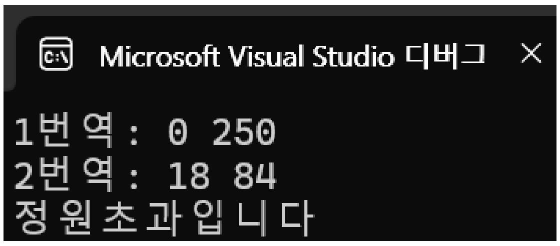
```
		virtual int station(int takeOff, int takeOn) { return 0; }
protected:
		int mPeople; // 사람 수
};
class Ktx : public Train {
public:
		Ktx() : Train(0) {}
		Ktx(int people) : Train(people) {}
		~Ktx() {}
		// 기차에 사람이 타고 내리는 함수
		int station(int takeOff, int takeOn) {
				if (mPeople < takeOff) {
						cout << "정원미달입니다";
						exit(EXIT_FAILURE);
				}
				mPeople = mPeople - takeOff + takeOn;
				if (mPeople > 300) {
						cout << "정원초과입니다";
						exit(EXIT_FAILURE);
				}
				return mPeople;
		}
		int getPeople() {
				return mPeople;
		}
};
int main()
{
		Ktx k;
		int a, b;
		int m = 0;
		for (int i = 1; i <= 5; i++) {
				cout << i << "번역: ";
				cin >> a >> b;
				k.station(a, b);
				if (k.getPeople() > m) {
						m = k.getPeople();
				}
		}
		cout << "가장 많은 사람이 탑승 했을 때의 사람 수: " << m;
		return 0;
}
```



```
1번역: 0 210
2번역: 40 63
3번역: 50 20
4번역: 27 25
5번역: 201 0
가장 많은 사람이 탑승 했을 때의 사람 수: 233
```



```
1번역: 0 210
2번역: 143 34
3번역: 200 20
정원미달입니다
```

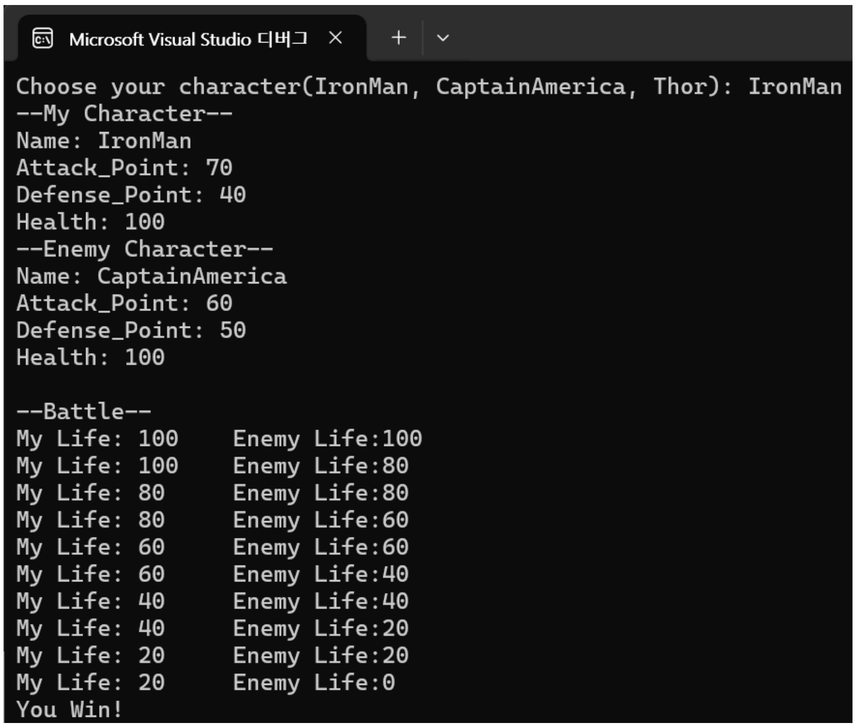


```
1번역: 0 250
2번역: 18 84
정원초과입니다
```

4.

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
using namespace std;

class Avengers {
public:
    Avengers() {
        name = "";
        attack_point = 0;
        defense_point = 0;
        health = 0;
    }
    ~Avengers() {}
    // 캐릭터 설정 함수
    virtual void set(string _name, int _attack, int _defense, int _health) {}
    // 공격 함수
    virtual int attack() { return 0; }
    // 방어 함수
    virtual void defense(int _attack_point) { }
    // 캐릭터 정보 출력 함수
    virtual void print_info() { }
protected:
    string name;   // 캐릭터 이름
    int attack_point; // 공격력
    int defense_point; // 방어력
    int health;   // 체력
};

class Character : public Avengers {
public:
    void set(string n) {
        if (n == "IronMan") {
            name = "IronMan";
            attack_point = 70;
            defense_point = 40;
            health = 100;
        }
        else if (n == "CaptainAmerica") {
            name = "CaptainAmerica";
            attack_point = 60;
            defense_point = 50;
            health = 100;
        }
        else if (n == "Thor") {
            name = "Thor";
            attack_point = 80;
            defense_point = 30;
            health = 100;
        }
    }
    int attack() override {
        return attack_point;
    }
    void defense(int a) override {
        health -= (a - defense_point);
    }
    void print_info() override {
        cout << "Name: " << name << endl;
        cout << "Attack_Point: " << attack_point << endl;
        cout << "Defense_Point: " << defense_point << endl;
        cout << "Health: " << health << endl;
    }
    int get_health() {
        return health;
    }
};

int main() {
    Character my_char;
Character enemy_char;

srand(time(0));
string a, b;
cout << "Choose your character(IronMan, CaptainAmerica, Thor): ";
cin >> a;
my_char.set(a);
int n = rand() % 3;
if (n == 0)
	b = "IronMan";
else if (n == 1)
	b = "CaptainAmerica";
else if (n == 2)
	b = "Thor";
enemy_char.set(b);

cout << "--My Character--" << endl;
my_char.print_info();
cout << "--Enemy Character--" << endl;
enemy_char.print_info();
cout << endl << "--Battle--" << endl;
cout << "My Life: " << my_char.get_health() << "₩t"
	<< "Enemy Life:" << enemy_char.get_health() << endl;

int i = 0;
while (1) {
	if (i % 2 == 0) {
		enemy_char.defense(my_char.attack());
	}
	if (i % 2 != 0) {
		my_char.defense(enemy_char.attack());
	}
	cout << "My Life: " << my_char.get_health() << "₩t"
		<< "Enemy Life:" << enemy_char.get_health() << endl;
	if (my_char.get_health() <= 0) {
		cout << "Enemy Win!";
		break;
	}
	else if (enemy_char.get_health() <= 0) {
		cout << "You Win!";
		break;
	}
	i++;
}
```

```
    return 0;
}
```



```
Choose your character(IronMan, CaptainAmerica, Thor): IronMan
--My Character--
Name: IronMan
Attack_Point: 70
Defense_Point: 40
Health: 100
--Enemy Character--
Name: CaptainAmerica
Attack_Point: 60
Defense_Point: 50
Health: 100

--Battle--
My Life: 100    Enemy Life:100
My Life: 100    Enemy Life:80
My Life: 80     Enemy Life:80
My Life: 80     Enemy Life:60
My Life: 60     Enemy Life:60
My Life: 60     Enemy Life:40
My Life: 40     Enemy Life:40
My Life: 40     Enemy Life:20
My Life: 20     Enemy Life:20
My Life: 20     Enemy Life:0
You Win!
```