```
// draw_shape.h
class Canvas {
private:
    // 그려진 모양을 저장할 수 있도록 데이터멤버를 정의 (resize 가능에 주의)
    friend ostream& operator<<(ostream& os, const Canvas& c);
public:
    Canvas(size_t row, size_t col);
    ~Canvas();
    // Canvas 크기를 w x h 로 변경한다. 그려진 내용은 보존한다
    void Resize(size_t w, size_t h);
    // (x,y) 위치에 ch 문자를 그린다. 범위 밖의 x,y 는 무시한다
    bool Draw(int x, int y, char brush);
    // 그려진 내용을 모두 지운다 ('.'으로 초기화)
    void Clear();
};

class Shape {  // 도형의 공통 속성을 정의
public:
    virtual ~Shape();
    virtual void Draw(Canvas* canvas) const = 0;
protected:
};

class Rectangle : public Shape { /* 필요한 멤버를 정의 */ };
class UpTriangle : public Shape { /* 필요한 멤버를 정의 */ };
class DownTriangle : public Shape { /* 필요한 멤버를 정의 */ };
class Diamond : public Shape { /* 필요한 멤버를 정의 */ };

istream& operator>>(istream& is, Rectangle& r);
istream& operator>>(istream& is, UpTriangle& t);
istream& operator>>(istream& is, DownTriangle& d);
istream& operator>>(istream& is, Diamond& dm);
```

```
// draw_shape_main.cpp
#include <iostream>
#include <string>
#include <vector>
#include "draw_shape.h"

using namespace std;
int main() {
    vector<Shape*> shapes;
    size_t row, col;
    cin >> row >> col;
    Canvas canvas(row, col);
    cout << canvas;

    while (true) {
        string tok;
```

```
        cin >> tok;

        if (tok == "add") {
            string type;
            cin >> type;
            if (type == "rect") {
                Rectangle* shape = new Rectangle();
                cin >> *shape;
                shapes.push_back(shape);
            }  else if (type == "tri_up") {
                UpTriangle* shape= new UpTriangle();
                cin >> *shape;
                shapes.push_back(shape);
            }  else if (type == "tri_down") {
                DownTriangle* shape = new DownTriangle();
                cin >> *shape;
                shapes.push_back(shape);
            }  else if (type == "diamond") {
                Diamond* diamond = new Diamond();
                cin >> *diamond;
                shapes.push_back(diamond);
            }  else continue;
        } else if (tok == "draw") {
            canvas.Clear();
            for (int i = 0; i < shapes.size(); ++i)
                shapes[i]->Draw(&canvas);
            cout << canvas;
        } else if (tok == "delete") {
            int index;
            cin >> index;
            if(index < shapes.size())
                shapes.erase(shapes.begin()+index);
        } else if (tok == "dump") {
            for(int i=0;i<shapes.size();i++){
                if(shapes[i]->type()=="rect")
                    cout << i << ' '<< "rect" <<' '<< shapes[i]->x() << ' ' << shapes[i]->y() <<
' ' << shapes[i]->w() << ' ' << shapes[i]->h() << ' ' << shapes[i]->brush() << endl;
```