문제 정의
저번 과제에서 만들었던 그래픽 편집기를 vector를 활용하여 구현하는 문제이다.
사용자가 도형(선, 원, 사각형)을 삽입, 삭제, 표시할 수 있어야 한다.
도형 목록을 동적으로 관리하여 다양한 작업을 수행 가능하게 해야 한다.
도형 삽입 시 동적으로 메모리를 할당하고 삭제 시 메모리를 해제하며 리스트를 관리해야 한 vector를 사용해 구현.

문제 해결 방법
기존의 포인터 기반 연결 리스트를 사용해서 만든 코드는 각 도형 객체를 Shape 포인터로 연결하여 리스트 구조를 구성하고 삽입 시 새 객체를 생성하고 마지막 노드에 연결. 그 후 삭제 시 특정 노드를 찾아 전 노드와 연결을 갱신하며 메모리를 해제한다. 도형 표시 시 첫 노드부터 순차적으로 탐색하여 모든 도형을 출력한다.
vector를 사용하여 만든 코드는 모든 도형 객체를 vector<Shape*>에 저장하여 관리하며 삽입 시 push_back으로 새 객체 추가, 삭제 시 인덱스를 기준으로 해당 객체를 erase하고 메모리를 해제한다. 도형 표시 시 for 반복문을 사용하여 vector를 순회하며 출력한다.

아이디어 평가
기존의 포인터 기반 연결 리스트의 코드와 비교를 해보겠다.

기준	포인터 기반 연결 리스트	vector
구현 난이도	상대적으로 복잡	간단하다
메모리 관리	수동적으로 동적 메모리 할당/해제	수동적으로 동적 메모리 할당/해제
가독성 및 유지 보수	코드가 복잡하고 디버깅이 어려움	코드가 간결하고 유지보수 및 디버깅 용이
메모리 연속성	없음	있음

결론적으로, vector를 사용하는 방식이 코드 가독성과 유지보수 측면에서 더 유리하며, 연결 리스트의 장점이 크게 필요하지 않은 상황에서는 적합한 선택이라할 수 있다.

문제를 해결한 키 아이디어 및 알고리즘
push_back 메서드를 이용한 새 도형 객체 추가, erase 메서드를 사용한 도형 제거 및 메모리 해제, for 반복문으로 vector의 객체를 순회하며 paint()메서드를 호출해 출력
이와 같이 삽입, 삭제, 조회 등의 작업이 메서드로 쉽게 구현이 가능해진다.
코드 간결성, 유지보수성, 구현 효율성 면에서 크게 향상이 되어 vector를 활용할 수 있으면 활용하는 방향이 좋다고 생각 됨