**4주차 예비보고서**

**전공: 경영학과 학년: 3학년 학번: 20190963 이름: 한다현**

**1.**

**4주차 실험에서는 객체 지향 프로그래밍의 다형성, 특히 파라미터적 다형성과 서브타입 다형성을 구현하는 것이 목표이다. 이번 실험은 linked list 자료구조를 사용하는데, 이를 위해 연결 리스트 역할을 할 Node라는 클래스를 작성한다. 이후에 기반 클래스인 LinkedList 클래스를 구현한다. 이 클래스는 첫 번째 노드를 가리키는 first 변수와 현재 노드의 개수를 저장하는 current\_size 변수가 protected로 선언되어 있고, first와 current\_size가 0인 새로운 linked list를 생성하는 생성자와, 리스트의 노드의 개수를 반환하는 int GetSize() 함수, 맨 앞에 원소를 삽입하는 void Insert(int element) 함수, 맨 뒤의 원소를 삭제하는 virtual bool Delete(int &element) 함수, 그리고 리스트를 출력하는 void Print() 함수로 구성되어 있다. 위와 같이 클래스를 작성할 경우, element의 자료형이 정수형이 아닌 경우에는 클래스의 멤버 함수를 사용할 수 없다. 따라서 c++의 파라미터적 다형성, 즉 템플릿을 사용하여 이러한 문제를 해결한다. 이는 저장되는 데이터의 자료형을 임의의 타입으로 바꾸고, 클래스의 선언에 템플릿 자료형이라는 선언부를 추가한 후에 클래스의 이름을 LinkedList<템플릿 자료형>으로 수정하면 된다. 템플릿으로 클래스를 작성한 후에는, 기반 클래스를 상속하여 스택 자료구조를 제공하는 Stack 클래스를 구현한다. 기반 클래스인 LinkedList 클래스는 queue의 기능을 하지만 Stack 클래스는 가장 나중에 들어온 노드가 먼저 삭제되는 스택의 기능을 하도록 구현한다. 따라서 앞서 구현한 템플릿 기반의 LinkedList 클래스를 상속한 후에 Delete() 함수만 재정의하여 구현하면 된다. 이러한 방식이 서브타입 다형성을 의미하는데, Stack 클래스는 기반 클래스의 포인터나 참조를 사용하여 클래스에 접근하기 때문에 재정의하는 함수를 virtual로 선언해야 한다. 앞서 Delete() 함수를 virtual로 선언한 이유이다. LinkedList 클래스와 Stack 클래스를 모두 구현했다면 자료 구조를 선택하고, 삽입, 삭제, 출력, 종료 중 원하는 기능을 선택하여 해당 명령에 대해 적절한 기능을 하도록 main 함수를 작성한다.**