4주차 예비보고서

전공: 신문방송학과 학년: 3학년 학번: 20191150 이름: 전현길

1. 예비보고서

주어진 문제 및 3-1의 문제 해결에 관한 내용을 이해하고 이 문제를 효율적으로 해결하기 위한 방법을 생각하여 이를 1쪽 이내로 요약하여 제출하시오. 문제 해결을 위한 간단한 단계별 수행 내용, 자료구조 등을 기술하시오.

1) 저장되는 변수를 template 자료형으로 변경

2) 후입선출 특성 구현을 위해 Delete() 함수를 Stack 클래스에 재정의(overriding)

LinkedList에 대한 선언부, 구현부 코드가 거의 모두 주어졌기 때문에, 문제를 해결하는 것은 그리 어렵지 않다. 파라미터적 다형성을 구현할 수 있도록 Node가 저장하는 **데이터를 template 자료형으로 변경**해 주고, LinkedList 클래스의 멤버 함수인 void Insert(int element), virtual bool Delete(int &element) 함수가 각각 template 자료형을 삽입하고 삭제할 수 있도록 바꾸어 주면 된다.

이후에 서브타입 다형성을 통해 LinkedList의 파생 클래스(자식 클래스)인 Stack을 구현하되, **Stack의 LIFO(후입선출) 특성을 구현하기 위해 Delete() 함수를 통해 Stack에 따로 구현**해 주면 된다. 부모 클래스와 자식 클래스에 같은 이름의 함수가 존재할 경우, 자식 클래스의 함수가 호출되므로 똑같은 이름의 함수를 구현하는 것만으로 요구하는 기능을 만족할 수 있다.

단, 다음 두 가지 사항을 추가적으로 고려할 필요가 있다. 먼저, Stack 클래스의 멤버 함수에서 LinkedList 클래스(=부모 클래스)의 멤버 변수인 Node *first에 접근해야 하므로 protected 접근 지정자를 사용해야 한다. 이는 기본 구현에서 이미 접근 지정자를 protected로 선언하였기 때문에 고려할 필요가 없다. 둘째로, 나중에 Stack 자료형을 LinkedList* 포인터로 받았을 때 Stack의 멤버 함수 Delete() 대신 LinkedList의 멤버 함수 Delete()를 불러올 수 있도록 하기 위해 부모 클래스의 Delete() 함수를 virtual 함수로 선언해야 한다. 이 역시 기본 구현에서 virtual 함수로 구현되어 있으므로 고려할 필요가 없다.

Delete() 함수의 구체적 구현

1) 삭제 성공 시 true, 실패 시 false 반환

2) first가 NULL일 경우 삭제 실패

3) current 노드 포인터가 first를 가리키게 함, first는 first->link를 가리키도록 한 뒤 current 삭제(메모리 반환)