4주차 예비보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

Stack 클래스는 LinkedList 클래스를 상속받으므로, 문제해결을 효율적으로 하기 위해서

먼저 LinkedList 클래스의 Print()함수를 구현하고 Template을 사용해 LinkedList 클래스를 파라미터적 다형성이 지원되게 바꾼다. 그 다음 Stack 클래스에서 LinkedList 클래스를 상속받은 뒤 Delete() 함수를 재정의해 Stack 클래스를 완성하면 된다.

Print()의 구현 : LinkedList, Stack클래스 모두 연결리스트를 사용한 자료구조이므로 current\_size가 0이 아니면 첫번째 노드를 가리키는 first의 값을 루프용 변수 i에 복사하고 i가 null이 될 때 까지 리스트를 순회(i=i->link)하면서 노드에 저장된 값을 출력한다.

(이때 출력의 형식에 맞게 if문을 사용하여 i == current\_size면 화살표를 출력하지 않는다.)

모든 노드의 값의 출력이 완료되면 마지막에 cout<<endl;으로 개행문자(‘\n’)를 출력한다.

LinkedList 클래스에 파라미터 다형성 적용 (Template 사용) :

Node, LinkedList, Stack의 클래스 선언부에 template <typename T>를 추가해서 템플릿 클래스로 만든다. 그리고 Node의 데이터를 담고있는 자료형과 Insert 및 Delete 함수의 자료형을 모두 템플릿 변수 T로 변경하고, 클래스 이름 뒤에 <T>를 붙인다.

(Node를 Node<T>로, LinkedList를 LinkedList<T>로, Stack을 Stack<T>로 변경)

Delete()의 재정의 : LinkedList와 다르게 Stack클래스는 Stack의 성질을 만족해야 하므로 Delete() 호출시 맨 끝이 아닌 맨 앞의 노드가 삭제되게 구현해야 한다.

먼저 current\_size가 0이 아니면 current(Node<T> 포인터 타입)에 first의 값을 저장한다.

(current\_size가 0이면 false를 리턴하고 함수를 종료한다.)

그 다음 element에 current->data의 값을, first에 current->link의 값을 저장해서

첫번째 노드를 리스트에서 삭제한 다음, delete current;를 사용해 리스트에서 삭제된 노드에 할당된 메모리를 해제하고 current\_size의 값을 1 감소시킨다.

마지막으로 true를 리턴하고 함수를 종료한다.

자료구조 : LinkedList 클래스는 자료구조 중 연결리스트를 사용한 Queue에 해당한다.

Stack 클래스는 자료구조 중 연결리스트를 사용한 Stack에 해당한다.