오픈소스 응용 프로그래밍

- 3-1. 팀 과제 공지 -

YOUNGHO KIM

yhkim85@inha.ac.kr

본 사이트에서 수업 자료로 이용되는 저작물은 저작권법 제25조 수업목적저작물 이용 보상금제도에 의거, 한국복제전송저작권협회와 약정을 체결하고 적법하게 이용하고 있습니다. 약정범위를 초과하는 사용은 저작권법에 저촉될 수 있으므로 수업자료의 대중 공개·공유 및 수업 목적 외의 사용을 금지합니다.

2020. 3. 인하대학교·한국복제전송저작권협회

팀 구성 및 일정

- 팀 구성
  - 4명 팀: 3, 4학년 학생 2명 / 2학년 학생 2명 (외국인 학생은 학년 제한 없음)
  - 3명 팀: 3, 4학년 학생 2명 / 2학년 학생 1명 (팀을 4명으로 구성하지 못할 경우)
  - 모두 다 깃 버전 관리 및 깃 허브 코드 리뷰 시스템 (fork & pull request) 이용 필수
  - 채점 시 해당 내역을 통하여 팀원들의 참여 유무 1차 판단 예정
- 일정
  - 10/08 수업: 과제 세부내용 공지 (6주차)
  - 11/19(화) 수업: 중간 발표 순서 추첨 + Q&A 세션 (12주차)
  - 11/20(수) 23:59: 중간 발표 자료 제출
  - 11/21~12/5 수업: 팀별 중간 발표 (하루 당 3팀 발표, 발표가 아닌 팀은 팀과제 진행)
    - 발표(15분) + 질의응답(5~10분)
  - 12/20 (16주차) 23:59: 채점 서버를 활용한 코드 기능성 채점 및 최종 보고서 제출 (팀당 하나)

문제
STL(Standard Template Library) set 구현

- AVL 트리를 이용하여 오픈소스인 STL set과 관련된 함수(연산) 및 추가 함수 구현
  - AVL 트리는 정수 타입 (32-bit integer) 의 key를 저장
  - 기본 기능: Find, Insert, Empty, Size, Height, Ancestor, Average (슬라이드 7 참고)
  - 고급 기능: Rank, Erase (슬라이드 8 참고)

- 용어 정리
  - 원소는 AVL 트리의 내부노드(internal node)에 저장한다.
  - AVL 트리의 루트(root)의 깊이(depth)는 0이다.
  - 트리의 높이는 트리의 노드들의 깊이 중 최대 깊이이다.
  - 노드 x의 높이는 x가 루트인 부분트리의 높이이다. AVL 트리의 외부노드(external node, nullptr 또는 dummy node로 구현)의 높이는 0이다.

- 채점 서버를 통해 기능의 정확성(correctness) 및 효율성(efficiency)을 확인하고 평가함 (16주차)

채점 기준

- 팀 과제는 다음의 내용으로 100점 만점으로 평가
  - 라이선스 명시 및 작성자 명시: 5점
  - 코드 기능성: 35점 (기본 기능: 25점, 고급 기능: 10점)
  - 코딩 스타일 (포맷 및 주석 포함): 10점
  - 버전 관리 및 코드 리뷰 과정: 10점
  - 테스트 (구글테스트): 15점
  - 중간 발표: 10점
  - 최종 보고서: 15점

  ※ 마지막에 동료 평가를 통해, 팀원 간 점수가 조정될 수 있음

라이선스 명시 및 작성자 명시: 5점

- 프로젝트 저장소에 라이선스 명시를 해야 됨
  - GitHub의 저장소 라이선스 명시 기능을 사용
- 각 파일들의 최상단에 주석으로 라이선스 명시가 있어야 됨
- 라이선스를 포함하는 주석은 다음과 같은 조건을 만족 해야 됨
  - 라이선스들이 **모든 파일들에** 일관되게 적혀 있어야 됨
  - 라이선스명과 라이선스를 설명하는 문구가 일치 되어야 함
  - 라이선스는 permissive 혹은 copyleft 라이선스를 사용 해야 됨
  - 각 파일들 별 라이선스 관련 주석의 최하단에 작성자 및 작성일이 명시 되어 있어야 됨
- 구글링을 하거나 GitHub에서 검색 가능한 타 프로젝트의 예시들을 참고할 것을 권장함

코드 기능성: 35점

- STL set과 관련된 기본 기능과 고급 기능을 구현하고, 채점 서버를 이용하여 기능들의 정확성을 평가

- 기본 기능: 반드시 구현해야 할 기능 (기본 기능이 모두 채점서버에서 제대로 작동해야 25점)
  - Find x: 노드 x ($1 \leq x \leq 300{,}000$)의 깊이와 높이의 합을 출력한다. 만약 노드 x가 없다면, 0을 출력한다.
  - Insert x: 새로운 노드 x를 삽입하고, 노드 x의 깊이와 높이의 합을 출력한다. 이때 x는 현재 set에 존재하지 않은 값으로 주어진다.
  - Empty: set이 비어 있으면 1을 출력하고, 비어 있지 않으면 0을 출력한다.
  - Size: set에 저장된 원소의 수를 출력한다.
  - Height: AVL 트리의 높이를 출력한다. 만약 트리가 비어 있다면, -1을 출력한다.
  - Ancestor x: 노드 x의 깊이와 높이의 합, x의 부모부터 루트까지의 경로의 노드들의 key 값들의 합을 공백으로 구분하여 출력한다. 이때 x는 현재 set에 저장되어 있는 원소로만 주어진다. 만약 x가 루트인 경우, 'K 0'이 출력된다. 이때 K는 x의 깊이와 높이의 합이며, 현재 AVL 트리의 형태에 따라 값이 다를 수 있다.
  - Average x: 노드 x가 루트인 부분트리에서 노드들의 key 값의 최솟값(a)과 최댓값(b)의 산술평균((a+b)/2)을 출력한다. 이때 x는 현재 set에 저장되어 있는 원소로만 주어진다.

코드 기능성: 35점

- 고급 기능: 선택적으로 구현할 수 있는 기능 (고급 기능이 모두 채점서버에서 제대로 작동해야 10점)
  - Rank x: 노드 x의 깊이와 높이의 합, 순위(rank)를 공백으로 구분하여 출력한다. 만약 노드 x가 없다면, 0을 출력한다.
    - x의 순위는 set에서 'x보다 작은 원소의 수' + 1로 정의한다. 예를 들어, set에 원소 1, 3, 5, 7이 저장되어 있다면, 3의 순위는 2, 7의 순위는 4이다.
  - Erase x: 노드 x의 깊이와 높이의 합을 출력하고 해당 노드를 삭제한다. 만약 노드 x가 없다면, 0을 출력한다.
    - 삭제 연산에서 x의 두 자식이 모두 내부노드인 경우 후임자(successor)를 이용하여 삭제 수행한다.

코드 기능성: 35점

- 효율성
  - Empty, Size, Height는 O(1) 시간에 수행해야 한다.
  - 나머지 연산들은 각각 O(log(n)) 시간 내에 수행해야 한다.
  - 테스트 케이스에 대한 구체적인 시간 제한은 추후 채점 서버 문제로 공개함

코딩 스타일 (포맷 및 주석 포함): 10점

- 프로젝트의 모든 코드들은 CSE3210 스타일 가이드를 따라야 됨
- 코드 포맷(예: 라인 당 글자 수 등) 은 CSE3210 스타일 가이드에 포함 되어 있지는 않지만 포맷을 위해 사용한 방법이나 기준을 명시해 줘야 됨
  - clang-format / Windows 의 기본 코드 포맷 도구들 혹은 타 운영체제의 적절한 코드 포맷 도구들을 활용하여 코드 포맷을 맞춰 주면 좋음
  - 수동으로 맞춰 주었을 경우, 어떤 기준으로 맞춰 주었는지 그리고 모든 소스 코드들에 해당 기준을 일관적으로 적용하기 위하여 어떠한 방법을 사용했는지를 명시해 줘야 됨
- 각 파일 별로 라이선스 주석은 필수로 요구함
- 이 외의 주석은 타 팀원들이 해당 소스코드를 보았을 때 곧장 소스 코드를 이해할 수 있을 정도로 추가해줘야 됨
  - 예: 클래스 및 메소드 별 설명 – getter/setter 메소드 제외, 복잡한 반복 연산에 대한 설명

버전 관리 및 코드 리뷰 과정: 10점

- 각 팀의 한 명(가급적 팀장)이 저장소를 GitHub에 생성하고 팀원들과 공유해야 됨
  - 팀의 저장소는 private로 지정하고, 이를 collaborator 기능을 통하여 팀원과 공유해야 됨
  - Collaborator 기능을 통하여 분반 교수를 해당 저장소의 공동 작업자로 추가해야 됨
  - Collaborator로 팀 저장소에 포함되지 않은 분은 팀 과제 점수 부여 불가
  - 각 팀원은 팀 메인 저장소를 fork하여 작업 후 pull request를 통하여 작업 내용을 반영해야 됨
  - 팀 메인 저장소를 만든 팀원도 해당 저장소를 fork하여 작업용 저장소를 만든 후 pull request 해야 됨
- 코드 개발 및 코드 리뷰 과정은 Git의 버전 관리 기능과 GitHub의 pull request를 활용해야 됨
  - 변경 사항 반영을 위하여 pull request 요청 시 필히 한 명 이상의 reviewer를 지정해야 됨
  - Git의 세부적인 기능들의 활용 여부는 채점 기준에 들어가지 않음
  - 예를 들어, reset, branch 등을 활용 한 것은 채점 기준에 들어가지는 않음
  - Git 활용도를 보여주기 위하여 최종 보고서에서는 Git을 어떻게 활용을 했는지 언급해 주는 것을 권장함
- 모든 팀원은 코드 개발(테스트 코드 포함) 및 코드 리뷰에 일정 비율로 참여해야 됨
- 버전 관리 및 코드 리뷰 과정을 통하여 팀원들 중 누가 어떤 부분을 개발 하였는지, 그리고 어떠한 부분을 리뷰하였는지 보여줄 수 있어야 됨
- 코드 리뷰 요청 및 코드 리뷰 시, ”change” 등의 무의미한 내용이 아닌, 적절한 내용을 작성해야 함

테스트 (구글테스트): 15점

- 테스트는 필수로 작성해야 되며 구글테스트를 활용해야 됨
- 구글테스트를 활용한다면 어떤 방식으로 사용해도 상관 없음
  - WSL과 Cmake를 활용 (권장)
  - Windows visual studio의 구글테스트 통합 기능 활용
  - Docker를 활용
  - 기타 다른 OS에(예: Ubuntu, Fedora, MacOS) CMake를 설치 후 활용
    - Apple silicon 하드웨어 노트북들은 구글테스트 설치가 원할하지 않다는 보고가 많이 있음
- 작성한 모든 함수(생성자, 소멸자 포함)에 대한 테스트가 존재해야 됨
- Fixture 및 value-parameterized test 필수
- 테스트에 적절한 expect/assert 활용 필수
- Matcher는 사용해도 되고, 사용하지 않아도 됨

중간 발표: 10점

- 중간 발표 (발표(15분) + 질의 응답(5~10분))
  - 아래의 outline 내용들을 필수로 포함해야 되며, 기타 다른 내용들을 포함하여도 됨
  - 중간 발표 outline
    - 팀원 소개 (역할분담 포함)
    - 설계, 구현, 테스트 관련
      - 코드 디자인 (클래스 구조도) 및 해당 디자인 선택 이유
      - 중간 발표자료까지의 코드 구현 설명 (아주 간략하게)
      - 스타일 가이드 및 포맷과 주석의 일관성을 효율적으로 지키기 위하여 사용한 방법들
      - 팀에서 효율적으로 리뷰 과정을 수행하기 위하여 사용한 방법들
      - 현재 테스트 코드 상태 혹은 수행 계획
      - 이후 남은 과제 기간 동안의 계획
    - 느낀 점 및 궁금한 점

최종 보고서: 15점

- 최종 보고서
  - 아래의 outline 내용들을 필수로 포함해야 되며, 기타 다른 내용들을 포함하여도 됨
  - 최종 보고서 outline
    - 팀원 소개 (역할분담 포함)
    - 설계 및 구현 관련
      - 코드 디자인 (클래스 구조도) 및 해당 디자인 선택 이유
      - 각 클래스들 별 상세 디자인 및 구현 시 내린 여러 결정들에 대한 내용들
      - 코드 구현 설명
      - 스타일 가이드 및 포맷과 주석의 일관성을 효율적으로 지키기 위하여 사용 한 방법들
      - 팀에서 효율적으로 리뷰 과정을 수행하기 위하여 사용 한 방법들
      - 팀의 개발 및 리뷰 과정 요약
    - 테스트 관련
      - 테스트 설계 시 원칙
      - 테스트 케이스들 요약본 및 결과
    - 느낀 점

질문?