Programming Project 2

프로젝트 요약

- 구현 목표
  - 연결리스트의 활용한 사전 구현
- 구현 내용
  - 텍스트 입력으로부터 데이터 구조의 생성
    - Linked list 사용
  - 특정 단어의 번역 단어 탐색
  - 특정 단어의 번역 단어 치환
  - 특정 단어의 번역 단어 삭제
  - 탐색 경로의 출력

프로젝트 상세

- 입력의 형태
  - 첫째 줄: 사전에 단어별로 넣을 입력 문장

  - 번역 단어
    - 각 어휘가 문장에서 나타나는 순서
    - 이미 발견된 어휘는 번역 단어가 이미 결정된 것으로 취급
  - 예)
    - I do not really want to do this exhaustive coding .
      - I: 0, do: 1, not : 2, really : 3, want : 4, to : 5, this : 6, exhaustive :7, coding :8, . : 9
    - The the the the
      - The: 0
      - the: 1

프로젝트 상세

- 입력의 형태
  - 첫째 줄: 사전에 단어별로 넣을 입력 문장

  - 데이터 구조:
    - 2차원 연결리스트
    - 1차원 연결리스트의 구조
      - 단어의 첫 번째 character들이 value
      - Head -> "a" -> "b" -> "c" -> "d" -> "e" ... -> "A" -> "B" -> ... "Z" -> "."
    - 1차원 연결리스트의 node는 2차원 연결리스트의 header
      - "a" -> "apple" -> "area"
      - "c" -> "count" -> "check" -> "cipher"

프로젝트 상세

- 입력의 형태
  - 첫째 줄: 사전에 단어별로 넣을 입력 문장

  - 데이터 구조:
    - 2차원 연결리스트의 노드는 값 두 개로 구성
      - Key, value
      - Key는 단어 (영어 대/소문자 alphabet)
      - Value는 숫자
    - 2차원 연결리스트의 노드 순서는 처음 주어진 입력 문장에서 관측되는 순서
    - 정렬 불필요
    - Sub 기능으로 대체되는 경우 해당 리스트의 제일 뒤에 추가

프로젝트 상세

- 입력의 형태
  - 첫째 줄: 사전에 단어별로 넣을 입력 문장
  - 그 이후 줄: del, sub, sch 중 하나의 operator 입력
  - del xxxx
    - 나타나는 xxxx 삭제
  - sub xxxx yyyy
    - 나타나는 xxxx를 yyyy로 치환
  - sch xxxx
    - 나타나는 xxxx를 탐색
    - xxxx를 찾기위해 방문한 연결 노드들의 값들을 모두 출력

- sample_input.txt의 내용

I want to play music .

sub play clay

sch clay

del I

del music

sch .

0

프로젝트 상세

- Substitution시 삭제되는 node의 id를 생성되는 노드의 ID로 세팅
- 생성되는 노드가 있어야할 linked list에 이미 존재하는 경우, 존재하는 노드가 가진 ID 사용 (생성 불필요)
- Sch의 결과 출력시, value만 출력 (1차원 리스트의 원소 출력 불필요)
- String.h stdlib.h 등 standard library 사용 가능
- List 관련 standard library 함수 사용하지 말 것
- Sub 수행시 자기 자신으로 대체하는 경우는 구조 변경 불필요

프로젝트 상세

- “단어”의 정의는 띄어쓰기가 아닌 허용된 character들의 sequence
- 각 기능은 인자로 “단어”를 반드시 받음
- 입력 문장은 “단어”들의 sequence로 구성됨
- “단어 " 는 항상 길이가 0보다 큼 (empty string은 단어로 취급하지 않음)
- 입력 문장에서 띄어쓰기가 연속으로 나오더라도 empty string은 무시

프로젝트 상세

- Sch 수행시 해당 단어를 저장해야하는 리스트의 원소 수가 0인 경우 newline만 출력
- Sub 수행시 xxxx가 일치하는 노드가 없으면 기능 수행 안함 (무시)
- Del 수행시 xxxx가 일치하는 노드가 없으면 기능 수행 안함 (무시)
- Sch 수행시 xxxx가 일치하는 노드가 없으면 x로 시작하는 linked list내의 모든 원소의 id를 순서대로 출력
- Sch 수행 결과 출력시, id들 사이는 띄어쓰기 출력할 것. 제일 마지막 id 뒤에는 띄어쓰기 없이 newline출력할 것

프로젝트 상세

- 출력의 형태
  - 주어진 입력에 대해서 한 번 sch 수행할 때의 결과를 한 줄에 출력

- Sample_output.txt의 내용

3

5

프로젝트 상세

- 입출력 예2
- 입력)
- Chen can create the creature
- sch can
- sch create
- sch creature

- 출력)
- 1
- 1 2
- 1 2 4

프로젝트 상세

- 입출력 예3
- 입력)
- Chen can create the creature
- sch can
- del can
- sch create
- del create
- sch creature

- 출력)
- 1
- 2
- 4

프로젝트 상세

- 입출력 예4
- 입력)
- Chen can create the creature
- sch creatore

- 출력)
- 1 2 4

프로젝트 상세

- 사용자로부터 입력 받도록 설정
  - 프로그램 실행 후 사용자로부터의 입력 시작
  - 첫 줄이 입력 첫번째 줄
  - 그 후 이어지는 입력들이 각 기능들
  - 0 입력시 종료
- 패턴의 형태
  - 영어 알파벳 (대문자, 소문자 구분)과 마침표, 띄어쓰기만 사용 (입력 문장 및 패턴)
    - 숫자, 그 외 특수기호 (x)
  - 입력 문장의 길이 무제한
  - (마찬가지로 명령어 뒤에 따라오는 패턴들의 길이도 무제한)
  - 변환된 문장 외 불필요한 출력 금지 (공백문자, 사용자용 메시지, 오류메시지 등)

프로젝트 검증

- 검증 환경 테스트
  1. Server 접속
  2. /tmp/pp18/project2/eval.sh를 본인 홈폴더에 복사
     1. 홈폴더: 자신의 id가 ppst18일 때 /tmp/pp18/ppst18
  3. /tmp/pp18/project2/sample_input.txt를 본인 홈폴더에 복사
  4. /tmp/pp18/project2/sample_output.txt를 본인 홈폴더에 복사
  5. 구현이 완료된 자신의 코드와 compile.sh를 본인 홈폴더에 복사
  6. Compile시 생성되는 실행 파일은 본인의 홈폴더에 project2.out으로 저장
  7. $ ./eval.sh
  8. Success 혹은 Fail이 화면에 출력되면 평가 성공
  9. Fail시 자신의 코드의 Stdout으로의 출력 결과는 err 파일로 출력됨

프로젝트 평가 기준 (100점 만점)

- Code 유사도 검증
  - Code 유사도 100% 일치 – 대조 및 검증 없이 바로 F
  - Code 유사도 의심 수준 – 기본 점수 (0점)
  - 미제출 – (0점)

- Code 검증 통과한 경우
  - 임의의 평가 문장에 대해서
    모든 패턴의 출력결과 일치 (Success 출력) – 정확도 * 90점
    - 정확도 = Success 수/전체 입출력 문장 샘플 수
  - 샘플 평가 문장에 대해서
    모든 패턴의 출력결과 일치 (Success 출력) – 10점

프로젝트 제출 유의사항

- 프로젝트 제출 기한 (2주)
  - 5월 15일 밤 12:00
  - 시간 넘겨서 업로드 된 코드는 미제출 처리
  - 서버에 남는 시간 기록 기준
  - ll –tr로 자신이 올린 파일의 시간 확인 필수
- 체크 리스트
  - 자신의 프로젝트 ID
  - 서버 접속
  - 서버에 자신의 소스코드 카피
  - 샘플 입출력에 대한 eval.sh의 정상 동작