Team Project #1
- Scientific Calculator

- 목표
  - 공학용 계산기 구현
- 입력
  - 수식을 표현하는 문자열
  - 사용되는 표현
    - 사칙연산
    - 삼각함수
    - 지수/로그함수
    - 괄호
    - 레지스터

사칙연산, 지수, 로그함수의 표현

- +,-,*,/ : 사칙연산
  - 모두 실수형 연산 (출력 형태에 대해서는 정수, 실수 상관없이 값이 맞으면 됨. 소수점 자리수나 오버플로, 언더플로로 발생하는 오차는 무시. 정답값의 +- 1%로 오차 허용.)
- sin, cos : 삼각함수
  - sin(1), cos(1)로 사용
  - 들어가는 인자는 radian 값으로 인식
- exp : exponential function
  - exp(2) : e의 2승
- log : logarithmic function
  - log(4) : log_e 4 (자연로그)

괄호 표현

- () : 괄호
  - 반복 사용 가능

  예) (((4+5)*2) / (1+9))

레지스터

- -> [레지스터이름] : 레지스터 저장

예) 8+5 -> [x]

- [레지스터이름] : 레지스터값 호출

예) [x] + 5

- 호출을 이용한 계산시, 호출된 [x]의 연산 우선순위가 가장 높음 ([x] 먼저, + 는 나중)
- 레지스터 이름 규칙
  - 영문 알파벳 a~z까지 가능
  - 1 character만 사용 가능

부호

- (부호 expression): 부호 수식 표현법
- 부호는 +, -
- (+1) : 양수 1
- (-1) : 음수 1
- (+1 + 3) : 오류
- +1 오류
- +[x] 오류
- +(1+3) 오류
- 1+ (1+3) 오류 아님
- (+ sin(1)) 오류 아님
- (+ sin(+1)) 오류 아님

계산

- CAL : 계산

예) 1+ 5 CAL
결과: 6.0

- 계산 방식은 항상 [expression] CAL 하는 경우에만 결과 출력
- [expression] 표현 규칙에서 벗어나는 경우 ERROR 출력
- 레지스터 저장 구문은 [assignment]라고 할 때, 이 구문은 [expression] -> [register expression]과 동일
  예) 1+5 -> [x] 에서 1+5는 하나의 expression, [x]는 register expression

계산

- 계산기 입력 방식 예
  - 0 혹은 양수의 [assignment]
  - 하나의 [expression]
  - 즉, [assignment]…[assignment] [expression]
  - [assignment] 문 반복하는 표현 사이에 들어오는 expression은 사용하지 않는 것으로 가정
    - 예) [assignment] [expression] [assignment] CAL (비정상 입력)
    - 예) [assignment] [assignment] CAL (비정상 입력)
  - Assignment의 레지스터를 반복적으로 저장하는 경우 가장 나중에 assignment 값으로 저장
    - 예) 1+3 -> [x] 2+4 -> [x] [x] CAL
      - 결과 : 6

계산

- 계산 오류케이스
  - 계산시 위의 표현에 해당하지 않는 경우
  - [assignment1]에서 레지스터 값을 호출하는 경우, 이 변수를 저장하는 [assignment2]가 [assignment1]이전에 나오지 않은 경우
    - 예) 문제 없는 수식 2+ 5 -> [x] 3 + [x] -> [y] [y]
    - 예) 오류 2+ 5 -> [z] 3 + [x] -> [y] [y]
  - log (0)
  - [expression] / 0

사용자의 입력 방식

- 유저는 공백 줄바꿈 사용할 수 있음 (계산시 무시해서 처리할 것)
  - 줄바꿈등으로 입력시에 나눠져있는 reserved keyword도 붙어있는 것으로 처리
- Sequence에서 CAL이 발견되면 CAL 이전까지 나온 수식을 처리해서 결과를 출력하고 같이 입력된 버퍼에 저장된 표현들은 모두 무시
  - 버퍼를 사용하는 경우에만 해당, 버퍼를 안쓰면 CAL이 인식될 때 바로 결과 계산해서 출력
  - 버퍼에 EXIT가 따라나와도 무시
- 결과 출력 후 모든 상태 초기화(레지스터 포함)
- 상태 초기화 후 다시 수식 입력 대기
- 입력에서 EXIT 발견시 프로그램 종료
- 한 번의 계산에 사용되는 최대 expression의 길이는 100개 (공백, 줄바꿈 제외한 character 수)로 가정

예)
1 + (
2 *
3)
동일한 수식 1+(2*3)

예)
1 + (
2
3)
동일한 수식 1+(23)

예)
1 + (
s
in (4))
동일한 수식 1+(sin(4))

예)
1 + (
s
in (4)) C
AL
동일한 수식 1+(sin(4)) CAL

사용자의 입력 언어 (optional)

- 입력 문장의 language definition (context free grammar)

```
L   ->  A E T
    ->  E T
A   ->  A A
    ->  E -> [ R ]
E   ->  N
    ->  U ( E )
    ->  E B E
    ->  ( E )
    ->  ( W X )
    ->  U ( W X )
    ->  [ R ]
X   ->  N
    ->  U ( E )
    ->  ( E )
    ->  ( W X )
    ->  U ( W X )
    ->  [ R ]
B   ->  +
    ->  -
    ->  *
    ->  /
```

```
U   ->  sin
    ->  cos
    ->  exp
    ->  log
W   ->  +
    ->  -
R   ->  a
    ->  b
            .
            .
            .
    ->  z
N   ->  모든 숫자 (실수포함)
T   ->  CAL
    ->  EXIT
```

이 규칙들을 사용해서 최종적으로
L을 도출해낼 수 있는 언어를 사용자의 수식 입력 표현 방식으로 정의

코드 및 발표자료 제출 기한 :

- 6월 7일 24:00 까지 이메일로 제출
- 제출시 메일 제목은 다음 형식 준수
  [pp-team1]

- 발표 일정
  - 6월 8일 : team project group1
  - 6월 13일 : team project group2
  - 발표 순서는 랜덤 (미정)

발표 포함 내용

- 코드의 구성
- 팀의 구성
  - 1 project manager
  - 2 ~ 3 coders
- 구현 작업의 분배 (누가 어디 구현했는지 명시)
  - 분배시 작업 요청 내용
  - 분배시 협의된 테스트 케이스 내용
- 분배 작업별 테스트 코드 수행 데모
  - 기능 개별 테스트 코드 구현
  - 기능 구현이 안될 경우를 가정한 dummy data structure 사용하여 테스트
  - PM, coder1,2,3 각각 개별 기능 테스트
- 전체 코드 수행 데모
  - 자신이 가져온 데모 테스트 케이스
  - 임의로 들어오는 케이스 계산 수행 (제가 불러드립니다.)
- 발표시간 팀별 15분
- 보고서 제출하지 말 것, 발표자료로 대체

평가

- Coder
  - 협의된 test case 수행에 문제가 없으면 90
    - 작업에 비해 너무 협소한 test case의 경우 -10
- Project manager
  - 모든 기능 동작에 문제가 없는 경우 90
    - 개별 코더가 협의된 test case를 수행에 실패하는 경우, 감점 없음
      - Dummy structure로 진행
      - Dummy structure의 coverage가 너무 좁은 경우 -10
- 데모시 수행하는 임의 테스트 케이스까지 잘 돌아가는 경우 + 10 (모든 팀원)
- 코드 분배 협의가 독단적으로 진행되어 팀원이 발표 시점까지 동의하지 못한 상태로 진행되면 팀원 전체 0점 처리
- 6월 1일까지 연락이 안 되거나, 코드 분배 협의 회의 거부 등 팀 프로젝트 진행 의지가 없는 팀원은 팀에서 삭제 조치하고 프로젝트 0점 처리
  - 질병 등 불가피한 상황으로 진행 과정을 듣지 못한 경우는 제외