[2023년도 봄학기] **프로그래밍기초**

중간고사 라이브 코딩 (버전 1.2)

2023. 4. 27. 오후 7시 ~ 오후 9시 사이에 90분 @ 제1학술관 101, 102호

내려받을 파일

- `2023midTerm.pdf` : 중간고사 문제 파일 (현재 읽고 있는 파일)
- `2023MID_2023000000.py` : 중간고사 뼈대코드 파일

먼저 해야 할 작업

- 준비 과정과 테스트 제출
  - 뼈대코드 파일 `2023MID_2023000000.py` 의 파일 이름에서 `2023000000` 부분을 본인의 학번으로 바꾸세요.
  - 뼈대코드 파일을 열어서 주석에 자신의 학번과 이름을 입력합니다.
- 문제는 **크게 3번**까지 있으며 세부문제를 고려하면 총 12개 입니다. 이 문서는 **7쪽**까지 있습니다.
  - 모든 문제마다 점수가 정해져 있습니다. 라이브 코딩의 전체 점수는 **30점** 입니다.
  - (세부) 문제를 하나 풀 때마다 (문법 오류가 있는지 확인 한 후) 제출합니다. (강력 권장)
- 별도로 명시한 경우를 제외하고 부분점수는 없습니다. 즉, 일부라도 틀릴 경우 해당 문제의 점수는 0점입니다.

제약사항

- 시험범위에 나오는 다음 예약어만 사용할 수 있습니다. 이 외 다른 예약어를 사용한 문제는 0점처리 됩니다.

  `False`, `None`, `True`, `and`, `def`, `elif`, `else`, `for`, `from`, `if`, `import`, `in`, `lambda`, `not`, `or`, `pass`, `return`, `while`

- 사용가능하고 추천하는 함수 (혹은 메소드)

  `partition`, `strip`, `int`, `str` 를 포함한 교재 1~7 장에 있는 모든 함수 (혹은 메소드)

- (정수, 실수, 문자열, 논리)식과 값, 리스트, 튜플, 정수범위를 사용할 수 있습니다.

  아직 배우지 않은 집합, 딕셔너리 등은 사용할 수 없습니다. (사용시 점수를 받을 수 없습니다)

- 시험범위(1~7장)에 나오지 않는 메소드는 사용할 수 없습니다. 사용 가능 여부가 궁금하다면 시험 시간에 물어보세요.
- 각 문제마다 모의 채점 도구 함수를 제공합니다. 문제를 풀 때마다 `checker1()`, `checker2()`, `checker3()` 그리고 모든 함수를 동시에 채점하는 `checker_all()` 을 사용해보세요. 뼈대코드 맨 아래에 있는 채점코드는 수정하거나 지우면 안됩니다.
- 모의 채점 도구에서 제공하는 예시들은 실제 가능한 유형의 일부입니다. 실제 채점에는 더 많은 예시들을 사용한다는 점을 주의하기 바랍니다.

1. log_2n 구하기 [각 3점씩 총 9점]

6장의 검색 알고리즘은 순차 검색과 이분 검색의 두가지로 나눠서 배웠습니다. 검색 알고리즘에서 최악의 경우(찾으려는 키가 리스트에 없는 경우) 검색 완료시까지 비교 횟수는 아래 표와 같이 크게 차이납니다.

리스트 길이	순차 검색의 비교횟수	이분검색의 비교횟수
28	28	5
1,024	1,024	11
1,048,576	1,048,576	21
4,294,967,296	4,294,967,296	33

이분검색의 비교횟수는 대략 log_2n 에 비례하므로 log_2n 이 대략 얼마인지 구하는 함수를 작성해 봅시다.

로그 함수는 아래와 같은 특성을 가지고 있습니다.

$$log_n(n^k) = k$$

그래서 밑base이 2인 $log_2(n)$ 에서 n 이 2의 k 제곱이라면 그 결과는 아래와 같이 k 가 됩니다.

$$log_2(2^k) = k$$

그래서 $log_2(n)$ 이 얼마인지 **대략 어림잡아** 알아보는 함수는 다음과 같은 수식으로 표현할 수 있습니다.

$$log_2(n) = \begin{cases} 1 + log_2(\frac{n}{2}) & if\ n > 1 \\ 0 & if\ n = 0 \end{cases}$$

[3점] 1.1 위 수식을 보고 $log_2(n)$ 을 어림잡아 계산하는 재귀함수 `log2r` 을 완성하세요.

[3점] 1.2 작성한 재귀함수 `log2r` 을 참고해서 꼬리재귀함수 `log2t` 를 완성하세요.

[3점] 1.3 작성한 꼬리재귀함수 `log2t` 를 참고해서 while 루프 버전의 `log2w` 를 완성하세요.

* **주의 사항**

- 뼈대코드의 함수 이름을 수정하면 안됩니다.
- 완성 후 `checker1()` 함수를 실행해서 계산 값이 제대로 나오는지 확인해보세요.
- 각 함수의 리턴 값은 정수 타입이어야 합니다(2로 나눌때 결과가 실수가 되지 않도록 하세요).

2. 가위바위보 게임 처리기 [각 2점씩 총 10점]

H대학에서 프로그래밍 수업을 잘 듣고 졸업한 하냥이는 졸업 후 유명한 게임회사에 입사했습니다. 그리고 차기 기대작으로 손꼽히는 대작게임 'RSP'를 만드는 팀에 배정되었습니다. 하냥이에게 주어진 첫 임무는 'RSP' 게임에서 아주 중요한 가위바위보 게임을 처리하는 알고리즘을 작성하는 것 입니다.

* 완성 후 checker2() 함수를 실행해서 계산 값이 제대로 나오는지 확인해보세요.

[2점] 2.1 컴퓨터와 가위바위보 게임을 했을 때 누가 이겼는지 검사하는 기초 함수 `eval` 을 완성하세요. 아래 조건을 모두 만족해야 합니다.

- 2개의 인자 `you` 와 `com` 을 입력받습니다.
- `you` 와 `com` 은 항상 `"R"`, `"S"`, `"P"` 의 세가지 문자열 중 하나가 온다고 가정합니다. 세 문자열은 각각 바위, 가위, 보를 의미합니다. 바위는 가위를 이기고, 가위는 보를 이기고, 보는 바위를 이깁니다. (In English terms, it means rock, scissors, and paper. Rock beats scissors, scissors beats paper, and paper beats rock.)
- 만약 `you` 가 `com` 을 이긴 경우, 정수 `1`을 리턴합니다.
- 만약 `you` 가 `com` 에게 진 경우, 정수 `-1` 을 리턴합니다.
- 비긴 경우, 정수 `0`을 리턴합니다.

하냥이의 상사는 가위바위보 게임 기록을 문자열의 튜플의 리스트로 저장하기로 결정했습니다. 아래는 게임 기록의 여러 예시 입니다.

```
[]
[("R", "S")]
[("R", "S"), ("S", "R"), ("R", "P")]
[("P", "S"), ("P", "R"), ("P", "P"), ("S", "S")]
```

각 리스트 내 원소 튜플에 담긴 두 문자열은 각각 플레이어와 컴퓨터의 것입니다.

보스가 준 두번째 임무는 게임 기록을 입력받아서 플레이어가 몇번 이겼는지 확인하는 함수입니다. 하냥이는 머리를 짜내어 다음과 같은 알고리즘을 작성했습니다.

하냥이의 게임 기록 처리 함수 알고리즘 (ver 0.1)

입력 : 게임 기록 (두 문자열을 담은 튜플의 리스트)

출력 : 입력받은 게임 기록에서 플레이어가 이긴 횟수

* 주의사항: 각 튜플은 게임 한판을 의미하고 두 문자열 중 첫째가 플레이어의 기록임

구현 절차 :

- 게임 기록 리스트를 입력받는다.
- 만약 리스트가 비어있지 않다면 리스트의 첫 원소를 꺼낸다. 이 원소는 튜플이므로 아래와 같이 두 값을 동시에 꺼낼 수 있다.

  `플레이어가_낸_가위바위보, 컴퓨터가_낸_가위바위보 = 리스트의_첫_원소`
- 플레이어가 이겼는지를 확인한다. 이때 먼저 작성했던 `eval` 함수를 사용한다. (`eval` 함수에 두 값을 넣은 결과가 1이면 플레이어가 이긴 것이다)
- 만약 플레이어가 이겼으면 남은 리스트에서 플레이어가 이긴 횟수를 재귀적으로 구한 값에 1을 더해 리턴한다. 그렇지 않으면 남은 리스트에서 플레이어가 이긴 횟수를 재귀적으로 구한 값을 그대로 리턴한다.

[2점] 2.2 위 알고리즘 대로 재귀함수 `wins_r` 을 완성하세요.

[2점] 2.3 재귀함수 `wins_r` 을 보고 꼬리재귀함수 `wins_t` 를 완성하세요.

[2점] 2.4 꼬리재귀함수 `wins_t` 를 보고 while 루프 버전의 `wins_w` 를 완성하세요.

[2점] 2.5 `wins_w` 는 for 루프 버전으로 작성할 수 있을까요?

불가능하다고 생각한다면:

변수 wins_for_version_is_possible 의 값을 `False` 로 바꾸세요.

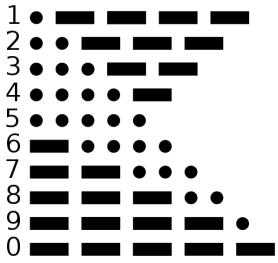
가능하다고 생각한다면:

변수 wins_for_version_is_possible 의 값을 `True` 로 바꾸고 함수 `wins_f` 를 완성하세요.

3. 모스부호 변환기 [합계 11점]

모스부호는 새뮤얼 핀리 브리즈 모스(Samuel Finley Breese Morse, 1791~1872)가 고안한 전신기호 입니다. 짧은 발신 전류와 긴 발신 전류 두가지 만으로 문장을 구성하여 전신기로 전송할 수 있게 하였으며, 현대에도 비상통신수단으로 사용합니다.

오른쪽 그림은 아라비아 숫자의 모스 부호입니다.



출처 : https://ko.wikipedia.org/wiki/모스_부호

짧은 발신 전류는 문자열 `"."`, 긴 발신 전류는 문자열 `"-"` 로 표현한다고 정했을 때, 모스 부호를 변환하는 여러가지 함수를 만들어 봅시다.

* 완성 후 checker3() 함수를 실행해서 계산 값이 제대로 나오는지 확인해보세요.

[2점] 3.1 0 에서 9 사이의 정수를 입력받아서 모스부호(문자열)를 리턴하는 함수 `digit2morse` 를 작성해 봅시다.

실행 사례:

```
>>> digit2morse(3)
"...--"
>>> digit2morse(0)
"-----"
```

[2점] 3.2 입력받은 문자열이 0부터 9 사이의 정수에 해당하는 단일 모스부호인지를 확인하는 함수 `is_digit_morse` 를 작성해봅시다.

실행 사례:

```
>>> is_digit_morse("-----")
True
>>> is_digit_morse(".--")
False
```

다음과 같이 단일 부호의 앞뒤에 공백이 들어간 경우에는 정상적인 것으로 판정하고 `True` 를 리턴해야 합니다.

실행 사례:

```
>>> is_digit_morse(" .----")
True
>>> is_digit_morse("----.    ")
True
>>> is_digit_morse("             -----    ")
True
```

하지만 아래와 같이 부호 사이에 공백이 들어가 있거나 0에서 9 사이의 모스부호 값이 아닌 경우에는 `False` 를 리턴해야 합니다.

실행 사례:

```
>>> is_digit_morse("--.  --")        # 가운데 공백이 있음
False
>>> is_digit_morse(" -    -.--   ") # 앞 뒤는 괜찮은데 가운데 공백이 있음
False
>>> is_digit_morse("-.-.-")          # 0 에서 9 사이의 값이 아님
False
```

[2+2점] 3.3 모스부호(문자열) 하나를 입력받아서 0 에서 9 사이의 정수를 리턴하는 함수 `morse2digit` 를 작성해봅시다.

실행 사례:

```
>>> morse2digit("-----")
0
>>> morse2digit("...--")
3
```

만약 입력받은 문자열이 0 에서 9사이의 정수에 해당하는 모스부호가 아닐 경우에는 `None` 을 리턴해야 합니다. 이 조건을 검사하기 위해 3.2 에서 작성한 함수 `is_digit_morse` 를 사용해도 좋습니다.

실행 사례:

```
>>> morse2digit("...-")
None
>>> morse2digit("  ----- ")
0
>>> morse2digit("..      ..-")
None
```

문자열의 길이가 정확히 5인 정상적인 입력에 대해서만 잘 동작할 경우에는 2점만 획득할 수 있습니다. 비정상적인 경우나 앞 뒤에 공백이 들어간 경우까지 모두 제대로 처리할 경우에만 추가로 2점을 획득할 수 있습니다.

[3점] 3.4 모스부호(문자열)를 입력받아서 정수를 리턴하는 함수 `morse2number` 를 만들어 봅시다. 모스부호 사이에는 구분을 위해 공백이 들어가야 하므로 부호 사이사이마다 공백문자열을 최소 하나 이상 입력해야 합니다.

실행 사례:

```
>>> morse2number("-----")
0
>>> morse2number("...-- -----")
30
>>> morse2number("..--- -----    ..---  ...--")
2023
>>> morse2number("----.                    ----.")
99
```

만약 중간에 하나라도 잘못된 모스부호가 있거나 입력이 빈 문자열인 경우 -1 을 리턴해야 합니다.

실행 사례:

```
>>> morse2number("")
-1
>>> morse2number("...-- - ----")
-1
```