> Das Wesen der Mathematik liegt in ihrer Freiheit(수학의 본질은 그 자유로움에 있다)
>
> *Georg Ferdinand Ludwig Philipp Cantor*

1. 사용자에게 자신의 이름을 입력하도록 요청하는 함수를 작성하세요. 이 프로그램은 사용자의 이름을 사용하여 사용자에게 인사하는 메시지로 응답해야 합니다[0.1].

2. 방의 너비와 길이를 활용하여 넓이를 계산하는 함수를 작성하세요. 해당 함수는 방의 면적을 반환해야 합니다. 길이와 너비는 부동 소수점 숫자입니다[0.1].

3. 미국의 사용자가 식당에서 주문한 식사 비용을 입력하면, 식사에 대한 세금과 팁을 계산하는 함수를 작성하세요. 납부해야 할 세액을 계산할 때 현지 세율을 사용합니다. 팁은 식사 금액의 18%(세금 제외)로 계산합니다. 함수의 반환값은 세금 금액, 팁 금액, 세금과 팁을 모두 포함한 식사 총합계가 포함되어야 합니다. 모든 값이 소수점 이하 두 자리까지 표시되도록 출력 형식을 지정합니다[0.3].

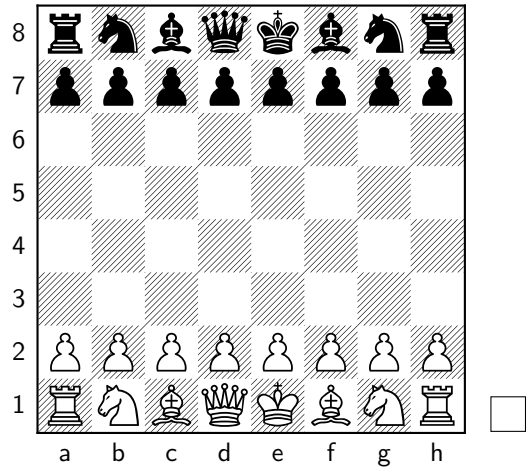
4. 사용자로부터 양의 정수 n을 읽은 다음 1부터 n까지의 모든 정수의 합을 반환하는 함수를 작성하세요. n개의 양의 정수의 합은 공식($\text{sum} = \frac{n(n+1)}{2}$)을 사용하여 계산할 수 있습니다[0.3].

5. 한 빵집에서 빵을 개당 3.49달러에 판매합니다. 하루 지난 빵은 60% 할인됩니다. 사용자로부터 구매한 하루 된 빵의 개수를 인자로 입력받으세요. 그런 다음 빵의 정상 가격, 하루 지난 빵이기 때문에 할인된 금액, 총 가격을 반환해야 합니다[0.5].

6. 프로그램에 정수가 짝수인지 홀수인지를 반환하는 함수를 작성하세요[0.5].

7. 이차 함수는 $f(x) = ax^2 + bx + c$의 형태를 가지며, 여기서 a, b, c는 상수이고 a는 0이 아닙니다. 이차 방정식 $ax^2 + bx + c$을 만족하는 x의 값을 찾으면 그 근을 확인할 수 있습니다. 이러한 값은 이차 방정식의 해를 구하는 공식을 사용하여 계산할 수 있습니다. 이차 함수는 0, 1 또는 2개의 실근을 가질 수 있습니다. $root = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ 식의 제곱근 기호 아래 부분을 판별식이라고 합니다. 판별식이 음수이면 이차 방정식에는 실근이 없습니다. 판별식이 0이면 방정식에는 실근이 하나 있습니다. 그렇지 않으면 방정식은 두 개의 실근을 가집니다. a,b,c의 값을 함수의 인자로 전달합니다. 실근의 개수를 나타내는 메시지와 함께 실근의 값(있는 경우)을 표시해야 합니다[1.5].

8. 체스판의 위치는 문자와 숫자로 식별됩니다. 문자는 열을, 숫자는 행을 식별합니다.



함수의 인자 값으로 체스판의 위치를 입력받아 해당 위치에 있는 사각형의 색상을 결정하는 함수를 작성하세요. 예를 들어 사용자가 a1을 입력하면 프로그램은 사각형이 검은색이라고 반환해야 합니다. 사용자가 d5를 입력하면 프로그램은 사각형이 흰색이라고 반환해야 합니다. 프로그램은 항상 유효한 위치가 입력되었다고 가정합니다[1.0].

9. 다음 공식(Tomohiko Sakamoto's Algorithm)을 사용하여 특정 연도 1월 1일의 요일을 결정할 수 있습니다.($year + (year - 1)//4 - (year - 1)//100 + (year - 1)//400)\%7$ 이 공식으로 계산된 결과는 요일을 나타내는 정수입니다. 일요일은 0

으로 표시되며, 나머지 요일은 순차적으로 증가하여 토요일을 6으로 표시합니다. 위의 공식을 사용하여 사용자로부터 연도를 읽고 해당 연도 1월 1일의 요일을 반환하는 함수를 작성합니다. 반환값에는 요일의 전체 이름이 포함되어야 합니다[1.5].

10. 룰렛 휠에는 38개의 칸이 있습니다. 이 중 18칸은 검은색, 18칸은 빨간색, 2칸은 녹색입니다. 녹색 칸의 번호는 0과 00입니다. 빨간색 칸은 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 29 31, 33, 35으로 번호가 매겨져 있습니다. 1에서 36 사이의 나머지 정수는 검은색 칸에 번호를 매기는 데 사용됩니다. 룰렛에는 다양한 베팅을 할 수 있지만, 아래 룰만 고려하겠습니다:

- 단일 번호 (1 36, 0, 00)
- Red vs. Black
- Odd vs Even (0과 00은 짝수가 아님)
- 1 18 vs 19 36

Python의 난수 생성기를 사용하여 룰렛 휠의 회전을 시뮬레이션하는 함수를 작성합니다. 선택한 숫자와 지불해야 하는 모든 베팅을 표시합니다. 예를 들어 13이 선택되면 함수는 아래와 같은 내용을 출력해야 합니다.

- 난수 결과 13
  - 13에 지불
  - 검은색 지불
  - 홀수 지불
  - 1에서 18까지 지불

만약, 시뮬레이션 결과가 0 또는 00이면 프로그램에 추가 출력 없이 0 또는 00이 표시됩니다[3.5].

11. 정수 집합에서 최대값을 찾는 함수를 살펴봅니다. 각 정수는 1에서 100 사이의 숫자 중에서 무작위로 선택됩니다. 정수 집합에 중복된 값이 포함될 수 있으며, 1에서 100 사이의 정수 중 일부는 존재하지 않을 수도 있습니다.

많은 사람이 각 정수를 차례로 확인하고 현재 고려 중인 숫자가 이전에 본 가장 큰 숫자보다 큰지 스스로에게 물어볼 것입니다. 만약 그렇다면 이전 최대 숫자는 잊어버리고 현재 숫자를 새로운 최대 숫자로 기억합니다. 이는 합리적인 접근 방식이며, 이 과정을 주의 깊게 수행하면 정답을 얻을 수 있습니다. 이 작업을 수행하는 경우 최대값을 업데이트하고 새 숫자를 기억하는 데 몇 번이나 필요할 것으로 예상하시나요?
확률 이론을 사용하여 답할 수 있지만, 상황을 관찰할 수 있는 시뮬레이션을 작성하기로 합니다. 1에서 100 사이의 임의의 정수를 선택하는 것으로 시작하는 함수를 작성합니다. 이 정수를 지금까지 발생한 최대값으로 저장합니다. 초기 정수가 선택된 후 1에서 100 사이의 무작위 정수를 99개 더 생성합니다. 생성되는 각 정수를 확인하여 지금까지 발생한 최대 수보다 큰지 확인합니다. 이 함수는 최대 발생 수를 업데이트하고 업데이트를 수행한 사실을 계산해야 합니다. 정수를 생성한 후 각 정수를 표시합니다. 새로운 최대값을 나타내는 정수에 표기법을 포함시킵니다. 정수를 100개 표시한 후 함수는 최대값이 업데이트된 횟수와 함께 발견된 최대값을 표시해야 합니다[3.5].

```
30
74 <== Update
58
17
40
37
13
34
46
52
80 <== Update
37
97 <== Update
45
55
73
...
```

반환값은 (100, 5) 입니다.

12. 이 함수는 1의 곱하기 1부터 10의 곱하기 10까지 모든 정수 조합의 곱을 표시하는 구구단을

표시하는 함수를 작성하세요. 구구단 표의 상단에는 숫자 1부터 10까지가 포함된 레이블 행이 포함되어야 합니다. 왼쪽 아래에는 숫자 1부터 10까지로 구성된 레이블이 포함되어야 합니다. 프로그램의 예상 출력은 아래와 같습니다. 콘솔이나 노트북에 표시되는 형태도 아래와 같아야 합니다(단순 구구단을 문제로 출제하지 않습니다)[1.5].

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Table 1: 구구단 표, 1에서 10까지의 곱셈, 테이블 형태로 출력

13. π의 값은 다음과 같은 무한급수로 근사화할 수 있습니다.

$$pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + ...)$$

π의 근사값 55개를 표시하는 함수를 작성합니다. 첫 번째 근사는 무한 급수의 첫 번째 항만을 사용해야 합니다. 프로그램에 의해 표시되는 각 추가 근사는 급수의 항을 하나 더 포함해야 하며, 이전에 표시된 근사보다 더 나은 의 근사가 되어야 합니다[5.0].

14. 패리티 비트는 전화선과 같이 신뢰할 수 없는 연결을 통해 전송되는 데이터의 오류를 감지하기 위한 간단한 메커니즘입니다. 기본 개념은 전송의 단일 비트 오류를 감지할 수 있도록 각 8비트 그룹 뒤에 추가 비트를 전송하는 것입니다. 패리티 비트는 짝수 패리티 또는 홀수 패리티로 계산할 수 있습니다. 짝수 패리티를 선택하면 전송되는 패리티 비트는 전송되는 1비트(데이터 8비트와 패리티 비트)의 총 개수가 짝수가 되도록 선택됩니다. 홀수 패리티를 선택하면 전송되는 패리티 비트는 전송되는 1비트의 총 개수가 홀수가 되도록 선택됩니다. 짝수 패리티를 사용하여 사용자가 입력한 8비트 그룹에 대한 패리티 비트를 계산하는 함수를 작성하세요. 프로그램은 사용자가 빈 줄을 입력할 때까지 8비트가 포함된 문자열을 읽어야 합니다. 사용자가 각 문자열을 입력할 때마다 프로그램은 패리티 비트가 0인지 1인지를 나타내는 명확한 메시지를 표시해야 합니다. 사용자가 8비트 이외의 비트를 입력하면 적절한 오류 메시지를 표시합니다[2.5].

15. 동전을 세 번 연속으로 던져 같은 결과(세 개 모두 앞면 또는 세 개 모두 뒷면)가 나오려면 최소 몇 번을 던져야 하나요? 필요한 최대 동전 던지기 횟수는 몇 번인가요? 평균적으로 몇 번의 뒤집기가 필요한가요?
동전 던지기를 시뮬레이션하는 함수를 만들어 관찰을 해보겠습니다. Python의 난수 생성기를 사용하여 동전 던지기를 여러 번 시뮬레이션하는 함수를 작성합니다. 시뮬레이션된 동전은 앞면이 나올 확률과 꼬리가 나올 확률이 같아야 하는 공정한 동전이어야 합니다. 프로그램은 3번 연속 앞면 또는 3번 연속 꼬리가 나올 때까지 시뮬레이션된 동전을 뒤집어야 합니다. 결과가 앞면이 나올 때마다 H를, 뒷면이 나올 때마다 T를 표시하고 한 시뮬레이션의 모든 결과를 같은 줄에 표시합니다. 그런 다음 동일한 결과가 3회 연속으로 발생하는 데 필요한 뒤집기 횟수를 표시합니다. 프로그램이 실행되면 시뮬레이션을 10회 수행하고 필요한 평균 뒤집기 횟수를 보고해야 합니다[6.0].

샘플 출력은 아래와 같습니다.

```
H T T T (4 flips)
H H ... T (19 flips)
...
T H T T T (5 flips)
```

평균적으로 7.9번의 flips이 필요했습니다.

16. 하위 목록은 더 큰 목록의 일부인 목록입니다. 하위 목록은 단일 요소, 여러 요소 또는 요소가 전혀 포함되지 않은 목록일 수 있습니다. 예를 들어 [1], [2], [3], [4]는 모두 [1, 2, 3, 4]의 하위

목록입니다. 목록 [2, 3]도 [1, 2, 3, 4]의 하위 목록이지만 [2, 4]는 더 긴 목록에서 요소 2와 4가 인접하지 않으므로 [1, 2, 3, 4]의 하위 목록이 아닙니다. 빈 목록은 모든 목록의 하위 목록입니다. 따라서 []는 [1, 2, 3, 4]의 하위 목록입니다. 목록은 그 자체의 하위 목록이므로 [1, 2, 3, 4]도 [1, 2, 3, 4]의 하위 목록입니다. 이 연습에서는 한 목록이 다른 목록의 하위 목록인지 여부를 결정하는 함수인 isSublist를 만들 것입니다. 함수의 유일한 매개변수로 큰 목록과 작은 목록 두 개를 사용해야 합니다. 이 함수는 작은 목록이 큰 목록의 하위 목록인 경우에만 True를 반환해야 합니다[2.5].

17. 과학 실험의 일부로 수집된 데이터를 분석할 때 다른 계산을 수행하기 전에 가장 극단적인 값을 제거하는 것이 바람직할 수 있습니다. 값의 목록과 음수가 아닌 정수 n을 매개변수로 받는 함수를 작성합니다. 이 함수는 가장 큰 요소 n 개와 가장 작은 요소 n 개를 제거한 목록의 새 복사본을 만들어야 합니다. 그런 다음 함수의 유일한 결과로 목록의 새 복사본을 반환해야 합니다. 반환된 목록의 요소 순서가 원본 목록의 요소 순서와 일치할 필요는 없습니다. 이 프로그램은 사용자로부터 숫자 목록을 읽고 앞에서 설명한 함수를 호출하여 가장 큰 값 두 개와 가장 작은 값 두 개를 제거해야 합니다. 이상값이 제거된 목록을 표시한 다음 원래 목록을 표시합니다. 사용자가 4개 미만의 값을 입력하면 프로그램은 적절한 오류 메시지를 생성해야 합니다[2.5].

18. 최적의 직선은 n개의 데이터 포인트 모음에 가장 근접한 직선을 말합니다. 각 점에 x 좌표와 y 좌표가 있다고 가정합니다. 기호 $\bar{x}$와 $\bar{y}$는 각각 컬렉션의 평균 x 값과 컬렉션의 평균 y 값을 나타내는 데 사용됩니다. 최적 선은 $y = mx + b$ 방정식으로 표현되며, 여기서 m과 b는 다음 공식을 사용하여 계산됩니다.

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$
$$b = \bar{y} - m\bar{x}$$

사용자로부터 점 모음을 읽는 프로그램을 작성합니다. 사용자는 첫 번째 x 좌표를 입력하고 그 다음 줄에 첫 번째 y 좌표를 입력합니다. 프로그램이 x 좌표에 대한 빈 줄을 읽을 때까지 사용자가 각자의 줄에 x 및 y 값을 입력한 상태로 좌표를 계속 입력하도록 허용합니다. 앞의 수식으로 계산한 값으로 m과 b를 대체하여 $y = mx + b$ 형식으로 가장 잘 맞는 선에 대한 수식을 표시합니다. 예를 들어 사용자가 좌표 $(1, 1)$, $(2, 2.1)$ 및 $(3, 2.9)$를 입력하면 프로그램에 $y = 0.95x + 0.1$이 표시되어야 합니다[4.5].

19. 토큰화는 문자열을 토큰이라고 하는 하위 문자열 목록으로 변환하는 프로세스입니다. 대부분의 경우 원래 문자열은 간격이 불규칙할 수 있기 때문에 토큰 목록이 원래 문자열보다 작업하기가 훨씬 쉽습니다. 어떤 경우에는 한 토큰이 끝나는 위치와 다음 토큰이 시작되는 위치를 파악하기 위해 상당한 작업이 필요하기도 합니다. 수학 표현식에서 토큰은 연산자, 숫자, 괄호와 같은 항목입니다. 이 문제에서 고려할 연산자 기호는 $*$, $/$, $\hat{}$, $-$ 및 $+$입니다. 연산자와 괄호는 토큰이 항상 단일 문자이고 해당 문자가 다른 토큰의 일부가 아니기 때문에 쉽게 식별할 수 있습니다. 숫자는 토큰이 여러 문자로 구성될 수 있기 때문에 식별하기가 조금 더 어렵습니다. 연속된 숫자 시퀀스는 모두 하나의 숫자 토큰으로 취급해야 합니다.
수학식이 포함된 문자열을 유일한 매개변수로 받아 이를 토큰 목록으로 나누는 함수를 작성합니다. 각 토큰은 괄호, 연산자 또는 숫자여야 합니다. (간단하게 하기 위해 이 문제에서는 정수로만 작업하겠습니다.). 함수의 유일한 결과로 토큰 목록을 반환합니다.
함수에 전달된 문자열에는 항상 괄호, 연산자 및 정수로 구성된 유효한 수학식이 포함되어 있다고 가정할 수 있습니다. 그러나 함수는 이러한 요소 사이의 다양한 공백(공백이 없는 경우 포함)을 처리해야 합니다. 사용자로부터 표현식을 읽고 토큰 목록을 인쇄하여 토큰화 함수를 시연하는 메인 프로그램을 포함하세요[6.0].

20. 빙고 카드는 5개의 숫자로 이루어진 5개의 열로 구성되어 있습니다. 열에는 B, I, N, G, O라는 글자가 표시되어 있으며, 각 글자 아래에 나타날

수 있는 숫자는 15개입니다. 특히, B 아래에 나타날 수 있는 숫자는 1에서 15까지, I 아래에 나타날 수 있는 숫자는 16에서 30까지, N 아래에 나타날 수 있는 숫자는 31에서 45까지입니다.
(1) 무작위 빙고 카드를 생성하여 사전에 저장하는 첫번째 함수를 작성합니다. (2) 키를 문자 B, I, N, G, O이고 값은 다섯 개의 숫자 목록입니다. 열에 적절한 레이블이 붙은 빙고 카드를 표시하는 함수를 작성합니다.
이 함수를 사용하여 무작위 빙고 카드를 표시하는 프로그램을 작성합니다.

21. 당첨된 빙고 카드에는 모두 호출된 5개의 숫자 줄이 있습니다. 플레이어는 일반적으로 호출된 숫자를 지우거나 마커로 표시하여 기록합니다.
빙고 카드 사전에서 숫자를 0으로 대체하여 숫자가 호출되었음을 표시합니다. 빙고 카드를 나타내는 사전을 유일한 매개변수로 사용하는 함수를 작성합니다. 카드에 다섯 줄의 0(세로, 가로 또는 대각선)이 포함되어 있으면 함수는 True를 반환하여 카드가 당첨되었음을 나타내야 합니다. 그렇지 않으면 함수는 False를 반환해야 합니다. 가로선이 있는 카드 하나 이상, 세로선이 있는 카드 하나 이상, 대각선이 있는 카드 하나 이상, 일부 숫자가 지워져 있지만 당첨선이 포함되지 않은 카드 하나 이상을 사용하여 함수를 확인하세요[1.5].

22. 이 연습에서는 하나의 카드에 대해 빙고 게임을 시뮬레이션하는 프로그램을 작성합니다. 먼저 모든 유효한 빙고 호출(B1 O75) 목록을 생성하는 것으로 시작합니다. 목록이 생성되면 무작위 모듈에서 셔플 함수를 호출하여 요소의 순서를 무작위로 지정할 수 있습니다. 그런 다음 프로그램은 목록에서 호출을 소비하고 카드에 당첨 라인이 포함될 때까지 카드의 숫자를 지워야 합니다. 1,000개의 게임을 시뮬레이션하고 카드가 승리하기까지 필요한 최소, 최대 및 평균 통화 수를 보고합니다.