

1. プログラミング基礎

1. (基本文法の確認) 以下のプログラムを作成せよ。

a) (変数の定義) 始点、終点、そして到達するまでの経過時間を元に速度を計算するプログラムを作れ。

1 PT

```
1 def speed(start: int, end: int, time_elapsed: int) -> int:
2     # TODO: 以下に`distance` 変数を定義して正しい値を計算せよ。
3
4
5     # 以下の行は変更しないこと
6     return distance // time_elapsed
7
8 assert speed(10, 30, 10) == 2
9 assert speed(10, 30, 2) == 15
10 assert speed(10, 31, 10) == 2
```

以降のプログラム作成問題では、特に指示がない限り `assert` でプログラムの動作チェックを行うことをおすすめする。

b) (分岐) 奇数を判定するプログラムを書け。

1 PT

```
1 def is_odd(n: int) -> bool:
2     pass
3
4 assert is_odd(1) == True
5 assert is_odd(2) == False
6 assert is_odd(101) == True
```

c) (分岐) 夏日は最高気温が 25℃以上の日のことである。その日の最高気温から夏日か判定するプログラムを書け。

1 PT

```
1 def is_summerday(high_temperature: int) -> bool:
2     pass
3
4 assert is_summerday(25) == True
5 assert is_summerday(24) == False
```

d) (分岐) 気象庁では温度ごとに天候が以下のように定義されている。

2 PTs

- 最低気温が 0℃未満: 冬日
- 最高気温が 0℃未満: 真冬日
- 最高気温が 25℃以上: 夏日
- 最高気温が 30℃以上: 真夏日
- 最高気温が 35℃以上: 猛暑日

最高気温と最低気温に基づいて、各天候を文字列として返す関数を書け。条件に当てはまらない場合の呼称を考慮してみよ。

```
1 def weather(high_temperature: int, low_temperature) -> str:
2     pass
```

e) (エラー処理) 速度を計算するプログラムについて、経過時間が0の場合エラー処理を行え。必要に応じてエラー処理、ゼロ除算について調べよ。

2 PTs

```
1 def speed(start: int, end: int, time_elapsed: int) -> int:
2     # TODO
3     pass
4
5 assert speed(10, 30, 0) == "Zero division error"
6 assert speed(10, 30, 10) == 2
7 assert speed(10, 30, 2) == 15
8 assert speed(10, 31, 10) == 2
```

f) (再帰) (基本情報技術者試験 サンプル問題 改題) 非負整数 `n` を引数に取り、階乗を計算する関数 `fact` の正しいプログラムを作れ。実装には再帰を使え。

2 PTs

```
1 def fact(n: int) -> int:
2     pass
3
4 assert fact(0) == 1
5 assert fact(1) == 1
6 assert fact(2) == 2
7 assert fact(5) == 120
```

g) (while) 非負整数 `n` を引数に取り、階乗を計算する関数 `fact` の正しいプログラムを作れ。実装には `while` を使え。

1 PT

```
1 def fact(n: int) -> int:
2     pass
3
4 assert fact(0) == 1
5 assert fact(1) == 1
6 assert fact(2) == 2
7 assert fact(5) == 120
```

h) (for) 非負整数 `n` を引数に取り、階乗を計算する関数 `fact` の正しいプログラムを作れ。実装には `for` を使え。

1 PT

```
1 def fact(n: int) -> int:
2     pass
3
4 assert fact(0) == 1
5 assert fact(1) == 1
6 assert fact(2) == 2
7 assert fact(5) == 120
```

2. (変数と関数) BMI (Body Mass Index) は体重と身長から算出される、肥満度を表す体格指標である。計算式は以下の通りである。

$$\text{BMI} = \frac{\text{体重 (kg)}}{\text{身長 (m)}^2}$$

a) 身長と体重を表す変数を各々用意して、BMI を計算してみよ。

1 PT

- b) BMI を計算する関数を設計せよ。 1 PT
- c) 関数が正しく計算されていることを確認するための `assert` 文を書け。 1 PT
- d) Wikipedia の BMI の項を参考に、体重と身長を入力値として、状態を出力するプログラムを作成せよ。 2 PTs
- e) BMI 計算の入力値について何か注意する点はあるか。気付いた点について意見を述べよ。 2 PTs

3. (プログラムの読解) 以下にプログラムを挙げる。間に回答せよ。

a) 以下は `Hello, World` を出力することを意図した1行のプログラムである。バグを指摘し修正せよ。 1 PT

```
1 print(Hello, World)
```

b) (基本情報技術者試験 R5 類題) 以下のプログラムについて、`proc2` を呼び出した際の動作を説明せよ。 1 PT

```
1 def proc1():
2     print("A")
3
4 def proc2():
5     proc3()
6     print("B")
7     proc1()
8
9 def proc3():
10    print("C")
11    proc1()
```

c) (基本情報技術者試験 サンプル問題) 以下のプログラムの動作を説明せよ。 1 PT

```
1 x = 1
2 y = 2
3 z = 3
4
5 x = y
6 y = z
7 z = x
8 print(y, z)
```

d) (基本情報技術者試験 サンプル問題 改題) 関数 `calc` は x, y を受け取り $\sqrt{x^2 + y^2}$ を計算する。 `**` 演算子のみを使ってこの関数を実装せよ。 1 PT

```
1 def calc(x: float, y: float):
2     pass
```

e) (基本情報技術者試験 サンプル問題) 以下の関数 `makeNewArray` を `makeNewArray([3, 2, 1, 6, 5, 4])` と呼び出した時、返り値のリストの添字 4 番目の値は何になるか。 2 PTs

```
1 def makeNewArray(input: list) -> list:
2     out = [input[0]]
3     for i in range(1, len(input)):
4         tail = out[len(out)-1]
5         out.append(tail + input[i])
6     return out
```

4. (数値演算と条件分岐 基本) 以下のプログラムを作成せよ。

a) `input()` 関数を使い、入力した文字列をそのまま表示するプログラムを作成せよ。 1 PT

b) `input()` 関数から 2 つの数字を受け取り、大きい数を表示するプログラムを作成せよ。 1 PT

c) `input()` 関数から 1 つの整数を受け取り、絶対値を表示するプログラムを作成せよ。 1 PT

d) アクションゲームの判定部を作る。操作キャラクターがスター状態かつ敵と体当たりしている場合、敵を倒すことができる。「スター状態であるか」と「体当たり状態か」を真偽値として入力し、「敵を倒すことができるか」を真偽値として返却する関数を作れ。テストに必要な `assert` 文が足りていないため追加せよ。 2 PTs

```
1 def beatable(is_star: bool, is_bumped: bool) -> bool:
2     pass
3
4     assert beatable(True, True) == True
5     assert beatable(False, True) == False
```

e) アクションゲームのミス判定部を作る。操作キャラクターが「パワーアップ状態か」と「敵と接触しているか」を受け取り、操作キャラクターがやられたかどうか(ミス)を判定する。具体的には、パワーアップ状態でないのに敵と接触した場合ミスとする。このような判定関数を作れ。 2 PTs

```
1 def has_beaten(is_powerup: bool, is_bumped: bool) -> bool:
2     pass
```

f) 割り算を実行し、商 (quotient) と余り (remainder) のペアを返す関数を実装せよ。 1 PT

```
1 def quot_rem(x: int, y: int) -> (int, int):
2     pass
3
4     assert quot_rem(4, 3) == (1, 1)
5     assert quot_rem(8, 3) == (2, 2)
6     assert quot_rem(10, 5) == (2, 0)
```

g) 分を受け取り、時間と分のタプル値 (hour, minutes) を返却するプログラムを作成せよ。 2 PTs

```
1 def minute2hours(m: int) -> (int, int):
2     pass
3
4     assert minute2hours(30) == (0, 30) # 30 分は 0 時間 30 分
5     assert minute2hours(150) == (2, 30) # 150 分は 2 時間 30 分
```

h) 通貨を両替するプログラムを書け。両替する額 (budget) と両替レート (exchange_rate) が与えられる。両替レートは外貨 1 単位に対して必要な自国通貨を表す。

1 PT

```
1 def exchange_money(budget: float, exchange_rate: float) -> float:
2     pass
3
4 assert exchange_money(127.5, 1.2) == 106.25
```

i) 両替金額と紙幣の単位を入力し、両替するのに紙幣が何枚必要になるかを計算するプログラムを作れ。例えば 10,500 円を 1,000 円で両替する場合、お札は 10 枚必要になる (端数は切り捨てる)。

1 PT

```
1 def number_of_bills(amount: float, denomination: int) -> int:
2     pass
3
4 assert number_of_bills(10500, 1000) == 10
```

j) (Excercise より) 通貨を両替するプログラムを書け。両替する額 (budget) と両替レート (exchange_rate)、手数料率 (spread) と両替単位 (denomination) が与えられる。手数料率は 整数値 % として与えられる。そのため、例えば両替レート 1.10, 手数料率 10 % の場合、実際の両替レートは 1.20 となる。両替単位は両替を行う紙幣単位を表す。両替は指定された紙幣単位未満で行うことができるため、端数は切り捨てる。

2 PTs

```
1 def exchangeable_value(budget: float, exchange_rate: float, spread: int,
2     denomination: int) -> int:
3     pass
4
5 assert exchangeable_value(127.25, 1.20, 10, 20) == 80
6 assert exchangeable_value(127.25, 1.20, 10, 5) == 95
```

k) ある年齢が、日本の法律に基づき成年か未成年か判定するプログラムを作成せよ。

1 PT

```
1 def is_adult(age: int) -> bool:
2     pass
3
4 assert is_adult(17) == False
5 assert is_adult(18) == True
```

l) 現在の年齢と年を与えて、その人物が令和、平成、昭和、大正、明治のいずれの生まれであるかを判定するプログラムを設計せよ。

2 PTs

m) ある映画館の料金は以下のようにになっている。

2 PTs

- 一般料金 2,000 円
- 6 歳未満 無料
- 6 歳以上 18 歳未満 1,000 円

- 18 歳以上 22 歳未満 1,500 円
- シニア（60 歳以上）1,400 円

入力を年齢として、映画料金を算定するプログラムを書け。

```
1 def admission_fee(age: int) -> int:  
2     pass
```

5. (数値演算 演習) 以下のプログラムを作成せよ。

a) (フィボナッチ数) 与えられた整数 `n` について、`n` 番目のフィボナッチ数を計算するプログラムを書け。 3 PTs

```
1 def fib(n: int) -> int:
2     pass
3
4 assert fib(1) == 1
5 assert fib(2) == 1
6 assert fib(3) == 2
7 assert fib(4) == 3
8 assert fib(5) == 5
9 assert fib(6) == 8
10 assert fib(7) == 13
```

b) (ユークリッドの互除法) 2つの正整数を受け取り、最大公約数を表示するプログラムを作成せよ。不明な場合はユークリッドの互除法を調べよ。 3 PTs

```
1 def gcd(a: int, b: int) -> int
2     pass
3
4 assert gcd(10, 8) == 4
5 assert gcd(31, 7) == 1
6 assert gcd(100, 50) == 50
```

c) (エラトステネスの篩) 与えられた正整数について素数かどうか判定するプログラムを書け。不明な場合はエラトステネスの篩を調べよ。 4 PTs

```
1 def is_prime(n: int) -> bool
2     pass
3
4 assert is_prime(1) == False
5 assert is_prime(2) == True
6 assert is_prime(3) == True
7 assert is_prime(4) == False
8 assert is_prime(11) == True
9 assert is_prime(105) == False
10 assert is_prime(109) == True
```

d) うるう年を判定するプログラムを書け。ここでうるう年は以下の条件に当てはまる年である。 2 PTs

- 4 の倍数かつ 100 の倍数ではない
- 400 の倍数である

e) (FizzBuzz) 1 から 100 までの数を表示せよ。ただし数値が 3 の倍数の時は `Fizz`、5 の倍数の時は `Buzz`、15 の倍数であれば `FizzBuzz` と表示せよ。 2 PTs

この問題を解かせる理由については、以下のジェフ・アトウッドの有名なエッセイを参照せよ。

- 『どうしてプログラマに・・・プログラムが書けないのか?』

・ http://www.aoky.net/articles/jeff_atwood/why_cant_programmers_program.htm

f) (ピタゴラス数) 100 以下のピタゴラス数を列挙せよ。ここでピタゴラス数とは $a^2 + b^2 = c^2$ となるような数の組 (a, b, c) である。 3 PTs

g) 二次方程式 $ax^2 + bx + c = 0$ の解を求めるプログラムを書け。解がない場合または虚数解である場合には適切なメッセージを表示するようにせよ。 4 PTs

h) (コラッツの問題) 任意の正整数 n について、以下の操作を繰り返す。 2 PTs

- ・ n が偶数の場合、 n を 2 で割る
- ・ n が奇数の場合、 $3n + 1$ とする

「有限回の操作で n は必ず 1 に到達する」という主張がコラッツ予想と呼ばれる。
 $n = 27$ について操作により数がどのように変化するか確認せよ。また操作回数は何回になるか。

i) 前問について、 $2 \leq n \leq 1000$ の範囲で操作回数が最大になる数はいくらか。また操作中の数の最大値はいくらになるか。 4 PTs

j) (アームストロング数) n 桁の正整数 N について、各桁を n 乗した総和が元の数に等しいとき N をアームストロング数と呼ぶ。例: $372 = 3^3 + 7^3 + 2^3$ 。 4 PTs

- ・ 1 つの数を受け取り、アームストロング数であるか判定する関数を作れ。
- ・ $1 \leq N \leq 1000000$ のアームストロング数をすべて求めよ。

6. (文字列) 以下のプログラムを作成せよ。

a) 与えられた文字列の数をカウントする関数を定義せよ。ただし大文字・小文字を区別せずカウントすること。 2 PTs

```
1 def count(s: str, c: str) -> int:
2     pass
3
4 assert count("AaAa", "a") == 4
```

b) 「よいしれうらなうなかよいあのこ」 を逆順にして文字列として結合せよ。 1 PT

c) スライス記法を使って「パタトクカシー」の先頭文字から1文字おきに文字列を取り出し、文字列として結合するプログラムを完成させよ。 1 PT

d) (たぬき暗号) たぬきの絵と一緒に どたたうくしつのたなかた という暗号が見つかった。このような暗号を解く関数 solve_tanuki を作れ。 2 PTs

```
1 def solve_tanuki(s: str):
2     pass
```

e) "Hello" と "world" を比較した時、どちらが大きな文字と判定されるか説明せよ。 1 PT

f) ある英文字列を大文字化 & 母音を削除するようなプログラムを書け。 2 PTs

```
1 def upper_remove_vowels(s: str) -> str:
2     pass
3
4 assert upper_remove_vowels("FM Yokohama") == "FM YKHM"
```

g) ある文字列が回文か判定するプログラムを書け。 3 PTs

```
1 def is_palindrome(s: str) -> bool:
2     pass
3
4 assert is_palindrome("madamimadam") == True
5 assert is_palindrome("madammadam") == True
6 assert is_palindrome("abcd") == False
7 assert is_palindrome("") == True
8 assert is_palindrome("a") == True
```

h) ある単語がイソグラム (isogram) か判定するプログラムを書け。イソグラムとは、単語中の各アルファベットが1回だけ出現する単語を指す。 3 PTs

```
1 def isogram(s: str) -> bool:
2     pass
3
4 assert is_isogram("isogram") == True
5 assert is_isogram("computer") == True
6 assert is_isogram("algorithm") == True
```

```
7 assert is_isogram("six-years-old") == True
8 assert is_isogram("aadvark") == False
```

i) ある単語がパングラム (pangram) か判定するプログラムを書け。パングラムとは、すべてのアルファベットが使われている文字である（アルファベットの使用回数は問わない）。

4 PTs

```
1 def is_pangram(s: str) -> bool:
2     pass
3
4 assert is_pangram("The quick brown fox jumps over the lazy dog.") ==
    True
5 assert is_pangram("Jackdaws love my big sphinx of quartz.") == True
```

j) (ISBN 検証: IT パスポート H24 春期 中間 D に類題あり) 書籍の識別番号である ISBN-10 についてその正当性を検証するプログラムを書け。

5 PTs

説明: ISBN-10 はハイフンで区切られた 9 桁の数字と 1 桁のチェック数字から構成される。チェック数字は 0~9 の数字または x の場合があり、x は 10 を表す。ハイフンはある場合もあるし、ない場合もある。

チェックディジットの正しさは以下の式で検証される (mod は左の式を右で割ったときの余りとなる)。

$$(d_1 * 10 + d_2 * 9 + d_3 * 8 + d_4 * 7 + d_5 * 6 + d_6 * 5 + d_7 * 4 + d_8 * 3 + d_9 * 2 + d_{10} * 1) \bmod 11 == 0$$

結果が真であれば正しい ISBN-10 コードであり、そうでなければ無効な ISBN である。

例として 3-598-21508-8 を取り上げる。これは以下の通り計算される。

$$(3 * 10 + 5 * 9 + 9 * 8 + 8 * 7 + 2 * 6 + 1 * 5 + 5 * 4 + 0 * 3 + 8 * 2 + 8 * 1) \bmod 11 == 0$$

結果が真であるためこれは正しい ISBN-10 コードである。

```
1 def is_isbn10(s: str) -> bool:
2     pass
3
4 assert is_isbn10("3-598-21508-8") == True
```

k) ある文字列が回文か判定するプログラムを書け。ただし、自然な英文が回文と判定されるよう注意せよ。考慮としては以下の assert が通る程度でよい。

5 PTs

```
1 def is_palindrome(s: str) -> bool:
2     pass
3
4 assert is_palindrome("madamimadam") == True
5 assert is_palindrome("madammadam") == True
6 assert is_palindrome("Madam, I'm Adam.") == True
```

```
7 assert is_palindrome("abcd") == False
8 assert is_palindrome("") == True
9 assert is_palindrome("a") == True
```

l) 改行文字 `\n` で区切られた文字列について、`\n` を削除して半角スペースで結合する関数を書け。 2 PTs

```
1 def split_join(s: str) -> str:
2     pass
3
4 assert split_join("Hello\nWorld\nPython") == "Hello World Python"
```

m) 文字列が整数かそうでないかを判定せよ。 3 PTs

```
1 def is_integer(s: str) -> bool:
2     pass
3
4 assert is_integer("123") == True
5 assert is_integer("-123") == True
6 assert is_integer("-123e4") == False
7 assert is_integer("-123.4") == False
8 assert is_integer("0") == True
9 assert is_integer("-0") == True
10 assert is_integer("") == False
```

n) (言語処理 100 本ノックより) 以下の文字列 3 PTs

Now I need a drink, alcoholic of course, after the heavy lectures involving quantum mechanics.

について、単語に分解し、各単語のアルファベットの文字数を出現順に並べたリストを出力せよ。

o) (IP アドレスの検証) IP アドレスは `192.168.0.1` のようなドット `.` で区切られた 4 つの数字で構成されている。各数字の値は `0` から `255` の範囲である必要がある。文字列が IP アドレスであるか否かを判定するプログラムを書け。 5 PTs

```
1 def is_valid_ip(s: str) -> bool:
2     pass
3
4 assert is_valid_ip("192.168.0.1") == True
5 assert is_valid_ip("256.0.0.0") == False
6 assert is_valid_ip("192.168.1") == False
```

p) (HTML タグの除去) HTML 文書の本文からタグを取り除いて出力する関数を作成せよ。正規表現を使う場合は適宜調べよ。 8 PTs

```
1 def remove_html_tags(html_str):
2     pass
3
4 html_doc = """
5 <html>
```

```

6 <head>
7 <title>HTML リムーバー</title>
8 </head>
9 <body>
10 <h1>はじめに</h1>
11 <p>この文書は <a href="https://example.com">HTML</a> タグの取り除き方について
    説明しています。</p>
12 <p>さまざまな方法がありますが、Python の<b>正規表現</b>を用いると比較的簡単に実装で
    きます。</p>
13 </body>
14 </html>
15 """
16
17 assert remove_html_tags(html_doc) == "はじめに\n この文書は HTML タグの取り除
    き方について説明しています。 \n さまざまな方法がありますが、Python の正規表現を用いる
    と比較的簡単に実装できます。"

```

q) (パスワード検証) パスワード文字列の書式を検証するプログラムを書け。書式の要件は以下の通りである。

3 PTs

- 長さ 8 文字以上
- 英大文字、英小文字、数字、記号 (!@#\$\$%^&*()_+) を各 1 文字以上含むこと

```

1 def is_valid_password(s: str) -> bool:
2     pass
3
4 assert is_valid_password("Abc123!@") == True
5 assert is_valid_password("Ab123!@") == False
6 assert is_valid_password("abcdefgh") == False
7 assert is_valid_password("ABCD1234") == False

```

r) 小文字を大文字に変換するプログラムを書け。ただし組み込みの `upper` メソッドは使わず、`ord`、`chr` を使って書け。適宜 ASCII コード表を参照せよ。

3 PTs

```

1 def upper(s: str) -> str:
2     pass
3
4 assert upper("Hello, World") == "HELLO, WORLD"

```

s) (シーザー暗号) シーザー暗号とは、アルファベットを `N` 文字ずらした文字に置き換えることで平文を暗号化する、簡単な暗号化手法である。特に 13 文字ずらしたものを ROT13 と呼び、アルファベットが 26 文字であることから 2 回 ROT13 をかけると元に戻る性質を持つ。

4 PTs

例えば "ABC" に ROT13 をかけると "NOP" となる。 `rot13` 関数を実装せよ。

```

1 def rot13(s: str) -> str:
2     pass
3
4 assert rot13("Hello, World") == "Uryyb, Jbeyq"
5 assert rot13("Uryyb, Jbeyq") == "Hello, World"

```

t) 九九の表を表示せよ。表が崩れないよう文字幅に注意せよ。

6 PTs

u) 今年のカレンダーを表示せよ。表が崩れないよう文字幅に注意せよ。

7 PTs

v) (メールアドレスの検証) メールアドレスが正しい書式か判定するプログラムを書け。ただし検証するのは以下の簡略化した条件とする。

7 PTs

メールアドレスは `username@hostname` という書式をしている。`username` をローカル部、`hostname` をドメイン部と呼ぶ。

- ローカル部
 - ▶ 英数字、アンダースコア `_`、ピリオド `.` のみを使用可能
 - ▶ 1文字目は英数字のみを使える
 - ▶ 末尾にピリオドは使えない
 - ▶ ピリオドは連続できない
- ドメイン部
 - ▶ 英数字、ハイフン `-`、ピリオド `.` のみを使用可能
 - ▶ 末尾にピリオドは使えない
 - ▶ ピリオドは連続できない
- ローカル部とドメイン部以外は存在しない

```
1 def is_valid_email(email_str: str) -> bool:
2     pass
3
4     assert is_valid_email("john.doe@example.com") == True
5     assert is_valid_email("john..doe@example.com") == False
6     assert is_valid_email("john.doe.@example.com") == False
7     assert is_valid_email("john@ex@mple.com") == False
8     assert is_valid_email("_john@example.com") == False
9     assert is_valid_email("john@example.music") == True
10    assert is_valid_email("john@example..com") == False
11    assert is_valid_email("john@example.com.") == False
```

7. (リスト) 以下のプログラムを作成せよ。

a) 整数のリストについて、各要素を2乗する関数を作成せよ。

1 PT

```
1 def square(ls: list[int]) -> list[int]:
2     pass
3
4 assert square([1,2,3,4]) == [1,4,9,16]
```

b) 整数のリストについて、偶数のリストのみ抽出する関数を作成せよ。

1 PT

```
1 def filter_even(ls: list[int]) -> list[int]:
2     pass
3
4 assert filter_even([1,2,3,4]) == [2,4]
```

c) 2つのリストについて、要素を交互に含む新しいリストを作成する関数を作成せよ。

2 PTs

```
1 def interleave(xs: list, ys: list) -> list:
2     pass
3
4 assert interleave([1,3,5], [2,4,6]) == [1,2,3,4,5,6]
```

d) リストの各要素に指定した要素を挿入するプログラムを作れ。

2 PTs

```
1 def intersperse(e, xs: list) -> list:
2     pass
3
4 assert intersperse(',', ["a", "b", "c"]) == ["a", ",", "b", ",", "c"]
```

e) リストの各要素に指定した要素を挿入したのち、文字列結合するプログラムを作れ。

2 PTs

```
1 def intercalate(e, xs: list) -> list:
2     pass
3
4 assert intercalate(',', ["a", "b", "c"]) == "a,b,c"
```

f) リストのリストを転置するプログラムを作れ。

3 PTs

```
1 def transpose(xs: list[list]) -> list[list]:
2     pass
3
4 assert transpose([[1,2,3], [4,5,6]]) == [[1,4],[2,5],[3,6]]
```

g) リストの各要素に関数を適用する関数を作れ。必要であれば `map` について調べよ。

4 PTs

```
1 def my_map(f, xs: list) -> list:
```

```

2     pass
3
4     def square(x: int) -> int:
5         return x * x
6
7     assert my_map(square, [1,2,3,4]) == [1,4,9,16]

```

h) リストの各要素を条件に基づいてフィルタする関数を作れ。必要であれば `filter` について調べよ。

4 PTs

```

1     def my_filter(f, xs: list) -> list:
2         pass
3
4     def even(x: int) -> bool:
5         return x % 2 == 0
6
7     assert my_filter(even, [1,2,3,4]) == [2,4]

```

i) リストを条件に基づいて分割する関数を作れ。

4 PTs

```

1     def partition(f, xs: list) -> (list, list):
2         pass
3
4     def even(x: int) -> bool:
5         return x % 2 == 0
6
7     assert partition(even, [1,2,3,4]) == ([2,4], [1,3])

```

j) リストの長さを返す関数を実装せよ。ただし `len()` は使わない。

2 PTs

```

1     def length(xs: list) -> int:
2         pass
3
4     assert length([]) == 0
5     assert length([1,2,3]) == 3

```

k) リストを逆順にする関数を実装せよ。ただし `.reverse()`, `reversed()` あるいはスライスの逆順記法は使わない。

2 PTs

```

1     def reverse(xs: list) -> list:
2         pass
3
4     assert reverse([]) == []
5     assert reverse([1,2,3]) == [3,2,1]

```

l) 英語の文章について、単語の順序を逆順にした文字列を返す関数を作れ。句読点は考慮しなくてよい。

2 PTs

```

1     def reverse_words(text: str) -> str:
2         pass
3
4     assert reverse_words("The quick brown fox") == 'fox brown quick The'

```


m) 英語の文章について、各単語を逆にした文字列を返す関数を作れ。句読点は考慮しなくてよい。

2 PTs

```
1 def reverse_each_words(text: str) -> str:
2     pass
3
4 assert reverse_each_words("The quick brown fox") == 'ehT kciuq nworb xof'
```

n) 2つのリストの共通リストを返すプログラムを作れ。

3 PTs

```
1 def intersection(xs: list, ys: list) -> list:
2     pass
3
4 assert intersection(["Good", "morning", "everyone"], ["Good", "evening", "everyone", "!"]) == ["Good", "everyone"]
```

o) リストの連続かつ重複した要素を1つにしたリストを返せ。

3 PTs

```
1 def uniq_consecutive(xs: list) -> list:
2     pass
3
4 assert uniq_consecutive(['a', 'b', 'b', 'a', 'c', 'c', 'c']) == ['a', 'b', 'a', 'c']
```

p) リストの連続かつ重複した要素をサブリストとしたリストを返せ。

3 PTs

```
1 def pack(xs: list) -> list[list]:
2     pass
3
4 assert pack(['a', 'b', 'b', 'a', 'c', 'c', 'c']) == [['a'], ['b', 'b'], ['a'], ['c', 'c', 'c']]
```

q) (ランレングス圧縮) リストをランレングス圧縮 (run-length encoding) せよ。リストのランレングス圧縮とは、リストの連続かつ重複した要素 E とその要素数 N をタプル (N, E) で表現したリストである。必要であれば前問の結果を使ってよい。

5 PTs

```
1 def encode_run_length(xs: list) -> list[(int, Any)]:
2     pass
3
4 assert encode_run_length(['a', 'b', 'b', 'a', 'c', 'c', 'c']) == [(1, 'a'), (2, 'b'), (1, 'a'), (3, 'c')]
```

r) (ランレングス圧縮の解凍) 前問からの続きで、ランレングス圧縮したデータを解凍 (decode) せよ

3 PTs

```
1 def decode_run_length(xs: list[(int, Any)]) -> list:
2     pass
3
```

```
assert
4 decode_run_length(encode_run_length(['a','b','b','a','c','c','c'])) ==
  ['a','b','b','a','c','c','c']
```

s) リストの各要素を重複させたリストを作れ。

2 PTs

```
1 def duplicate(xs: list) -> list:
2     pass
3
4 assert duplicate([1,2,3]) == [1,1,2,2,3,3]
```

t) リストの各要素を `n` 回重複させたリストを作れ。

2 PTs

```
1 def duplicate_n(xs: list, n: int) -> list:
2     pass
3
4 assert duplicate([1,2,3], 2) == [1,1,2,2,3,3]
5 assert duplicate([1,2,3], 3) == [1,1,1,2,2,2,3,3,3]
```

u) リストの各要素をグループ化する関数を作れ。

4 PTs

```
1 def group(xs: list) -> list[list]:
2     pass
3
4 assert group([1,1,2,2,3,3]) == [[1,1], [2,2], [3,3]]
```

v) 文字列を区切り文字で分割したリストにする関数を作れ。

4 PTs

```
1 def split(s: str, text: str) -> list:
2     pass
3
4 assert split(',', "Hello,World") == ["Hello", "World"]
```

w) リストの順列を作る関数を作れ。必要であれば `itertools.permutation` を参考にしてよいが、実装にこの関数を使うな。

5 PTs

```
1 def permutation(xs: list) -> list:
2     pass
3
4 assert sorted(permutation(["a", "b", "c"])) == sorted(['a', 'b', 'c'],
  ('a', 'c', 'b'), ('b', 'a', 'c'), ('b', 'c', 'a'), ('c', 'a', 'b'),
  ('c', 'b', 'a'))
```

x) リストの組み合わせを列挙する関数を作れ。ここでリストの組み合わせ (combination) とは N 個の要素を持つリストから K 個選んだ場合の組み合わせを重複なしで列挙するものである。必要であれば `itertools.combination` を参考にしてよいが、実装にこの関数を使うな。

5 PTs

```
1 def combination(xs: list, k: list) -> list:
```

```

2     pass
3
4     assert list(combinations(['a','b','c'], 2)) == [('a', 'b'), ('a', 'c'),
    ('b', 'c')]

```

y) 整数 0,1,2,3,4,5 から異なる 4 つの数字を選んで 4 桁の整数を作る。2400 より大きくなる数字を列挙せよ。

4 PTs

z) (4 つの 4) 4 つの 4 と四則演算を用いて 10 を作れ。計算式を出力せよ。この問題については以下を参照せよ。

6 PTs

• [4 つの 4 - Wikipedia](#)

aa) リストをスライスする関数を作れ。スライスは 2 つの添字 I と K を指定して行われる。 I はスライスを開始する添字、 K は自身を含まないスライス終端の添字である。

2 PTs

```

1 def slice(i: int, k: int, xs: list) -> list:
2     pass
3
4     assert slice(2, 5, [1,2,3,4,5,6]) == [3,4,5]

```

ab) リストを指定した数の要素だけ回転させる関数を作れ。

2 PTs

```

1 def rotate(n: int, xs: list) -> list:
2     pass
3
4     # 左側に回転
5     assert rotate(3, ['a', 'b', 'c', 'd', 'e', 'f', 'g']) == ['d', 'e', 'f',
6     'g', 'a', 'b', 'c']
7     # 右側に回転
8     assert rotate(-3, ['a', 'b', 'c', 'd', 'e', 'f', 'g']) == ['e', 'f',
9     'g', 'a', 'b', 'c', 'd']

```

ac) リストの指定した位置に指定した要素を挿入するプログラムを書け。

2 PTs

```

1 def insert(elem, n: int, xs: list) -> list:
2     pass
3
4     assert insert('b', 1, ['a', 'c', 'd']) == ['a', 'b', 'c', 'd']

```

ad) 指定した範囲に含まれる整数のリストを返す関数を書け。ただし終端の数は含まない。

2 PTs

```

1 def my_range(start: int, end: int) -> list[int]:
2     pass
3
4     assert my_range(2, 10) == [2,3,4,5,6,7,8,9]

```

ae) 指定された要素数でゼロ埋めされたリストを返す関数を書け。

2 PTs

```

1 def zeros(n: int) -> list[int]:
2     pass
3
4 assert zeros(5) == [0, 0, 0, 0, 0]

```

af) 指定された要素数 n, m でゼロ埋めされた二次元リストを返す関数を書け。

2 PTs

```

1 def zeros(n: int, m: int) -> list[list[int]]:
2     pass
3
4 assert zeros(3, 2) == [[0, 0], [0, 0], [0, 0]]

```

一般の次元に拡張された関数については `numpy.zeros` を参考にせよ。

ag) 指定された要素数 n, m で1埋めされた二次元リストを返す関数を書け。

2 PTs

```

1 def ones(n: int, m: int) -> list[list[int]]:
2     pass
3
4 assert ones(3, 2) == [[1, 1], [1, 1], [1, 1]]

```

一般の次元に拡張された関数については `numpy.ones` を参考にせよ。

ah) 指定された要素数 n, m の二次元リストを返す関数を書け。ただし各要素について0または1のランダムな値をセットせよ。必要であれば `random` モジュールについて調べよ。余裕があればこの関数をテストする方法を考えてみよ。

2 PTs

```

1 def rand_field(n: int, m: int) -> list[list[int]]:
2     pass

```

ai) 前問の `rand_field` で生成した2次元リストについて、要素が1なら#、0なら.を描画するプログラムを書け。各行の描画が終わったら改行するようにせよ。

2 PTs

```

1 def print_field(ls: list[list]):
2     pass

```

aj) (ライフゲーム) 前問からの続きで、生成した2次元リスト L の各要素について、以下の条件で値を更新する関数を作成せよ。

7 PTs

- 2次元リスト L の要素 $L(x, y)$ について (x, y は要素の添字)、 $L(x + dx, y + dy)$ を隣接セルと呼ぶ。ここで $(dx, dy) \in \{(1, 1), (1, 0), (1, -1), (0, 1), (0, -1), (-1, 1), (-1, 0), (-1, -1)\}$
- 要素の値が1の時生存、0の時死と呼ぶ。
- 要素 $L(x, y)$ が死で隣接セルの値が3つ生存している場合、要素を生存に更新する (誕生)。
- 要素 $L(x, y)$ が生存で隣接セルが2つないし3つ生存している場合、要素は生存のままとする (維持)。
- 要素 $L(x, y)$ が生存で隣接セルが1以下ないし4つ以上生存の場合、要素は死に更新する (過疎および過剰による死)。

```
1 def tick_field(ls: list[list]) -> list[list]:  
2     pass
```

完成したら、`sleep` 関数などを参考に、ある程度の大きさのフィールドを作り、更新を繰り返しながら適当な時間間隔で表示してみよ。

この問題の出題意図についてはリンク先（【実践事例】情報I(3) P.83-90）を参照せよ。

8. (実践問題) 以下の問題を解け。必要に応じて `dict`, `set`, 正規表現について調べよ。

a) 以下のリンクを開き、ページを保存せよ。

2 PTs

<https://raw.githubusercontent.com/dolph/dictionary/master/enable1.txt>

これは英単語が 1 行 1 語で辞書順に格納されたファイルである。このファイルの単語数をカウントするプログラムを書け。必要であれば `open`, `close`, `readlines` について調べよ。

以降の問題は下記リンクの問題を参考にしている。

[2021-07-19] Challenge 399 [Easy] Letter value sum

b) 文字数 20 以上の単語はいくつあるか。それはどのような単語か。

2 PTs

c) 辞書中で最も長い単語は何か。

2 PTs

d) 各単語を頭文字が同じものにグループ化したとき、各頭文字ごとの単語数を表示せよ。

3 PTs

e) 辞書からアナグラムを探して表示せよ。アナグラムとは `debit`, `bited` のように並び替えると同じになる単語である。

4 PTs

f) 単語のスコアを以下のように定義する。

3 PTs

- ・ 単語の点数は、各文字の点数の総和になる。
- ・ 各文字の点数は、`a` は 1 点から `z` は 26 点までアルファベット順に増加する。

単語のスコアを計算するプログラムを作れ。

```
1 def lettersum(s: str) -> int:
2     pass
3
4 assert lettersum("") == 0
5 assert lettersum("a") == 1
6 assert lettersum("z") == 26
7 assert lettersum("cab") == 6
8 assert lettersum("excellent") == 100
9 assert lettersum("microspectrophotometries") == 317
```

g) 辞書から最大のスコアを持つ単語を探せ。

3 PTs

h) スコアが偶数となる単語はいくつあるか？

3 PTs

i) スコアが 100 になる単語は 1921 存在する。100 の次に単語の存在頻度が多いものは何か。

4 PTs

j) `cytotoxicity` と `unreservedness` は同じスコア 188 を持つが、互いに共通する文字を持たない。このような単語は他にあるか調べ、存在するなら挙げてみよ。

6 PTs

k) 30 人のクラスで誕生日が同じクラスメートが存在する確率を計算せよ。ここで 1 年は 365 日としてよい。

2 PTs

l) 前問についてシミュレーションを行い、実際に誕生日が衝突する様子を確認してみよ。 **random** モジュールについて調べよ。 2 PTs

m) じゃんけんを行うプログラムを作ってみよ。 4 PTs

n) ビデオゲーム上での戦闘をシミュレーションするプログラムを作成する。 7 PTs

戦闘は以下の条件で推移するものとする。キャラクターシートと行動パターンの表を参照しつつ、どちらが勝ったか出力するプログラムを作成せよ。

- ・ 自分と敵が交互に行動する
 - ▶ 自分と敵双方の行動が完了したことをターンが経過したといい、1ターン加算する
 - ▶ ターンは1から起算する
 - ▶ 各キャラクターは行動パターンの行動を条件に従い行う。or で書かれた行動は均等な確率でいずれかが実行される
- ・ すばやさの高いキャラクターが先に行動する
- ・ 敵の HP を 0 にすれば勝ち、自分の HP が 0 になれば負け
- ・ 自分を「ゆうしゃ」、敵を「りゅうおう」とする
 - ▶ ゆうしゃはステータスに装備の値を加算する
 - ▶ りゅうおうは $n+4$ ターン目で n ターン目の行動を繰り返す

名前	ステータス	そうび	行動パターン
ゆうしゃ	HP: 210 MP: 190 こうげき: 140 すばやさ: 130 しゅび: 60	ロトのつるぎ: こうげき+40 ロトのよろい: しゅび+28 みかがみのたて: しゅび+20 りゅうのうろこ: しゅび+5	たたかう ベホイミ (HP50% 未満時)
りゅうおう	HP: 350 こうげき: 150 すばやさ: 90 しゅび: 150	なし	1 ターン: はげしいほのお 2 ターン: こうげき 3 ターン: ひのいき or かえんのいき 4 ターン: こうげき or ひのいき

表 1: キャラクターシート

各行動パターンとその効果は以下の通り。

行動パターン	効果
たたかう	<p>自分がたたかう際は以下の点数を敵 HP から減算する $(\text{乱数} * (\text{こうげき} - (\text{敵しゅび}/2) + 1)/256 + \text{こうげき} - \text{敵しゅび}/2)/4$ ここで乱数は 0~255 の値である。</p> <p>敵がたたかう際は以下の点数を自分の HP から減算する $(\text{乱数} * (\text{敵こうげき} - \text{しゅび}/2 + 1)/256 + \text{敵こうげき} - \text{しゅび}/2)/4$</p>
ベホイミ	<p>自分の MP8 点を減算し自分の現在 HP に 85~100 点加算する。 加算された HP は HP の初期値を超えることはない。 MP が 8 以上残っていない場合は使用せず他の行動を取る。</p>
はげしいほのお	敵使用時、自分の HP から 48~54 点を減算する。
かえんのいき	敵使用時、自分の HP から 12~16 点を減算する。
ひのいき	敵使用時、自分の HP から 9~15 点を減算する。

表 2: 行動パターン

o) 前問について、さらなるシミュレーション機能を提案し追加せよ。以下のような仕様案を参考にしてもよい。 7 PTs

- ・ 行動のバリエーションを追加する
- ・ そうびのバリエーションや機能を追加する
- ・ アイテムという概念を導入する
- ・ 一定の確率で大ダメージを与える
- ・ 一定の確率で回避・ミスする
- ・ 味方を複数にする
- ・ 敵を複数にする
- ・ 行動順にも乱数による幅を持たせる

p) (HQ9+) 難解プログラミング言語 HQ9+を実装せよ。HQ9+の仕様は以下の通り。 5 PTs

- ・ HQ9+は 4 つの命令 H,Q,9,+から構成される。
- ・ H コマンドは Hello, world を表示する。
- ・ Q コマンドは実行しているプログラム自身を表示する (Quine)
- ・ 9 コマンドは 『99 Bottles of Beer』の歌詞を出力する。
- ・ + コマンドはアキュムレータをインクリメントする

q) (Brainfuck) 難解プログラミング言語 Brainfuck を実装せよ。Brainfuck の仕様は以下の通り。 5 PTs

Brainfuck 処理系はインストラクションポインタ、配列 (30000 要素以上)、前記の配列を指すデータポインタ、入出力からなる。また命令語は以下の 8 つであり、以下ポインタとはデータポインタのことを指す。

- ・ > ポインタをインクリメントする
- ・ < ポインタをデクリメントする

- ・ **+** ポインタが指す値をインクリメントする
- ・ **-** ポインタが指す値をデクリメントする
- ・ **-** ポインタが指す値をデクリメントする
- ・ **.** ポインタが指す値を出力する
- ・ **,** 入力から 1 バイト読み込み、ポインタが指す先に代入する
- ・ **[** ポインタが指す値が 0 なら対応する **]** の直後にジャンプする。
- ・ **]** ポインタが指す値が 0 でないなら対応する **[** の直後にジャンプする。

```
1 def bf(program: str):  
2     pass  
3  
4 bf("+++++++[>+++++>++++++>++++++>++++>++++++>+<<<<-]>+  
+.+.>--..++.>+.>--..<.++.-----.<-.>+.>.") # Hello World! と出力さ  
れる (らしい)
```

9. (ストラウストラップのプログラミング入門より) 以下に挙げたテーマから少なくとも1つを選んで議論せよ(800文字以上1200文字以下)。根拠に基づいて議論すること。

a) ソフトウェアとは何か。ソフトウェアはなぜ重要なのか。 10 PTs

b) 計算機科学(コンピュータサイエンス)とプログラミングの違いはなにか。何が重要なのか。 10 PTs

c) ソフトウェアが正常に動作しない場合、どのような問題が起きるか。具体例を挙げること。 10 PTs

d) ソフトウェア開発が困難になる理由はなにか。 10 PTs

10. (ストラウストラップのプログラミング入門より) 以下に挙げたテーマから少なくとも1つを選んで議論せよ。文字数は問わないが端的かつ必要十分な議論をすること。また、根拠に基づいて意見を述べること。

a) 自分が何らかの知識を持っている職業について、その仕事にソフトウェアがどのように関わっているかを分析してみよ。 10 PTs

b) 優れたプログラマーないしエンジニアに共通する特性を議論せよ。 10 PTs

c) 自分が関心のあるソフトウェアを複数挙げ、その中で自分が将来関わりたいものを選び、なぜ選んだか理由を述べよ。 10 PTs

d) 人間が行う活動のうち、いかなる形(間接的に)でもコンピュータが関わらない活動を挙げよ。根拠を述べること。 10 PTs

11. (令和5年度技術士第二次試験(情報工学部門)改題) 生成AIの技術レベルが著しく向上し、用途も広がっている一方で、その利活用・普及に伴う社会的課題も顕在化してきている。このような状況を踏まえ、生成AIを活用する具体的なサービスを想定し、その構築や運用を行う立場で以下の問いに答えよ。

a) 技術者としての立場で多面的な観点から3つの課題を抽出し、それぞれの観点を明記した上で、その課題の内容を示せ。課題を表す用語としては以下を参考にせよ。 10 PTs

幻覚(ハルシネーション)、差別(バイアス)、プライバシー、環境問題(計算コスト)、データ汚染、悪用。

b) 前問で抽出した課題のうち最も重要と考える課題を1つ挙げ、その課題に対する解決策を、情報工学の専門技術用語を交えて示せ。 10 PTs

c) 前問で示した解決策を実施するに当たり生じる波及効果と専門技術を踏まえた懸念事項への対応策を示せ。 10 PTs

d) 上記の業務遂行に当たり、技術者としての倫理・社会の持続可能性の観点から必要となる要件・留意点を題意に即して述べよ。 10 PTs

Assignment	1	2	3	4	5	6	7	8	9	10	11	Total
Points	11	7	6	19	31	80	105	64	40	40	40	443
Awarded												

Σ : ____ / 443 PTs