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

```python
def speed(start: int, end: int, time_elapsed: int) -> int:
    # TODO: 以下に`distance` 変数を定義して正しい値を計算せよ。


    # 以下の行は変更しないこと
    return distance // time_elapsed

assert speed(10, 30, 10) == 2
assert speed(10, 30, 2) == 15
assert speed(10, 31, 10) == 2
```

b) **（分岐）** 奇数を判定するプログラムを書け。 1 PT

```python
def is_odd(n: int) -> bool:
    pass

assert is_odd(1) == True
assert is_odd(2) == False
assert is_odd(101) == True
```

c) **（エラー処理）** 前々問の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

```python
def speed(start: int, end: int, time_elapsed: int) -> int:
    # TODO
    pass

assert speed(10, 30, 0) == "Zero division error"
assert speed(10, 30, 10) == 2
assert speed(10, 30, 2) == 15
assert speed(10, 31, 10) == 2
```

d) **（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

```python
def fact(n: int) -> int:
    pass

assert fact(0) == 1
assert fact(1) == 1
assert fact(2) == 2
assert fact(5) == 120
```

e) **（`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

```python
def fact(n: int) -> int:
    pass

assert fact(0) == 1
assert fact(1) == 1
assert fact(2) == 2
assert fact(5) == 120
```

f) （**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

```python
def fact(n: int) -> int:
    pass

assert fact(0) == 1
assert fact(1) == 1
assert fact(2) == 2
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。

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

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。以降のプログラム作成問題でも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。 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。問に回答せよ。

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

```
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

```
def proc1():
    print("A")

def proc2():
    proc3()
    print("B")
    proc1()

def proc3():
    print("C")
    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

```
x = 1
y = 2
z = 3

x = y
y = z
z = x
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

```
def calc(x: float, y: float):
    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

```
def makeNewArray(input: list) -> list:
    out = [input[0]]
    for i in range(1, len(input)):
        tail = out[len(out)-1]
        out.append(tail + input[i])
    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) 割り算を実行し、商 (quotient) と余り (reminder) のペアを返す関数を実装せよ。 1 PT

```python
def quot_rem(x: int, y: int) -> (int, int):
    pass

assert quot_rem(4, 3) == (1, 1)
assert quot_rem(8, 3) == (2, 2)
assert quot_rem(10, 5) == (2, 0)
```

e) 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

```python
def minute2hours(m: int) -> (int, int):
    pass

assert minute2hours(30) == (0, 30)  # 30 分は 0 時間 30 分
assert minute2hours(150) == (2, 30) # 150 分は 2 時間 30 分
```

f) 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

```python
def is_adult(age: int) -> bool:
    pass

assert is_adult(17) == False
assert is_adult(18) == True
```

g) 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

h) ある映画館の料金は以下のようになっている。 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。

```
def admission_fee(age: int) -> int:
    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

```python
def fib(n: int) -> int:
    pass

assert fib(1) == 1
assert fib(2) == 1
assert fib(3) == 2
assert fib(4) == 3
assert fib(5) == 5
assert fib(6) == 8
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

```python
def gcd(a: int, b: int) -> int
    pass

assert gcd(10, 8) == 4
assert gcd(31, 7) == 1
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

```python
def is_prime(n: int) -> bool
    pass

assert is_prime(1) == False
assert is_prime(2) == True
assert is_prime(3) == True
assert is_prime(4) == False
assert is_prime(11) == True
assert is_prime(105) == False
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

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

```python
def count(s: str, c: str) -> int:
    pass

assert count("AaAa", "a") == 4
```

b) スライス記法を使って `Hello, World` の先頭文字から 1 文字おきの文字列を取り出すプログラムを完成させよ。 1 PT

```python
hello = "Hello, World"
result = hello[???]

assert result == "Hlo ol"
```

c) **（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

```python
def solve_tanuki(s: str):
    pass
```

d) `"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

e) 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

```python
def upper_remove_vowels(s: str) -> str:
    pass

assert upper_remove_vowels("FM Yokohama") == "FM YKHM"
```

f) ある文字列が回文か判定するプログラムを書け。 3 PTs

```python
def is_palindrome(s: str) -> bool:
    pass

assert is_palindrome("madamimadam") == True
assert is_palindrome("madammadam") == True
assert is_palindrome("abcd") == False
assert is_palindrome("") == True
assert is_palindrome("a") == True
```

g) 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

```python
def is_palindrome(s: str) -> bool:
    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
```

h) 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"
```

i) 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
```

j) **（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
```

k) **（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>
```

```
<body>
<h1>はじめに</h1>
<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>
<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>
</body>
</html>
"""

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。"
```

l) **（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

```
def is_valid_password(s: str) -> bool:
    pass

assert is_valid_password("Abc123!@")  == True
assert is_valid_password("Ab123!@")   == False
assert is_valid_password("abcdefgh")  == False
assert is_valid_password("ABCD1234")  == False
```

m) 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

```
def upper(s: str) -> str:
    pass

assert upper("Hello, World") == "HELLO, WORLD"
```

n) **（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。

```
def rot13(s: str) -> str:
    pass

assert rot13("Hello, World") == "Uryyb, Jbeyq"
assert rot13("Uryyb, Jbeyq") == "Hello, World"
```

o) 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

p) 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

q) **（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

```python
def is_valid_email(email_str: str) -> bool:
    pass

assert is_valid_email("john.doe@example.com") == True
assert is_valid_email("john..doe@example.com") == False
assert is_valid_email("john.doe.@example.com") == False
assert is_valid_email("john@ex@mple.com") == False
assert is_valid_email("_john@example.com") == False
assert is_valid_email("john@example.music") == True
assert is_valid_email("john@example..com") == False
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

```
def square(ls: list[int]) -> list[int]:
    pass

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

```
def filter_even(ls: list[int]) -> list[int]:
    pass

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

```
def interleave(xs: list, ys: list) -> list:
    pass

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

```
def intersperse(e, xs: list) -> list:
    pass

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

```
def intercalate(e, xs: list) -> list:
    pass

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

```
def transpose(xs: list[list]) -> list[list]:
    pass

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

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

```
    pass

def square(x: int) -> int:
    return x * x

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

```
def my_filter(f, xs: list) -> list:
    pass

def even(x: int) -> bool:
    return x % 2 == 0

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

```
def partition(f, xs: list) -> (list, list):
    pass

def even(x: int) -> bool:
    return x % 2 == 0

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()` は使うな。 2 PTs

```
def length(xs: list) -> int:
    pass

assert length([]) == 0
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うな。 2 PTs

```
def reverse(xs: list) -> str:
    pass

assert reverse([]) == []
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

```
def reverse_words(text: str) -> str:
    pass

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

```
def reverse_each_words(text: str) -> str:
    pass

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

```
def intersection(xs: list, ys: list) -> list:
    pass

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

```
def uniq_consecutive(xs: list) -> list:
    pass

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

```
def pack(xs: list) -> list[list]:
    pass

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

```
def encode_run_length(xs: list) -> list[(int, Any)]:
    pass

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

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

```

```
assert
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

```
def duplicate(xs: list) -> list:
    pass

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

```
def duplicate_n(xs: list, n: int) -> list:
    pass

assert duplicate([1,2,3], 2) == [1,1,2,2,3,3]
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

```
def group(xs: list) -> list[list]:
    pass

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

```
def split(s: str, text: str) -> list:
    pass

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

```
def permutation(xs: list) -> list:
    pass

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

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

```
    pass

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

```
def slice(i: int, k: int, xs: list) -> list:
    pass

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

```
def rotate(n: int, xs: list) -> list:
    pass

# 左側に回転
assert rotate(3, ['a', 'b', 'c', 'd', 'e', 'f', 'g']) == ['d', 'e', 'f',
'g', 'a', 'b', 'c']
# 右側に回転
assert rotate(-3, ['a', 'b', 'c', 'd', 'e', 'f', 'g']) == ['e', 'f',
'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

```
def insert(elem, n: int, xs: list) -> list:
    pass

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

```
def my_range(start: int, end: int) -> list[int]:
    pass

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

```python
def zeros(n: int) -> list[int]:
    pass

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

```python
def zeros(n: int, m: int) -> list[list[int]]:
    pass

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

```python
def ones(n: int, m: int) -> list[list[int]]:
    pass

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

```python
def rand_field(n: int, m: int) -> list[list[int]]:
    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

```python
def print_field(ls: list[list]):
    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）。

```python
def tick_field(ls: list[list]) -> list[list]:
    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。

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。

```
def lettersum(s: str) -> int:
    pass

assert lettersum("") == 0
assert lettersum("a") == 1
assert lettersum("z") == 26
assert lettersum("cab") == 6
assert lettersum("excellent") == 100
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

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案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	8	7	6	11	21	64	105	32	40	40	40	374
Awarded												

$\sum$: ____ / 374 PTs