アルゴリズム入門　共通問題（2021 年度 A セメスター試験）

[試験日時：2022 年 1 月 31 日第 4 限（60 分），答案用紙：1 部，持ち込み一切不可]

内容に関する質問は受け付けない．問題の記述が曖昧な場合は，適切な仮定をおいて回答し，どのような仮定をおいたのか明記せよ．

問題 1

配列の要素を逆順に並び替える関数を作成したい．例えば変数 `x = [3, 5, 2, 1, 4, 6]` を引数として与えると，この関数の実行後には `x = [6, 4, 1, 2, 5, 3]` となることを期待している．これについて以下の問に答えよ．

(1) これを実現するため，図 1 の関数 `reverse1` を作成した．なお `ita.array.make1d(n)` は長さ `n` で全要素が 0 の配列を返す．`reverse1([1, 2, 3])` を実行した際，下線部 (a) は 3 回実行されるが，そのそれぞれの直後での整数 `i` の値と配列 `r` の全要素を答えよ．

(2) 関数 `reverse1` のように新しい配列 `r` を用意するのは無駄だと考え，図 2 の関数 `reverse2` を作成した．この関数は期待通りには動作しない．`reverse2([1, 2, 3, 4])` を実行した際，下線部 (b) は 4 回実行されるが，そのそれぞれの直後の整数 `i` の値と配列 `x` の全要素を答えよ．

(3) 関数 `reverse2` を修正するには配列要素を入れ替える必要があると考え，図 3 の関数 `swap` を作成した．この関数は，配列 `x` と整数 `i` および `j` が `swap(x, i, j)` と与えられたとき，`x[i]` と `x[j]` の内容を交換する．空欄 (c) ∼ (e) を埋めよ．

(4) 関数 `swap` を用いて図 4 の関数 `reverse3` を作成したが，これも正しく動作しない．`reverse3([1, 2, 3, 4])` を実行した際，下線部 (f) は 4 回実行されるが，そのそれぞれの直後の整数 `i` の値と配列 `x` の全要素を答えよ．

(5) 関数 `reverse3` を 1 カ所だけ修正して，期待通りに動作する関数を作成したい．どの行のどの部分をどのように修正すれば良いかを答えよ．

```
import ita
def reverse1(x):
  n = len(x)
  r = ita.array.make1d(n)
  for i in range(0, n):
    r[n - i - 1] = x[i]          (a)
  for i in range(0, n):
    x[i] = r[i]
```

図 1: 関数 `reverse1`

```
def reverse2(x):
  n = len(x)
  for i in range(0, n):
    x[n - i - 1] = x[i]          (b)
```

図 2: 関数 `reverse2`

```
def swap(x, i, j):
  t = [ (c) ]
  x[j] = [ (d) ]
  x[i] = [ (e) ]
```

図 3: 関数 `swap`

```
def reverse3(x):
  n = len(x)
  for i in range(0, n):
    j = n - i - 1
    swap(x, i, j)                (f)
```

図 4: 関数 `reverse3`

問題 2

配列を使って分数を表すことを考える．最初の要素は分子，次の要素は分母だとし，例えば配列 [2, 3] は分数 $\frac{2}{3}$ を表す．なお，分子・分母ともに正の整数であるとする．

(1) 以下の空欄 A ～ F を埋め，分数のかけ算・割り算・足し算を行う関数を完成させよ．例えば multiply([1, 2], [4, 3]) の返値は [2, 3] である．ただし reducer は約分を計算する関数であり，例えば reducer([3, 12]) の返値は [1, 4] である．

```
def multiply(a, b):    # かけ算
  return reducer([ A ,  B ])

def divide(a, b):      # 割り算
  return multiply(a, [ C ,  D ])

def add(a, b):         # 足し算
  return reducer([ E ,  F ])
```

(2) 約分を計算する関数 reducer を作成するため，最大公約数を計算する関数 gcd_naive を以下の通り作成した．gcd_naive(8, 6) を実行した場合，変数 a, b, d の値がそれぞれどのように変化し，最終的にどのような結果が返されるか，2 行程度で簡単に説明せよ。

```
def gcd_naive(a, b):
  d = a
  while (a % d != 0 or b % d != 0):
    d = d - 1
  return d
```

(3) 以下の空欄 G ～ J を埋め，約分を計算する関数 reducer を完成させよ．

```
def reducer(a):
  d = gcd_naive( G ,  H )
  return [ I ,  J ]
```

(4) 関数 gcd_naive(a, b) の a に関する時間計算量を O 記法を用いて答えよ．さらに a の 10 進数表記での桁数を n としたとき，n に関する時間計算量を O 記法を用いて答えよ．理由も簡単に述べること．なお，整数の基本的な計算に要する時間は桁数によらないとしてよい．

(5) 以下の関数 gcd_smart は gcd_naive とは異なるアルゴリズムによって最大公約数を計算する．この関数は，2 つの正の整数 a と b について「a を b で割ったあまりが r のとき，a と b の最大公約数は b と r の最大公約数に等しい」という性質を利用している．空欄 K ～ M を埋め gcd_smart を完成させよ．

```
def gcd_smart(a, b):
  while b > 0:
    r = a % b
    a =  K 
    b =  L 
  return  M 
```

(6) 関数 gcd_smart(a, b) の時間計算量を，a の 10 進数での桁数 n を用いて O 記法で答えよ．理由も簡単に述べること．なお，a を b で割ったあまりを r としたとき，$a > b > 0$ であれば $r < \frac{a}{2}$ であることを利用してよい．また，整数の基本的な計算に要する時間は桁数によらないとしてよい．

問題 3

ある Python 処理系での実行の様子を図 5 に示す．枠内がプログラムであり，これを上から順に実行している．枠外には直上の枠のプログラムの実行結果が示されている．これについて以下の問に答えよ．

(1) 下線部 (a) の実行結果が `1` ではなく `1.0` である理由を 1 行程度で答えよ．

(2) 下線部 (b) の実行結果は下線部 (a) と異なっている．この誤差の理由を 1 行程度で答えよ．

(3) 下線部 (c) の実行結果は，下線部 (b) の実行結果に比べ，下線部 (a) との差が大きくなっている．この誤差の理由を 1 行程度で答えよ．

(4) 下線部 (d) の実行結果が `0.0` である理由を 1 行程度で答えよ．

(5) 下線部 (e) の実行結果が下線部 (c) と異なる理由を 1 行程度で答えよ．（ヒント：コンピュータの中では二進数を使って数を表現している）

(6) 空欄 (f) を埋め，その理由を 1 行程度で答えよ．

```
def f(a):
  return ((1 / a + 1) - 1) * a
```

```
f(2)
```

`1.0`(a)

```
f(10)
```

`1.0000000000000009`(b)

```
f(1000 ** 4)
```

`1.000088900582341`(c)

```
f(1000 ** 6)
```

`0.0`(d)

```
f(1024 ** 4)
```

`1.0`(e)

```
f(1024 ** 6)
```

(f)

図 5: プログラムの実行結果