情報１　２学期期末考査

福井県立勝山高等学校

2023 年 11 月 30 日 3 限目

注意事項

- 開始のチャイムが鳴るまで開かないこと.
- チャイムの前に問題用紙・解答用紙に記名して良い.
- 解答は全て数値・番号・記号で答えること.
- 計算用紙として, 解答用紙の裏面を使用しても構わない.
- 終了後, 問題冊子は持ち帰ること.

＿＿＿年＿＿＿組＿＿＿番 氏名＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

1 各シミュレーションについて, それぞれの問いに答えよ. 【30 点】

- コイン投げのシミュレーション.

```
1 import random
2 N=int(input("how many coin",))
3 [(1)]
4 for i in range(N):
5  coin=random.randrange(2)
6  if coin==0:
7   count=count+1
8 print("probability of omote =",[(2)])
```

(1) コード中 [(1)] に当てはまるものとして適切なものを 1 つ選べ.

ⓐ
```
1 count=1
```

ⓑ
```
1 count=0
```

ⓒ
```
1 coin=0
```

ⓓ
```
1 coin=1
```

(2) 表の出た比率を表示したい場合, コード中 [(2)] に当てはまるものとして適切なものを 1 つ選べ.

ⓐ
```
1 n/count
```

ⓑ
```
1 count
```

ⓒ
```
1 coin
```

ⓓ
```
1 count/n
```

(3) 以下の記述のうち, 正しいものを全て選べ.

ⓐ コンピュータ内で実際にコインを投げて実験して確率を求めている.

ⓑ 0 か 1 の数字をコンピュータが好きに選び, 表か裏かを決めて確率を求めている.

ⓒ 0 か 1 の数字をコンピュータが無作為に選び, 表か裏かを決めて確率を求めている.

ⓓ 複数回実行した場合, 実行結果は毎回同じである.

ⓔ 複数回実行した場合, 実行結果は毎回同じとは限らない.

ⓕ 複数回実行した場合, 実行結果が同じになることはない.

(4) 実行後 10 と入力した場合に, 結果として表示され得るものを全て選べ.

ⓐ

```
1 0.5
```

ⓑ

```
1 0.55
```

ⓒ

```
1 0.1
```

ⓓ

```
1 0.435
```

(5) 実行後 100 と入力した場合に, 結果として表示され得るものを全て選べ.
(ただし, 例えば 0.10 は 0.1 のように表示されるものとする.)

ⓐ

```
1 0.545
```

ⓑ

```
1 0.555
```

ⓒ

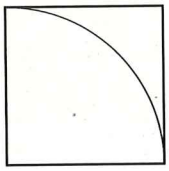
```
1 0.5
```

ⓓ

```
1 0.61
```

- モンテカルロ法 (円周率の近似値を求める). 手順は以下の通り.

a) 右図のように, 1 辺が 1 の正方形の中に $\frac{1}{4}$ 円を考える.
b) 0 以上 1 以下の小数乱数で, (x, y) 座標を生成する.
c) この座標が扇形の中に入っている ([(6)]) か否かを評価する.
d) b),c) を繰り返す.
e) (扇形内の個数)÷(点を打った個数) の値を 4 倍すると, π の近似値を出せる.



コードは下に示す通りである.

```
1 import random
2 N=int(input("how many point ",))
3 count=0
4 for i in range(N):
5   x=random.uniform(0,1)
6   y=random.uniform(0,1)
7   if [(6)]:
8     count=count+1
9 print([(7)])
```

(6) 手順・コード中 [(6)] に当てはまるものの組み合わせとして適切なものを 1 つ選べ.

選択肢	手順内	コード内
ⓐ	$x + y \leqq 1$	x+y<=1
ⓑ	$x^2 + y^2 \leqq 1$	x**2+y**2<=1
ⓒ	$x^2 + y^2 \leqq 1$	x*2+y*2<=1
ⓓ	$x + y < 1$	x+y<1

(7) π の近似値を表示したい場合, コード中 [(7)] に当てはまるものとして適切なものを 1 つ選べ.

ⓐ
```
1 count/4*N
```

ⓑ
```
1 count*N/4
```

ⓒ
```
1 count*4/N
```

ⓓ
```
1 count*4
```

(8) 以下の記述のうち, 正しいものを全て選べ.

ⓐ π の近似値が出せる理由は, 一辺が 1 の中に描いた半径

ⓑ π の近似値が出せる理由は, 一辺が 1 の中に描いた半径

ⓒ π の近似値が出せる理由は, 一辺が 1 の中に描いた半径
るから.

ⓓ 乱数で生成する x, y の値は, 整数値でもよい.

0×1÷10 0
1×4÷10 4
2×4÷10 8
3×4÷10 12
4×4÷10 16
5×4÷10 20
6×4÷10 24
7×4÷10 28
8×4÷10 32
9×4÷10 36
10×4÷10 40

(9) 実行後 10 と入力した場合に, 結果として表示され得るものを
(ただし, 例えば 3.30 は 3.3 のように表示されるものとする.

ⓐ
```
1 2
```

ⓑ
```
1 3
```

ⓒ
```
1 3,2
```

ⓓ
```
1 3.14
```

面積π

(10) $\frac{1}{4}$ 円を考えるのではなく, 半径 1 の円を考えて円周率の近似値を求めることもできる.
この考え方をもとに, π の近似値を求めることのできるプログラムを 1 つ選べ.

ⓐ
```
1 N=int(input("how many point ",))
2 count=0
3 for i in range(N):
4  x=random.uniform(-1,1)
5  y=random.uniform(-1,1)
6  if x**2+y**2<=1:
7   count=count+1
8 print(count*4/N)
```

ⓑ
```
1 N=int(input("how many point ",))
2 count=0
3 for i in range(N):
4  x=random.uniform(-1,1)
5  y=random.uniform(-1,1)
6  if x**2+y**2<=1:
7   count=count+1
8 print(count/N)
```

ⓒ
```
1 N=int(input("how many point ",))
2 count=0
3 for i in range(N):
4  x=random.uniform(0,1)
5  y=random.uniform(0,1)
6  if x**2+y**2<=1:
7   count=count+1
8 print(count*4/N)
```

ⓓ
```
1 N=int(input("how many point ",))
2 count=0
3 for i in range(N):
4  x=random.uniform(0,1)
5  y=random.uniform(0,1)
6  if x**2+y**2<=1:
7   count=count+1
8 print(count/N)
```

- それぞれの出た目の比率を求めるサイコロ投げのシミュレーション．以下の手順で行う．

  a) それぞれの目が何回でたかを記録する配列を作成する．
  b) 以下の操作を入力された回数 n だけ繰り返す．
     i) 0 から 5 の乱数を生成する．
     ii) 生成した乱数番目の配列の値に 1 を加える．
  c) 配列のそれぞれの数値を n で割る．
  d) 表示．

コードは以下の通り．

```
1 import random
2 [(11)]
3 n=int(input("how many = "))
4 for i in range(n):
5  me=random.randint(0,5)
6  count[me]=count[me]+1
7 for i in range(len(count)):
8  count[i]=count[i]/n
9 print([(12)])
```

(11) コード中 [(11)] に当てはまるものとして適切なものを 1 つ選べ．

ⓐ
```
1 count=[]
```

ⓑ
```
1 count=[0, 0, 0, 0, 0, 0]
```

ⓒ
```
1 count=[0, 0, 0, 0, 0]
```

ⓓ
```
1 count=0
```

(12) それぞれの目の出た比率を表示したい場合，コード中 [(12)] に当てはまるものとして適切なものを 1 つ選べ．

ⓐ
```
1 count[i]/n
```

ⓑ
```
1 count
```

ⓒ
```
1 count[i]
```

ⓓ
```
1 count/n
```

目 1 2 3 4 5 6
[0, 0, 0, 0, 0, 0]
↑
a[0]

(13) 以下の記述のうち, 正しいものを全て選べ.

ⓐ 配列の初めから順に 1, 2, ···, 6 の目とすると, 1 の目の出た比率は count[1] に格納されている.

ⓑ 配列の初めから順に 1, 2, ···, 6 の目とすると, 4 の目の出た比率は count[5] に格納されている.

ⓒ 配列の初めから順に 1, 2, ···, 6 の目とすると, 3 の目の出た比率は count[2] に格納されている.

ⓓ コードの 5 行目では, 0 より大きく, 5 以下の整数乱数を生成している.

ⓔ コードの 5 行目では, 0 以上 5 以下の整数乱数を生成している.

$\frac{1}{6} = 1.66\cdots$

(14) 以下の記述のうち, 正しいものを全て選べ.

ⓐ n を大きくしていくと, それぞれの出た目の比率は 1.66 ··· に近づいていく.

ⓑ n を大きくしていくと, それぞれの出た目の比率は 1.5 に近づいていく.

ⓒ n を大きくしていっても, それぞれの出た目の比率はある値に近づいていくことはない.

ⓓ 1 から 6 のそれぞれの出た目の比率は必ず, 全て異なる値になる.

(15) 実行後 10 と入力した場合に, 結果として表示され得るものを全て選べ. (ただし, 例えば 1.0 は 1 のように表示されるものとする.)

ⓐ
```
1 [0.1, 0.3, 0.2, 0.1, 0.2, 0.1]
```

ⓑ
```
1 [0.1, 0.3, 0.5, 0.1, 0.2, 0.1]
```
和1以上 X

ⓒ
```
1 [0.1, 0.3, 0.2, 0.1, 0.2]
```
52 X

ⓓ
```
1 [0.4, 0, 0.1, 0.1, 0.4, 0]
```

2 ゲーム制作 【20 点】

以下のルールで CPU との対戦ゲームを作る.

ルール

- 先攻後攻をランダムで決める.
- $n = 0$ とする.
- 自分の手番が来たら 1 ～ 3 の好きな数字を言い, n にその数値を加える. これを交互に繰り返す.
- n を 31 以上にした人の負け.

このゲームを行った例. (プレイヤー先手)

	手番	言った数	手番後の n
1	プレイヤー	3	3
2	CPU	2	5
3	プレイヤー	2	7
4	CPU	2	9
⋮	⋮	⋮	⋮

これを実際に実装したものが以下である.

```
1  import random
2  te=random.randint(1,2)
3  if te==1:
4   print("you are first move")
5  else:
6   print("you are second move")
7  [(4)]
8  while number<31:
9   if te%2==1:
10   n=int(input("how many ? = ",))
11  else:
12   n=random.randint(1,3)
13   print("CPU call", n)
14  [(5)]
15  print("Now...",number)
16  if number>=31 and te%2==1:
17   print("you lose")
18  [(6)]
19   print("you win")
20  [(7)]
```

以上を踏まえて, 次のページからの問いに答えよ.

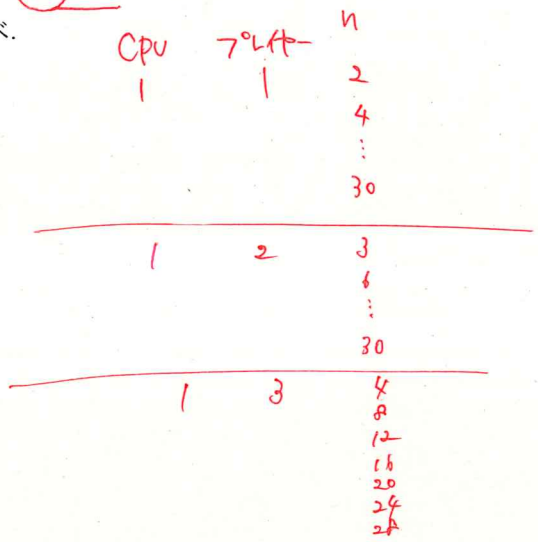
(1) 仮に, 常に同じ数値を言い続ける場合, CPU が先手で 1 と言い続ける場合にプレイヤーが勝つためにはどの数値を言い続ければいいか. 全て選べ.

ⓐ 1

ⓑ 2

ⓒ 3

ⓓ どの数値を言い続けても勝てない.



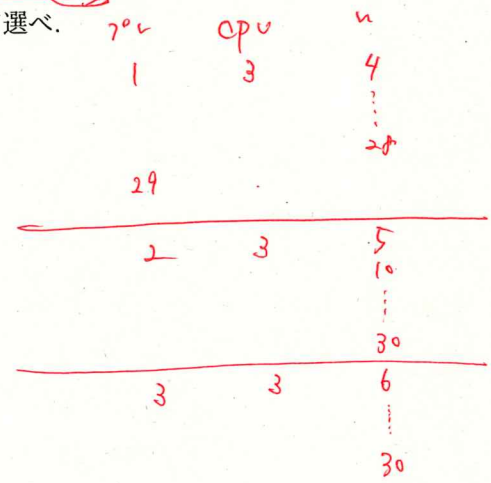
(2) 仮に, 常に同じ数値を言い続ける場合, CPU が後手で 3 と言い続ける場合にプレイヤーが勝つためにはどの数値を言い続ければいいか. 全て選べ.

ⓐ 1

ⓑ 2

ⓒ 3

ⓓ どの数値を言い続けても勝てない.



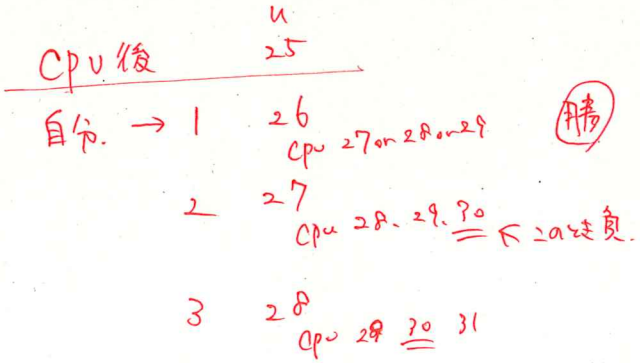
(3) CPU の手番終了後, $n = 25$ であった. プレイヤーの勝利を確定させるためにプレーヤーの言うべき数値として正しいものを全て選べ.

ⓐ 1

ⓑ 2

ⓒ 3

ⓓ 負け確定



(4) コード中 [(4)] に当てはまるものとして正しいものを 1 つ選べ.

ⓐ
```
1 number=0
```
ⓑ
```
1 number=1
```
ⓒ
```
1 te=0
```
ⓓ
```
1 te=1
```

(5) コード中 [(5)] に当てはまるものとして正しいものを 1 つ選べ.

ⓐ
```
1 number=number+n
```
ⓑ
```
1 number=n
```
ⓒ
```
1 n=n+number
```
ⓓ
```
1 n=number
```

(6) コード中 [(6)] に当てはまるものとして正しいものを 1 つ選べ.

ⓐ
```
1 elif number>=31 and te%2==0:
```
ⓑ
```
1 else:
```
ⓒ
```
1 elif number>31 and te%2==0:
```
ⓓ
```
1 elif number>=31 or te%2==0:
```

(7) コード中 [(7)] に当てはまるものとして正しいものを 1 つ選べ.

ⓐ
```
1 te=te+1
```
ⓑ
```
1 te=te+2
```
ⓒ
```
1 number=number+1
```
ⓓ
```
1 n=n+1
```

プレイヤーの手番について

(8) コードの 2 ～ 6 行目について、説明したものとして, 正しいものを全て選べ.

ⓐ ランダムで 1 か 2 を生成し, 1 が出たら先攻, 2 が出たら後攻となる.

ⓑ ランダムで 1 か 2 を生成し, 1 が出たら後攻, 2 が出たら先攻となる.

ⓒ ランダムで 1 以上 2 以下の実数を生成し, 1 が出たら先攻, それ以外の場合後攻となる.

ⓓ ランダムで 1 以上 2 以下の実数を生成し, 1 が出たら後攻, それ以外の場合先攻となる.

ⓔ ランダムで 0 か 1 を生成し, 0 が出たら先攻, 1 が出たら後攻となる.

ⓕ ランダムで 0 か 1 を生成し, 0 が出たら後攻, 1 が出たら先攻となる.

(9) コードの 7 行目以降について説明したものとして, 正しいものを全て選べ.

ⓐ「te」が 0 なのか 1 なのかで どちらの手番 かを判断している.

ⓑ「te」が 1 なのか 2 なのかで どちらの手番 かを判断している.

ⓒ「te」を 2 で割ったあまりが 1 か 0 かで どちらの手番 かを判断している.

ⓓ プレイヤーの言う数値を読み取る部分は 10 行目である.

ⓔ プレイヤーの言う数値を読み取る部分は 15 行目である.

(10) コードの 7 行目以降について説明したものとして, 正しいものを全て選べ.

ⓐ「number」の値が 31 未満であれば, 繰り返し続ける.

ⓑ 13 行目は, 現在の「number」の値を表示するためのコードである.

ⓒ この仕組みだと,「number」が 32 以上になることはない.

ⓓ この仕組みだと,「number」が 32 以上になることもある.

3 太郎さんと花子さんが, 授業で学んだソートアルゴリズムについて話している. 会話文を読んで, 各問いに答えよ. 【35 点】

太郎さん： 授業では選択ソートとバブルソートについて学んだね.

花子さん： バブルソートって面白い名前だよね. ちょっとバブルソートの確認をしてみようよ.

太郎さん： 何が面白いんだい. まあ, 確認してみようか. 下の手順書だと学んだよ.

手順書

配列 a を以下の手順で並べ替えていく.

I) 以下の操作を $i = 0$ から (配列長 -1) 回繰り返す.

i) 以下の操作を $j = 0$ から ((配列長) $- i - 1$) 回繰り返す.

A. j 番目の数 $> j + 1$ 番目の数であれば数の入れ替え

B. そうでなければ何もしない.

太郎さん： じゃあ, 下のような配列に対してどのように動いているか確認してみよう. 花子さん, やってみてよ.

```
1 L=[3, 9, 7, 1, 5, 2]
```

花子さん： 太郎くんにも伝わりやすいように, 表にしてみよう.

A) $i = 0$ のとき

	L[0]	L[1]	L[2]	L[3]	L[4]	L[5]
j = 0 終了後	3	9	7	1	5	2
j = 1 終了後	(2) 3	(3) 7	(4) 9	(5) 1	5	2
j = 2 終了後	(6) 3	(7) 7	(8) 1	(9) 9	(10) 5	2
⋮	⋮ 3	⋮ 7	⋮ 1	⋮ 5	⋮ 9	⋮ 2
j = (1) 4 終了後	3	7	1	5	2	9

B) $i = 1$ のとき

	L[0]	L[1]	L[2]	L[3]	L[4]	L[5]
j = 0 終了後	3	7	1	5	2	9
j = 1 終了後	(12) 3	(13) 1	(14) 7	(15) 5	2	9
j = 2 終了後	(16) 3	(17) 1	(18) 5	(19) 7	(20) 2	9
⋮	⋮	⋮	⋮	⋮	⋮	⋮
j = (11) 3 終了後	3	1	5	2	7	9

太郎さん： さすが, 花子さん. 正解だよ. そういえば, バブルソートの名前の由来って (22) らしいよ. 面白いね.

花子さん： ほんと面白いよね. しかも, 実装が簡単だしね. ただ, その分他のソートアルゴリズムよりも計算量が多いらしいね. とりあえず実装してみると, 以下のようになるよ.

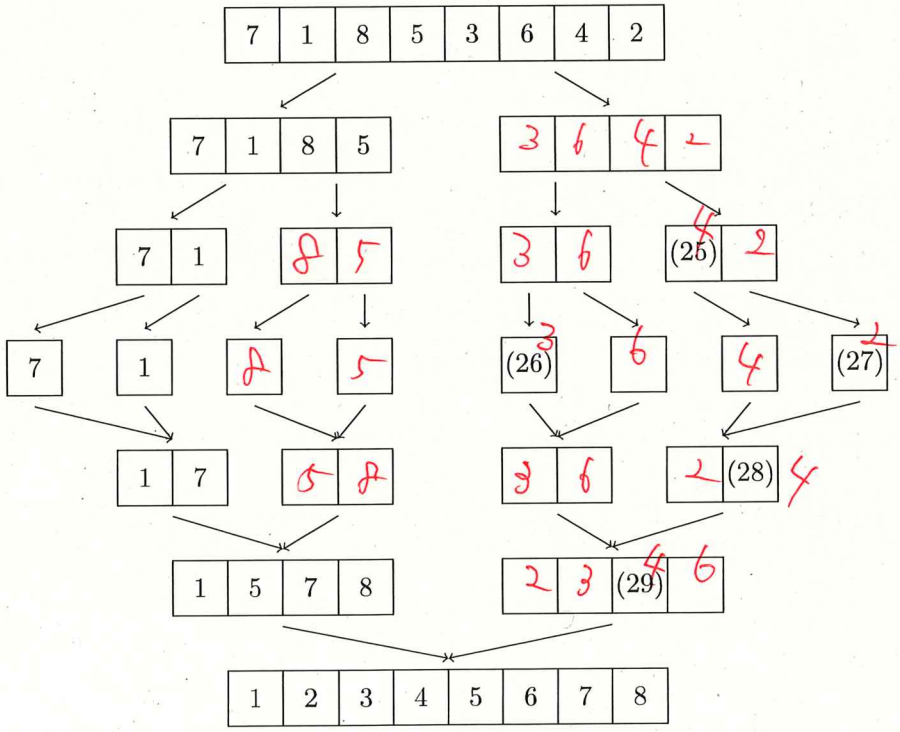
```
1 L=[3, 9, 7, 1, 5, 2]
2 for i in range(len(L)-1):
3  for j in range(len(L)-i-1):
4   if L[j]>L[j+1]:
5    tmp=L[j+1]
6    [(24)]
7    L[j]=tmp
8 print(L)
```

太郎さん： コード文中の「tmp」ってなんだい？

花子さん： これはね, (23)

太郎さん： 知ってたよ. 君が説明できるか確かめてみたんだ. じゃあ, 君はマージソートを知っているかい.

花子さん： 私は全知全能よ. 下に図で表してあげる.



花子さん： 配列を二分していき, それを結合する際に整列させる仕組みだよ.

太郎さん： 複数の配列を用意する必要がありそうだね.

表中の (1)~(20) に当てはまる数字を答えよ. 続いて, 以下の問いに答えよ.

(21) 実行結果として正しいものを 1 つ選べ.

ⓐ
```
1 [1, 2, 3, 5, 7, 9]
```

ⓑ
```
1 [9, 7, 5, 3, 2, 1]
```

ⓒ
```
1 [1, 2, 5, 3, 7, 9]
```

ⓓ
```
1 [9, 7, 5, 3, 1]
```

(22) 会話文中 (22) に当てはまるものとして最も適切なものを 1 つ選べ.

ⓐ 実際の泡を見た人が思いついたアルゴリズムだから

ⓑ 大きい値が, 泡のように配列の大きい方へ浮かんでくるから

ⓒ ビールの泡がバブルソートのような動きをするから

ⓓ 小さい値が, 泡のように配列の大きい方へ浮かんでくるから

(23) 会話文中 (23) に当てはまるものとして最も適切なものを 1 つ選べ.

ⓐ てんぷらの tmp だよ.

ⓑ 数値の入れ替え時, 上書きする際に, 元々そこにあった数値を格納しておくための変数だよ.

ⓒ 数値の入れ替え時, 上書きする際に, その上書きする数値を格納しておくための変数だよ.

ⓓ 数値の入れ替え時, 何回入れ替えたかを覚えておくための変数だよ.

(24) コード中 [(24)] に当てはまるものを選べ.

ⓐ
```
1 L[j]=L[j+1]
```

ⓑ
```
1 L[j+1]=L[j]
```

ⓒ
```
1 L[i+1]=L[i]
```

ⓓ
```
1 L[j+2]=L[j]
```

マージソートの他の部分を参考に, 図中の (25) ～ (29) に当てはまる数値を答え, 以下の問いに答えよ.

(30) マージソートの部分的な構成を行う. 2 つの配列 A, B を小さい順に結合して, 新しい配列 L を生成するプログラムとして正しいものを選べ. ただし, 配列 A, B については, 何かしらの方法で読み込んでいるものとする.

ⓐ

```
1  import math
2  L=[]
3  i=0;j=0
4  a=len(A);b=len(B)
5  while (i<a or j<b):
6   if A[i]<B[j]:
7    L.append(A[i])
8   else:
9    L.append(B[j])
10  if i>=len(A):
11   A.append(math.inf)
12  if j>=len(B):
13   B.append(math.inf)
14  i=i+1
15  j=j+1
16 print(L)
```

ok

X

ⓑ

```
1  import math
2  L=[]
3  i=0;j=0
4  a=len(A);b=len(B)
5  while (i<a or j<b):
6   if A[i]>=B[j]:
7    L.append(A[i])
8    i=i+1
9   else:
10   L.append(B[j])
11   j=j+1
12  if i>=len(A):
13   A.append(math.inf)
14  if j>=len(B):
15   B.append(math.inf)
16 print(L)
```

AかBどちらか上なるはず

X

ⓒ

```
1  import math
2  L=[]
3  i=0;j=0
4  a=len(A);b=len(B)
5  while (i<a or j<b):
6   if A[i]<B[j]:
7    L.append(A[i])
8   else:
9    L.append(B[j])
10  if i>=len(A):
11   A.append(math.inf)
12  if j>=len(B):
13   B.append(math.inf)
14 print(L)
```

ok

jとよ更新は?

ⓓ

```
1  import math
2  L=[]
3  i=0;j=0
4  a=len(A);b=len(B)
5  while (i<a or j<b):
6   if A[i]<B[j]:
7    L.append(A[i])
8    i=i+1
9   else:
10   L.append(B[j])
11   j=j+1
12  if i>=len(A):
13   A.append(math.inf)
14  if j>=len(B):
15   B.append(math.inf)
16 print(L)
```

ok

注) math.inf は, どんな値よりも大きい値 (∞) のこと.

4 日常生活への応用【15 点】

サマーウォーズにて, 健二が生年月日から曜日を暗算で求めていた. 求める手法として, 以下のような方法がある.

ツェラーの方法 (簡略化 ver.)

(1) 1 月, 2 月の場合, その前の年の 13 月, 14 月として扱う.
(2) A = 西暦の下 2 桁 とする.
(3) B = 西暦の下 2 桁を 4 で割った商
(4) 何月かにより, 以下の表に対応する値を C とする.

3 月	4 月	5 月	6 月	7 月	8 月	9 月	10 月	11 月	12 月	13 月	14 月
3	6	1	4	6	2	5	0	3	5	1	4

(5) D = 日付 とする.
(6) $X = A + B + C + D$ を求める. もし, 1900 年台の場合は 1 を足す.
(7) X を 7 で割ったあまりを求める. 以下の表に対応して曜日を求めることができる.

余り	1	2	3	4	5	6	0
曜日	日	月	火	水	木	金	土

これを以下のように実装した.

```
1  year=int(input("Year= "))
2  month=int(input("Month= "))
3  day=int(input("Day= "))
4  [(2)]
5   newyear=year-1
6   month=month+12
7  else:
8   newyear=year
9  [(3)]
10 b=(newyear%100)//4
11 c_list=[3,6,1,4,6,2,5,0,3,5,1,4]
12 [(4)]
13 d=day
14 x=a+b+c+d
15 [(5)]
16  x=x+1
17 w=x%7
18 week=["Saturday","Sunday","Monday","Tuesday","Wednesday","Thursday","Friday"]
19 print(day,"/",month,"/",year,"is...")
20 print(week[w])
```

次のページからの問いに答えよ.

2024年 13月25日

(1) 2025 年 1 月 25 日は何曜日か.

ⓐ 月曜日

ⓑ 水曜日

ⓒ 木曜日

ⓓ 土曜日

A=24
B=6
C=1
D=25
X=56
X÷7=8…0.
余り0.

(2) コード中 [(2)] に当てはまるものを選べ.

「1月、2月ならば」

ⓐ
```
1 if month!=1 and month!=2:
```

ⓑ
```
1 if month!=1 or month!=2:
```

ⓒ
```
1 if month==1 and month==2:
```

ⓓ
```
1 if month==1 or month==2:
```

(3) コード中 [(3)] に当てはまるものを選べ.

ⓐ
```
1 a=newyear
```

ⓑ
```
1 a=newyear%10
```

ⓒ
```
1 a=newyear%100
```

ⓓ
```
1 a=newyear%1000
```

下2ケタ

3月 4月
c = list[0] list[1]

(4) コード中 [(4)] に当てはまるものを選べ.

ⓐ
```
1 c=c_list[month-2]
```

ⓑ
```
1 c=c_list[month-3]
```

ⓒ
```
1 c=c_list[month-4]
```

ⓓ
```
1 c=c_list[month]
```

(5) コード中 [(5)] に当てはまるものを選べ. 1900年代なので newyear = 100 である事

ⓐ
```
1 if (newyear//100)==19:
```

ⓑ
```
1 if (newyear%100)==19:
```

ⓒ
```
1 elif (newyear//100)==19:
```

ⓓ
```
1 elif (newyear%100)==19:
```

(6) ツェラーの方法のうち, C の値を求める部分や対応する曜日を求める部分では, 条件分岐を用いた実装も考えることができるが, 配列で実装することの良い点を以下から 1 つ選べ.

ⓐ 配列を使う人はカッコよく見られる.
ⓑ コードが冗長にならないため, 見やすい.
ⓒ 計算時間が大幅に短縮できる.
ⓓ そもそも, 配列を用いて制作しなければならない.