情報入試問題提案 III

松浦 敏雄

大阪市立大学

「第3章：コンピュータとプログラミング」
に準じた問題案

松浦敏雄 (大阪市立大学)

第3章の学習内容

(ア) コンピュータの仕組み
- (1) コンピュータの仕組み
- (2) 計算誤差

(イ) アルゴリズムとプログラミング
- (1) 外部装置との接続
- (2) 基本的プログラム
- (3) 応用的プログラム
- (4) アルゴリズムの比較

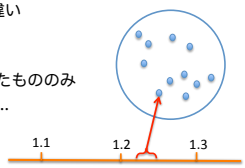
(ウ) モデル化とシミュレーション
- (1) モデル化とシミュレーション
- (2) 確定モデルと確率モデル
- (3) 自然現象のモデル化とシミュレーション

学習11 コンピュータの仕組み

＊コンピュータの構成，　演算の仕組み
- 主として知識問題か

＊AND・OR・NOT，真理値表
- 論理設計は面白い問題を作ることができるが…

＊計算誤差，プログラミングを使った計算誤差の確認
- [研修資料] 誤差に関しての記述は間違い
  魚の例はオーバーフローの話
  コンピュータが処理できるもの
  ⇒ 有限集合にマッピングされたもののみ
  整数の場合…　実数の場合…



学習12 学部装置との接続

＊計測・制御，センサ，アクチュエータ，計測・制御プログラム
- 授業で実習することは重要
  micro:bitの例が示されているが、入試では、特定のものに
  依存した問題は出せない ⇒ 抽象化？
- micro:bitもArduinoもワンボードマイコン
  独立したコンピュータ
  コンピュータと入出力装置との関係ではない。

学習13 基本的プログラム

＊アルゴリズム，プログラム，フローチャート，
順次・分岐・反復，変数
- 繰返しと条件分岐を組み合せて自力で簡単なプログラムが書けるか
- 二重ループのプログラムが書けるか
- さまざまな作題が可能

学習14 応用的プログラム

＊配列，乱数，関数，WebAPI
- 配列を使えるか
  研修資料では、リストと書かれているが実態は配列
  リスト特有の問題は出題しにくい
  プログラミングにどれだけ時間をかけられるか？
  ⇒ どこまで出題できるか？
  どの言語にも共通のもの　⇒ 配列まで？
- 関数を定義できるか、利用できるか
- WebAPI：研修資料ではこれを学習させたいと考えている？
  わずかなコードで高度なことができるのは面白いが、
  入試問題としては出題しにくい。

学習15 アルゴリズムの比較

＊探索アルゴリズムの比較，ソートアルゴリズムの比較
- アルゴリズムの計算量については、高度な問題はいくらでも作題可能だが、入試問題として適切なものはあまりなさそう。

7

学習16 確定モデルと確率モデル

＊確定モデルのシミュレーション，確率モデルのシミュレーション
- 確定モデル: 住宅ローンの返済計画、
- 確率モデル: 待行列（銀行、切符売り場、エレベータ、電車）
  交通渋滞予想、信号機の調整、…

8

学習17 自然現象のモデル化とシミュレーション

＊自然現象のモデル化とシミュレーション，モデルの妥当性の検討
- 天気予報、災害時の被害予想
- ダムの貯水量の予想
- 食物連鎖
  イノシシ ＞ ヘビ ＞ かえる ＞ 虫

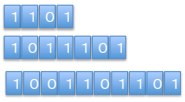
9

問題案

10

■[学習11-問題1] 10進2進変換
10進法で1桁の数すべてを2進法で表現するには、[あ] ビット必要である。
同様に、10進法で 2桁、および、3桁の数すべてを2進法で表現するには、
それぞれ、[い]ビット、および、[う]ビット必要となる。

$2^3 < 10 < 2^4$
$2^6 < 100 < 2^7$
$2^9 < 1000 < 2^{10}$



解答： [あ] 4 [い] 7 [う] 10

11

■[学習11-問題2] 多数決回路
0または1の値をとる3つの入力x, y, zに対して、2つ以上の入力が1のとき 1 を出力し、それ以外のときは 0 を出力する回路(多数決回路)を構成したい。

(a) 上記の回路の真理値表を完成せよ。

x	y	z	Q(出力)
0	0	0	[あ]
0	0	1	[い]
0	1	0	[う]
0	1	1	[え]
1	0	0	[お]
1	0	1	[か]
1	1	0	[き]
1	1	1	[く]

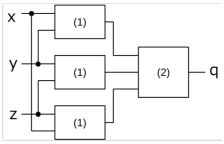


図1 多数決回路

解答例： [あ] 0 [い] 0 [う] 0 [え] 1 [お] 0
[か] 1 [き] 1 [く] 1

(b) 上記の回路を構成する方法はいろいろ考えられるが、図1 のように
4つの素子で回路を構成することもできる。
(1) および (2)の論理素子を名前を記せ。

解答例： (1) AND, (2) OR

12

■[学習13-問題1] 加算・乗算（繰返しと条件分岐の組み合わせ）
(問1)
　変数wに0以上の整数yを加えるプログラムを作れ。
ただし、演算として使えるのは「1を加える」か「1を引く」だけである。
したがって、wに1を加えることをy回繰り返すことになる。
　以下の選択肢から適切なものを選び、順に並べてプログラムを完成させよ。
なお、同じ選択肢を何回使ってもよいし、使わない選択肢があっても良い。

選択肢：
(A) z が0より大きい間くり返す
(B) ここまでがくり返しの範囲
(C) zにyを代入する
(D) wに1を加える
(E) zから1を引く

解答例：
(C) zにyを代入する
(A) zが0より大きい間くり返す
(D)　wに1を加える
(E)　zから1を引く
(B) ここまでがくり返しの範囲

13

■[学習13-問題1] 加算・乗算（繰返しと条件分岐の組み合わせ）（つづき）
(問2)
　0以上の整数をxとyに読み込み、その積を計算して表示するプログラムを作れ。
ただし、演算としてできるのは「1を加える」か「1を引く」だけである。
　以下の選択肢から適切なものを選び、順に並べてプログラムを完成させよ。
同じ選択肢を何回使ってもよいし、使わない選択肢があっても良い。

選択肢
(A) zが0より大きい間くり返す
(B) ここまでがくり返しの範囲
(C) zにyを代入する
(D) wに1を加える
(E) zから1を引く
(F) xに整数を入力する
(G) yに整数を入力する
(H) wを表示する
(I) xが0より大きい間くり返す
(J) ここまでがくり返しの範囲
(K) wに0を代入する
(L) xから1を引く

解答例：
(F) xに整数を入力する
(G) yに整数を入力する
(K) wに0を代入する
(I) xが0より大きい間くり返す
(C)　zにyを代入する
(A)　z が0より大きい間くり返す
(D)　wに1を加える
(E)　zから1を引く
(J)　ここまでがくり返しの範囲
(L)　xから1を引く
(J) ここまでがくり返しの範囲
(H) wを表示する

14

■[学習13-問題2] 条件を満たす数の和（繰返しと条件分岐の組み合わせ）
　1～1000までの整数のうち，3で割ると2余り，かつ，4で割ると1余る数
の合計を表示するプログラムを作成せよ.
　以下の短冊を用いてプログラムを完成させよ。各短冊を複数回用いても良い。
[　] の中には、適切な変数、定数、式を入れよ。

------ 短冊群 -----
[　] ← [　]
[　　] を表示する
もし [　　] ならば,
を実行する
[　　] の間,
を繰り返す

解答例：
S ← 0
n ← 1
n < 1000 の間,
　もし n % 3 = 2 かつ n % 4 = 1 ならば
　　S ← S + n
　を実行する
　n ← n + 1
を繰り返す
S を表示する

15

CBTのイメージ

1: 学習13-問題2: 1～1000までの整数のうち，3で割ると2余り，かつ，4で割ると1余る数の合計を表示するプログラムを作成せよ.

編集欄　実行画面を開く　戻す　進める　編集前に　全削除

1 変数 n, S
2 S ← 0
3 n ← 1
4 n < 1000 の間,
5 　もし n % 3 = 2 かつ n % 4 = 1 ならば
6 　　S ← S + n
7 　を実行する
8 　n ← n + 1
9 を繰り返す
10 S を表示する

選択肢欄　横幅：330　縦幅：360

変数
　←
　を表示する
もし [ ] ならば,
もし ～ そうでなければ
[ ] の間, 繰り返す

16

■[学習14-問題1] (リスト、乱数、関数分割)

　以下の関数が定義されているものとする。
　add(x,y)
　sub(x,y)
　mul(x,y)
　div(x,y)
それぞれx,yの加減乗除を計算するものである。
この関数を用いて、それを呼び出すことによって
以下の数式を計算する関数呼出しを書け。

　　a＋b×(c÷d-e)

解答例：
　add(a, mul(b, (sub(div(c,d),e))))

17

■[学習15-問題1] 最大値、最小値を求める最小の手数
　n個の要素を持つ配列A[0]～A[n-1]の各要素に予め相異なる整数が代入されているものとする。この中から、最大値と最小値を見つけたい。
　まず、配列Aの先頭から二つずつの組にわけ、各組内を小さい順に並べる。つぎに、各組の前の数(小さい方)の中から最小値を見つけ出し、各組の後ろの数(大きい方)の中から最大値を見つけ出せば良い。
(1) 上で説明したアルゴリズムを実現するプログラムを作成せよ。
(2) このアルゴリズムでの比較の回数はどれだけになるか答えよ。
以下の短冊を用いてプログラムを完成させよ。各短冊を複数回用いても良い。
[　] の中には、適切な変数、定数、式を入れよ。
------ 短冊群 -----
[　] ← [　]
もし [　　] ならば、
を実行する
[ ]を[ ]から[ ]まで[ ]ずつ増やしながら、
を繰り返す
「最大値：」 と [　] を表示する
「最小値：」 と [　] を表示する

解答例：
　k ← 0
　k を 0 から n - 2 まで2ずつ増やしながら,
　　もし A[k] > A[k+1] ならば
　　tmp ← A[k], A[k]←A[k+1], A[k+1]←tmp
　　を実行する
　を繰り返す
　min ← A[0],　max ← A[1]
　k を 2 から n - 2 まで2ずつ増やしながら,
　　もし　min > A[k] ならば
　　min ← A[k]
　　を実行する
　　もし　max < A[k+1] ならば
　　max ← A[k+1]
　　を実行する
　を繰り返す
「最大値：」 と max を表示する
「最小値：」 と min を表示する

18

■[学習16-問題1] 定積分

y = x*x と x = 1 と x軸で囲まれた部分の面積をモンテカルロ法によって求めるプログラムを書け。

以下の短冊を用いてプログラムを完成させよ。各短冊を複数回用いても良い。

[ ] の中には、適切な変数、定数、式を入れよ。

```
------ 短冊群 -----
変数 [     ]
[   ] ← [   ]
もし [     ] ならば、
を実行する
[   ]を1から[   ]まで1ずつ増やしながら,
を繰り返す
[   ] ← random([     ])
[     ] を表示する
[     ] を改行なしで表示する
```

解答例：

```
変数 x, y, k, N, count
count ← 0
N ← 1000
k を1からNまで1ずつ増やしながら,
    x ← random(1000) / 1000.0
    y ← random(1000) / 1000.0
    もし  y < x * x  ならば
        count ← count + 1
    を実行する
を繰り返す
"面積は"  を改行なしで表示する
count*1.0/N を表示する
```

19

■[学習16-問題2] すごろくのシミュレーション

次のコードは、40ますあるすごろくをサイコロ1個を繰り返し振っていくつであがるかのシミュレーションを行う関数 sugoroku1() と、それを繰り返し1000回実行して、あがるまでの回数の分布をリストに記録する関数 test() を含んでいる。

```
import random
def sugoroku():
    place = 0
    count = 0
    while place < 40:
        place = place + random.randint(1, 6)
        count = count + 1
    return count
def test():
    a = [0]
    for x in range(1000):
        n = sugoroku()
        if len(a) <= n:            # 配列にa[n]がまだなければ、
            a.extend([0 for x in range(n - len(a) + 1)])      # 必要な0の並びを追加
        a[n] = a[n] + 1
    print(a)
test()
```

上のプログラムを実行した出力の例を示す。

[0, 0, 0, 0, 0, 0, 0, 0, 13, 52, 146, 223, 210, 188, 97, 44, 22, 4, 1]

20

■[学習16-問題2] すごろくのシミュレーション　（つづき）

シミュレーションを次のように修正したい。

(a) 現在は40ます目に到達したら、ぴったりでなくてもあがれることになっている。ぴったりでない場合はあがれず、元の場所にとどまるように変更せよ。(sugoroku2)

(b) ぴったりでない場合は余ったぶんだけ出発点の方向に戻るように変更せよ。(sugoroku3)

(c) すごろくには「ふりだし」がつきものである。10、20、30のコマに止まったときは出発点に戻るように変更せよ。(sugoroku4)

関数sugoroku2〜sugoroku4を選択肢群から行を選んで適切な順に並べてプログラムを作成せよ。同じ選択肢を複数回用いてよい。使わない選択肢があっても良い。

選択肢群:

```
ア def sugoroku():
イ return count
ウ count = 0
エ place = 0
オ count = count + 1
カ place = place + 1
キ place = place + dice
ク place = place + random.randint(1, 6)
ケ place = 40 - (40 - place)
コ dice = randint(1, 6)
サ while place < 40:
シ if place + dice > 40:
ス if place > 40:
セ if place + dice < 40:
ソ if place < 40:
タ if place % 10 == 0:
チ if place % 10 != 0:
ツ (インデントを1レベル戻す)
```

(a) 解答例 (sugoroku2)

```
def sugoroku():
  place = 0
  count = 0
  while place < 40:
    dice = random.randint(1, 6)
    if place + dice <= 40:
      place = place + dice
    count = count + 1
  return count
```

21

■[学習16-問題2] すごろくのシミュレーション　（つづき）

(a) 解答例 (sugoroku3)

```
def sugoroku():
 place = 0
  count = 0
  while place < 40:
    dice = random.randint(1, 6)
    place = place + dice
    if place > 40:
      place = 40 - (place - 40)
    count = count + 1
  return count
```

(a) 解答例 (sugoroku4)

```
def sugoroku():
 place = 0
  count = 0
  while place < 40:
    place = place + random.randint(1, 6)
    if place < 40:
      if place % 10 == 0:
        place = 0
    count = count + 1
  return count
```

22

■[学習 11-問題 1] 10 進 2 進変換

10 進法で 1 桁の数すべてを 2 進法で表現するには、[あ] ビット必要である。同様に、10 進法で 2 桁、および、3 桁の数すべてを 2 進法で表現するには、それぞれ、[い]ビット、および、[う]ビット必要となる。

解答： [あ] 4 [い] 7 [う] 10

■[学習 11-問題 2] 多数決回路

0 または 1 の値をとる 3 つの入力 x, y, z に対して、2 つ以上の入力が 1 のとき 1 を出力し、それ以外のときは 0 を出力する回路(多数決回路)を構成したい。

(a) 上記の回路の真理値表を完成せよ。

x	y	z	Q(出力)
0	0	0	[あ]
0	0	1	[い]
0	1	0	[う]
0	1	1	[え]
1	0	0	[お]
1	0	1	[か]
1	1	0	[き]
1	1	1	[く]

解答例： [あ] 0 [い] 0 [う] 0 [え] 1 [お] 0
[か] 1 [き] 1 [く] 1

(b) 上記の回路を構成する方法はいろいろ考えられるが、図 1 のように 4 つの素子で回路を構成することもできる。

(1) および (2)の論理素子を名前を記せ。

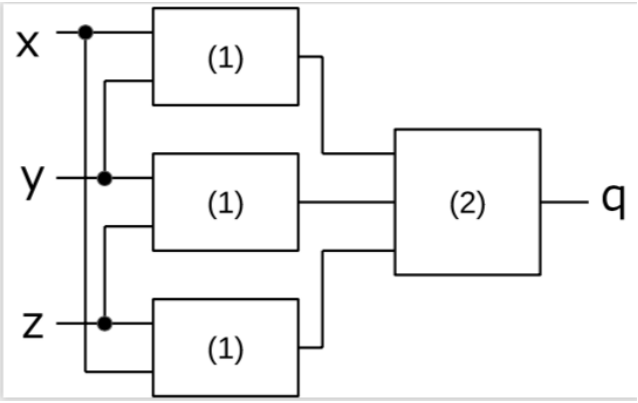


図 1 多数決を実現する回路

解答例： (1) AND, (2) OR

■[学習 13-問題 1] 加算・乗算（繰返しと条件分岐の組み合わせ）

(問 1) 変数 w に 0 以上の整数 y を加えるプログラムを作れ。ただし、演算として使えるのは「1 を加える」か「1 を引く」だけである。したがって、w に 1 を加えることを y 回繰り返すことになる。以下の解答群の選択肢を適切な順に並べてプログラムを完成させよ。なお、同じ選択肢を何回使ってもよいし、使わない選択肢があっても良い。

解答群

(A) z が 0 より大きい間くり返す
(B) ここまでがくり返しの範囲
(C) z に y を代入する
(D) w に 1 を加える
(E) z から 1 を引く

解答例：

(C) z に y を代入する
(A) z が 0 より大きい間くり返す
(D) w に 1 を加える
(E) z から 1 を引く
(B) ここまでがくり返しの範囲

(問 2)　0 以上の整数を x と y に読み込み、その積を計算して表示するプログラムを作れ。ただし、演算としてできるのは「1 を加える」か「1 を引く」だけである。以下の解答群の選択肢を適切な順に並べてプログラムを完成させよ。同じ選択肢を何回使ってもよいし、使わない選択肢があっても良い。

解答群

(A) z が 0 より大きい間くり返す
(B) ここまでがくり返しの範囲
(C) z に y を代入する
(D) w に 1 を加える
(E) z から 1 を引く
(F) x に整数を入力する
(G) y に整数を入力する
(H) w を表示する
(I) x が 0 より大きい間くり返す
(J) ここまでがくり返しの範囲
(K) w に 0 を代入する
(L) x から 1 を引く

解答例：

(F) x に整数を入力する
(G) y に整数を入力する
(K) w に 0 を代入する
(I) x が 0 より大きい間くり返す
(C)　　z に y を代入する
(A)　　z が 0 より大きい間くり返す
(D)　　　w に 1 を加える
(E)　　　z から 1 を引く
(J)　　ここまでがくり返しの範囲
(L)　　x から 1 を引く
(J) ここまでがくり返しの範囲
(H) w を表示する

※解説

問 1 のプログラムを利用して、2 重ループとして問 2 のプログラムを作る。各問の採点では正解例との一致、不一致だけでなく、正解例との違いに応じて部分点をつけることで、得点分布が極端にならないようにする。

■[学習 13-問題 2] 条件を満たす数の和（繰返しと条件分岐の組み合わせ）

1～1000 までの整数のうち，3 で割ると 2 余り，かつ，4 で割ると 1 余る数の合計を表示するプログラムを作成せよ．以下の短冊を用いてプログラムを完成させよ。各短冊を複数回用いても良い。[　] の中には、適切な変数、定数、式を入れよ。

------ 短冊群 -----

[　] ← [　　]

[　　　　] を表示する

もし [　　　]　ならば,

を実行する

[　　] の間,

を繰り返す

----- 解答欄 ------

解答例：

```
S ← 0
n ← 1
n < 1000 の間,
    もし n % 3 = 2 かつ n % 4 = 1 ならば
        S ← S + n
    を実行する
    n ← n + 1
を繰り返す
S を表示する
```

■[学習 14-問題 1] (リスト、乱数、関数分割)

以下の関数が定義されているものとする。

add(x,y)
sub(x,y)
mul(x,y)
div(x,y)

それぞれ x,y の加減乗除を計算するものである。
この関数を用いて、それを呼び出すことによって
以下の数式を計算する関数呼出しを書け。

a＋b×(c÷d-e)

解答例：

add(a, mul(b, (sub(div(c,d),e))))

■[学習 15-問題 1] 最大値、最小値を求める最小の手数

n 個の要素を持つ配列 A[0]～A[n-1]の各要素に予め相異なる整数が代入されているものとする。この中から、最大値と最小値を見つけたい。

2 つの数を比較したとき、小さい方の数は、最大値の候補にはなり得ないし、大きい方の数は、最小値の候補になり得ない。そこで、まず、配列 A の先頭から二つずつを組みとし、それぞれの組み内で値を比較し、大きいと判断した数のグループと小さいと判断した数のグループに分ける。大きい方のグループの中から最大値を見つけ出し、小さい方の中から最小値を見つけ出せば良い。

(1) 上で説明したアルゴルズムを実現するプログラムを作成せよ。
(2) このアルゴリズムでの比較の回数はどれだけになるか答えよ。

以下の短冊を用いてプログラムを完成させよ。各短冊を複数回用いても良い。
[　] の中には、適切な変数、定数、式を入れよ。

------ 短冊群 ----- ----- 解答欄 ------

[ ] ← [ ]

もし [ ] ならば、

を実行する

[ ] の間、

を繰り返す

「最大値：」 と [ ] を表示する

「最小値：」 と [ ] を表示する

解答例：

k ← 0

k を 0 から n - 2 まで 2 ずつ増やしながら,

もし A[k] > A[k+1] ならば

tmp ← A[k], A[k] ← A[k+1], A[k+1] ← tmp

を実行する

を繰り返す

min ← A[0], max ← A[1]

k を 2 から n - 2 まで 2 ずつ増やしながら,

もし min > A[k] ならば

min ← A[k]

を実行する

もし max < A[k+1] ならば

max ← A[k+1]

を実行する

を繰り返す

「最大値：」 と max を表示する

「最小値：」 と min を表示する

■[学習 16-問題 1] 定積分

y = x*x と x = 1 と x 軸で囲まれた部分の面積をモンテカルロ法によって求めるプログラムを書け。以下の短冊を用いてプログラムを完成させよ。各短冊を複数回用いても良い。[ ] の中には、適切な変数、定数、式を入れよ。

------ 短冊群 ----- ----- 解答欄 ------

変数 [ ]

[ ] ← [ ]

もし [ ] ならば、

を実行する

[ ] を 1 から [ ] まで 1 ずつ増やしながら,

を繰り返す

[ ] ← random([ ])

[ ] を表示する

[ ] を改行なしで表示する

解答例 :

```
変数 x, y, k, N, count
count ← 0
N ← 1000
k を 1 から N まで 1 ずつ増やしながら,
    x ← random(1000) / 1000.0
    y ← random(1000) / 1000.0
    もし y < x * x ならば
        count ← count + 1
    を実行する
を繰り返す
"面積は" を改行なしで表示する
count*1.0/N を表示する
```

■[学習 16-問題 2] すごろくのシミュレーション

以下の説明を読み、設問に答えよ。

次のコードは、40 ますあるすごろくをサイコロ 1 個を繰り返し振っていくつであがるかのシミュレーションを行う関数 sugoroku1() と、それを繰り返し 1000 回実行して、あがるまでの回数の分布をリストに記録する関数 test() を含んでいる。

```
import random

def sugoroku():
    place = 0
    count = 0
    while place < 40:
        place = place + random.randint(1, 6)
        count = count + 1
    return count

def test():
    a = [0]
    for x in range(1000):
        n = sugoroku()
        if len(a) <= n:                    # 配列に a[n]がまだなければ、
            a.extend([0 for x in range(n - len(a) + 1)]) # 必要な 0 の並びを追加
        a[n] = a[n] + 1
    print(a)

test()
```

上のプログラムを実行した出力の例を示す。

[0, 0, 0, 0, 0, 0, 0, 0, 13, 52, 146, 223, 210, 188, 97, 44, 22, 4, 1]

シミュレーションを次のように修正したい。

(a) 現在は 40 ます目に到達したら、サイコロがぴったりでなくてもあがれることになっている。ぴったりでない場合はあがれず、元の場所にとどまるように変更せよ。(sugoroku2)

(b) 上と類似しているが、ぴったりでない場合は余ったぶんだけ出発点の方向に戻るように変更せよ。(sugoroku3)

(c) すごろくには「ふりだし」がつきものである。10、20、30 のコマに止まったときは出発点に戻るように変更せよ。(sugoroku4)

それぞれに対応する関数 sugoroku2～sugoroku4 を選択肢群から行を選んで並べ構成せよ。同じ選択肢を複数回用いてよい。

選択肢群:

ア def sugoroku():
イ return count
ウ count = 0
エ place = 0
オ count = count + 1
カ place = place + 1
キ place = place + dice
ク place = place + random.randint(1, 6)
ケ place = 40 - (40 - place)
コ dice = randint(1, 6)
サ while place < 40:
シ if place + dice > 40:
ス if place > 40:
セ if place + dice < 40:
ソ if place < 40:
タ if place % 10 == 0:
チ if place % 10 != 0:
ツ (インデントを 1 レベル戻す)

解答例：

```
def sugoroku2():
    place = 0
    count = 0
    while place < 40:
        dice = random.randint(1, 6)
        if place + dice <= 40:
            place = place + dice
        count = count + 1
    return count

def sugoroku3():
    place = 0
    count = 0
    while place < 40:
        dice = random.randint(1, 6)
        place = place + dice
        if place > 40:
            place = 40 - (place - 40)
        count = count + 1
    return count

def sugoroku4():
    place = 0
    count = 0
    while place < 40:
        place = place + random.randint(1, 6)
        if place < 40:
            if place % 10 == 0:
                place = 0
        count = count + 1
    return count
```