情報

試験日			11/22
理系科目	大　問	Ⅰ	18点
		Ⅱ	22点
		Ⅲ	30点
		Ⅳ	30点
	合計		100点

解答方法は全試験日、すべてマーク式で試験時間は 45 分です。

出題内容・出題意図

大問の構成は、〔Ⅰ〕基礎的な事項に関する理解度〔Ⅱ〕システム設計で必要になる応用力〔Ⅲ〕プログラミングの能力（提示されたデータ処理を実現する処理を推定する）〔Ⅳ〕データ分析の能力（シミュレーションの結果から全体を制御する設定値を推定する）について、それぞれ問うものでした。

〔Ⅱ〕〔Ⅲ〕〔Ⅳ〕はセンサーデータの収集システム、手持ち楽曲のシャッフル再生、じゃんけんシミュレーションを題材に取っています。つまり身近で現実的な場面（条件設定）に対して、「問題を整理・把握」し「必要な対処」をするための思考力を問うことが全体を通した出題の骨になっています。

入試対策（アドバイス）

「情報Ⅰ」の教科書はよくできています。ただ、各事項は短く取り上げられているものが多いので、それらの役割や働きをしっかり理解することが、〔Ⅰ〕と〔Ⅱ〕に対する有効な対策になります。

〔Ⅲ〕は「やりたいことを実現するアルゴリズム」を検討し、それを「具体的なデータ構造と手続き（プログラムコード）」として表現できることを求めています。そうしたことを意識しながら、実際にプログラムを書く経験を増やすことが有効です。〔Ⅳ〕は「シミュレーション実験の設定と分析手続き」を的確に把握し、「得られた結果の意味を定量的に読み解く」能力を問うています。身近な疑問をモデル化し、計算機実験を行って得た結果の要点をグラフ等に表して他者に説明する練習を、日頃から行うと良いでしょう。

本学の「情報」では、プログラミングに関する問題・解答において独自の記法を用いてプログラムを記述します。以下はその記法の例示です。

プログラム表記の例示

本試験の設問、および解答においてプログラムを記述するために用いる記法について説明する。

本表記方法は既存のプログラミング言語とは異なる疑似言語によるものである。

演算子

数値に対して算術演算を行うことができる。演算子は以下の通り。なお、＊、/ と % の計算が + や − よりも先に行われる。

+	加算(足し算)	−	減算(引き算)
＊	乗算(掛け算)	/	除算(割り算)
%	剰余算(割り算の余り)。(例: 7%3 は 1)		

数値同士、文字列同士を比較できる。結果は真または偽である。

A == B	AとBの値が等しい。
A != B	AとBの値が等しくない。

数値の大小を比較できる。結果は真または偽である。

A <= B	AがB以下	A < B	AがBより小さい
A >= B	AがB以上	A > B	AがBより大きい

複数の条件を組み合わせたり、ある条件を否定するために以下の論理演算子を用いることができる。

条件1 **AND** 条件2　　条件1 **OR** 条件2　　**!**条件

算術演算と比較演算では、算術演算が先に計算される。また、ORよりもAND、ANDよりも！が優先される。式の中で ( ) を使い、計算の順序を示すことができる。

文

`count = count + 1`	変数に式の値を代入する。左の例は変数の値を1増やす。
for i = 0 **to** N−1 table[i] = 0 **end**	変数の値を0から(N−1)まで1ずつ増やして繰り返す。
while a[n] == b[n] n = n + 1 **end**	条件が成り立つ間、繰り返しを実行する。

while a[n] == b[n]
  if a[n] == 0
  break
  end
  n = n + 1
end
	if文の条件が成り立てば、while文の繰り返しを打ち切る(同様に、for文の繰り返しも打ち切ることができる)。

if kekka != 0 print("当選") **end**	条件が成り立つときに実行する。

if kion < 30
  print("実施")
else
  print("中止")
end
	条件が成り立つかどうかで、実行することを変える。

if tokuten >= 80
  print("優秀")
elif tokuten >= 60
  print("合格")
else
  print("追試")
end
	複数の条件を順番に調べて実行することを変える。

return gokei	関数の実行を終了して値を呼び出し側へ戻す。
return	関数の実行を終了する(戻り値の必要ない場合)。
print("答:", w+1)	値や文字列を表示し、改行する。

変数と配列： 変数(または配列)は、関数の内部で宣言した場合、その関数でのみ利用可能な変数(ローカル変数)となり、関数の外部で宣言するとどこからでも利用可能な変数(グローバル変数)となる。

var i, j **var** total = 0	変数を宣言する。宣言と同時に初期値を指定できるが、指定がない変数の初期値は不定(意味のない値)である。
var table[10]	配列を宣言する。上の例ではtable[0]からtable[9]まで10個の要素が用意される。
var s[] = {−1, 0, 1}	初期値を指定した配列を宣言する。上の例ではs[0]に−1、s[1]に0、s[2]に1が設定される。

関数

func add (a, b)
  var sum
  sum = a + b
  return sum
end
	関数(サブルーチン)を定義する。戻り値のある関数は return文で値を指定する。左の関数は次の例のように呼び出せる。
	total = add(m, 500)

func show (t) print("答:", t) **end**	戻り値のない関数は return文を省略できる。左の関数は次の例のように呼び出せる。 show(n ＊ 2)

コメント： プログラムの記述中に # が現れた場合、そこから行末までの文字列はコメント(注釈)とみなし、実行されない。ただし、文字列の中に現れた # はコメントとしては扱わない。

プログラム例1： 買い物の合計額を計算する関数の定義

買い物の項目数が n、配列 price, amount に買った品物の単価と個数が格納されているとする。合計金額が 2000円以上なら送料が無料になる。変数deliveryはグローバル変数である。

```
var delivery = 500  # 送料

func shopping(price, amount, n)
  var pay = 0
  var i

  for i = 0 to n − 1
    pay = pay + price[i] ＊ amount[i]
  end
  if pay >= 2000    # 合計が2000円以上
    return pay
  end
  return pay + delivery
end
```

プログラム例2： 三角形の種類を調べる関数の定義

```
func triangle(x, y, z)
# 引数は3辺の長さ。ただし、x ≧ y ≧ z とする。
  if x >= y + z
    print("三角形ではない")
  elif x == y OR y == z
    if x == z
      print("正三角形")
    else
      print("二等辺三角形")
    end
  else
    print("三角形")
  end
end
```

〔 Ⅰ 〕 以下の文章を読んで、設問に答えなさい。

空欄 ア ～ カ に入れるのに最も適切な語をそれぞれの解答群から1つずつ選び、その番号を解答欄にマークせよ。

1. 情報セキュリティとは、情報の機密性、完全性、 ア 性の 3 つを確保することと定義されている。機密性とは、許可された人だけが情報にアクセスできることで、完全性とは、情報の欠損や イ がないことを保証することで、 ア 性とは、情報を使いたいときにいつでも使える状態を保つことである。

ア ・ イ の解答群:

【 (0) 危険　(1) 安全　(2) 脆弱　(3) 可用　(4) 絶対
(5) 独立　(6) 改ざん　(7) 選択　(8) 設定　(9) 完成 】

2. 一般的なコンピュータの CPU(中央処理装置)の内部には、数個から数十個の ウ と呼ばれる何らかのデータを一時的に記憶するものが備わっている。例えば、次の命令の取り出し番地を指定する エ も ウ の一種である。

ウ ・ エ の解答群:

【 (0) 主記憶装置　(1) 外部記憶装置　(2) SSD　(3) HDD
(4) キャッシュメモリ　(5) インデックス　(6) ヒープ　(7) スタックポインタ
(8) プログラムカウンタ　(9) レジスタ 】

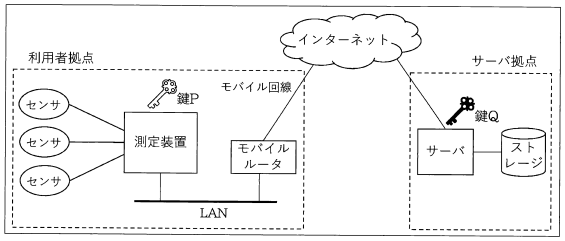
3. Web ページを作るための記述言語として オ があり、“<”と“>”で挟まれたタグで文字列を挟むことにより、Web ページの段落やハイパーリンクなどの設定を行う。また、Web ページのデザインや文字のフォントや色の指定を行うものとして カ が規定されている。これらにより、文書の構造と体裁を分離することが可能になっている。

オ ・ カ の解答群:

【 (0) JSON　(1) CSS　(2) URL　(3) W3C　(4) HTML　(5) Python
(6) サイトマップ　(7) タイトル　(8) ヘッダー　(9) フッター 】

〔 Ⅱ 〕 以下の文章を読んで、設問(A)～(C)に答えなさい。

遠隔地の環境を調査するための図Ⅱ-1 に示すような測定システムを考える。利用者拠点では、3 個のセンサがつながる測定装置が LAN に接続されており、モバイル(携帯電話)回線を利用したモバイルルータを経由してインターネットにつながっている。測定装置はセンサからの測定データを取得し、それを暗号化し、インターネットを経由してサーバに対して送信する。また、サーバ拠点に置かれたサーバは、受信したデータを復号して分析を行い、サーバにつながるストレージに分析結果を蓄積するようになっている。なお、モバイル回線を流れるデータの量は、センサの数に比例するものとし、1MB(メガバイト)は 1000KB(キロバイト)とする。



図Ⅱ-1. 測定システム全体の構成

設問(A)

次の空欄に入れる適切な語を、解答群から 1 つずつ選び、その番号を解答欄にマークせよ。

測定装置でデータを暗号化するための鍵を鍵 P とし、測定装置内の設定ファイルに保存する。また、サーバで受信データを復号するための鍵を鍵 Q とし、サーバ内の設定ファイルに保存する。このシステムでは、鍵 P と鍵 Q に別々の鍵を用いる ア 方式を使用する。このとき、測定装置側の鍵 P は利用者に見られても問題が起こらないようにするため、鍵 P としては イ を使い、サーバ側の鍵 Q としては ウ を使う。

ア ～ ウ の解答群:

【 (0) 共通鍵暗号　(1) 換字式暗号　(2) シーザー暗号　(3) 公開鍵暗号
(4) 共通鍵　(5) 秘密鍵　(6) 公開鍵　(7) 平文　(8) 暗号文 】

設問(B)

次の空欄に入れる数値を解答欄にマークせよ。

センサが 3 個のとき、モバイル回線に流れるデータの量は 1 分あたり 20KB で一定であった。このとき、モバイル回線に流れる 1ヶ月(30 日)あたりのデータの合計は エオカキ MB になると考えられる。なお、値が 4 桁未満のときは上位の桁に 0 を入れること。

設問(C)

次の空欄に入れる数値を解答欄にマークせよ。

今後センサを増設することで、モバイル回線を流れるデータ量が増加する見込みである。ただし、モバイル回線を流れるデータの量は、1 日あたり 100MB(メガバイト)以下にしたいという要望がある。このとき、設問(B)で示した状況(3 個のセンサからのデータの量は 1 分あたり 20KB)のもとでは、センサの数を最大で クケ 個とすることができる。なお、値が 2 桁未満のときは上位の桁に 0 を入れること。

〔 Ⅲ 〕 以下の文章を読んで、設問(A)～(D)に答えなさい。
この問題は、別紙に示す「プログラム表記の例示」にしたがってプログラムを記述する。

Kさんは自分が持つたくさんの音楽を楽しむためのプログラムを作ることにした。Kさんはいつも同じ順番で曲が再生されるより、その時々で曲の再生順が変わる方が楽しいと思い、乱数を用いて次に再生する曲を決めようと考えた。

設問(A)

プログラムⅢ-1 は自分の持つ曲の数が 100 曲のときに乱数に基づいて次に再生する曲の番号を 1 から 100 までの曲番号として算出するものである。ただしプログラム中で用いる関数 random は 0 から 65535 までの擬似乱数の整数値を返す関数である。

プログラムⅢ-1 中の空欄 ア と イ に入れるのに最も適切な値をそれぞれの解答群のうちから1つずつ選び、その番号を解答欄にマークせよ。

```
1: func selectMusicNumber()
2:     var randomNumber, numberSong
3:     randomNumber = random()
4:     numberSong = [ア] + randomNumber % [イ]
5:     return numberSong
6: end
```

プログラムⅢ-1. 乱数に基づき曲番号を選択する

ア の解答群:

【 (0) −2　(1) −1　(2) 0　(3) 1　(4) 2 】

イ の解答群:

【 (0) −1　(1) 1　(2) 99　(3) 100　(4) 101 】

設問(B)

Kさんはプログラム3-1の関数selectMusicNumberを用いて、自分の持つ100曲からランダムに曲を再生しようと考えた。しかし、乱数に基づいてそのまま曲を再生すると、続けて同じ曲がかかる場合がありそうだと気がついた。そこでまず、続けて同じ曲（1つ前と同じ曲）になる場合がどれくらいあるかを調べようとプログラム3-2を作成した。

プログラム3-2は1000回ランダムに曲番号を求め、それを出力する。さらに1つ前と同じ曲の番号である回数をカウントするプログラムである。

プログラム3-2中の空欄 ウ と エ に入れるのに最も適切な記述をそれぞれの解答群のうちから1つずつ選び、その番号を解答欄にマークせよ。

```
 1: var numberSong    # 次の曲の候補の曲番号
 2: var lastSong = 0 # 1つ前の曲の曲番号
 3: var count = 0     # 1つ前の曲と同じ曲の番号である回数
 4: for i = 1 to 1000
 5:     numberSong = selectMusicNumber()
 6:     print(numberSong)                  # 曲番号を表示(改行あり)
 7:     if numberSong  [ ウ ]  lastSong
 8:         count = count + 1 # 1つ前の曲と同じ曲の場合
 9:     else
10:         [      エ      ]
11:     end
12: end
13: print(count)
```

プログラム3-2．曲番号と回数を出力する

ウ の解答群:

【 (0) ==　(1) !=　(2) >=　(3) <=　(4) >　(5) <　】

エ の解答群:

【 (0) lastSong = lastSong + 1　(1) numberSong = numberSong + 1
(2) numberSong = lastSong　(3) numberSong = lastSong + 1
(4) lastSong = numberSong　(5) lastSong = numberSong + 1　】

設問(C)

Kさんはプログラム3-2 のプログラムを実行し、出力結果をよく見ると、思ったより 1 つ前と同じ曲がかかっていたことがわかった。また 1 つ前と同じ曲だけでなく、2，3 曲前に聴いた曲も存在していた。そのため、1 つ前と同じ曲だけでなく、さっき聴いたと感じる曲が何度もかかりそうに思えた。そこでKさんは 1 つ前と同じ曲だけでなく、10 曲前まで（直近の 10 曲と呼ぶ）と同じ曲がかからないようにして、再生順を 1000 曲分決めるプログラムを作成した。

具体的には、自分の持つ 100 曲それぞれについて、「最後にかかったのが何曲目だったか」を再生番号として記録する。その記録に基づいて、次にかけようとしている曲が直近の 10 曲分と同じかどうかを判定する。

プログラム3-3は直近の10曲と同じ曲がかからない合計1000曲の再生順を出力するプログラムである。プログラム3-3では、手持ちの曲100曲分の長さの配列songsを用いて、それぞれの曲の再生番号を記録し、次にかけようとしている曲が最後にかかったのが直近の10曲以内かを判定する。songs[0]には曲番号1の再生番号が記録され、それ以降の配列要素には曲番号順に再生番号が記録されるものとする。関数selectMusicNumberで選んだ曲が直近10曲以内でかかった曲でないと判断したら、変数flagを1としてその曲を次の曲とし、その再生番号と曲番号を出力する。

プログラム3-3中の空欄 オ と カ に入れるのに最も適切な記述をそれぞれの解答群のうちから1つずつ選び、その番号を解答欄にマークせよ。ただし2箇所の オ には同じ記述が入る。

オ の解答群:

【 (0) listNum　(1) numberSong　(2) songs[listNum]　(3) songs[numberSong]
(4) songs[listNum - 1]　(5) songs[numberSong - 1]
(6) songs[listNum + 1]　(7) songs[numberSong + 1]　】

カ の解答群:

【 (0) listNum >= range　(1) prevSong - range >= listNum
(2) listNum >= prevSong　(3) listNum - prevSong >= range
(4) prevSong >= range　(5) listNum >= prevSong - range　】

```
 1: var numberSong      # 次の曲の候補の曲番号
 2: var songs[100]      # 各曲の再生番号を記録する
 3: var range = 10      # 直近にかかったとみなす曲数
 4: var i, flag = 0
 5: var listNum = 0     # いまかかっている曲の再生番号
 6: var prevSong        # 次の曲の候補の再生番号の記録
 7: for i = 0 to 100 - 1
 8:     songs[ i ] = -1    # 再生番号を-1で初期化
 9: end
10:
11: while listNum < 1000
12:     numberSong = selectMusicNumber() # 次の曲の候補の曲番号
13:     prevSong =  [ オ ]
14:     # 次の曲の候補の判定処理
15:     if listNum < range                # range分再生していないことの判定
16:         if prevSong == -1
17:             flag = 1
18:         end
19:     elif  [ カ ]                      # 直近でかかった曲でないことの判定
20:         flag = 1
21:     end
22:
23:     if flag == 1                      # 選んだ曲を次の曲とする処理
24:         flag = 0
25:         [ オ ] = listNum
26:         listNum = listNum + 1
27:         print(listNum, ",", numberSong) # 再生番号と次の曲の曲番号を
28:                                         # 出力(改行あり)
29:     end
30: end
```

プログラム3-3．直近の 10 曲と同じ曲がかからない再生順を出力する

設問(D)

Kさんはプログラム3-3のプログラムの15行目から21行目で行っている次の候補の曲の再生番号に基づいて直近の10曲以内かを判定する処理が複雑すぎることが気になった。よく見直したところ、配列songsの初期値を -1 でなく、ある記述に設定するとプログラム3-4に示すように15行目から17行目の処理に簡単化できることに気がついた。プログラム3-4中の空欄 キ に入れるのに最も適切な記述を解答群のうちから1つ選び、その番号を解答欄にマークせよ。

```
 1: var numberSong      # 次の曲の候補の曲番号
 2: var songs[100]      # 各曲の再生番号を記録する
 3: var range = 10      # 直近にかかったとみなす曲数
 4: var i, flag = 0
 5: var listNum = 0     # いまかかっている曲の再生番号
 6: var prevSong        # 次の曲の候補の再生番号の記録
 7: for i = 0 to 100 - 1
 8:     songs[ i ] =  [ キ ]   # 再生番号を初期化
 9: end
10:
11: while listNum < 1000
12:     numberSong = selectMusicNumber() # 次の曲の候補の曲番号
13:     prevSong =  [ オ ]
14:     # 次の曲の候補の判定処理
15:     if  [ カ ]                        # 直近でかかった曲でないことの判定
16:         flag = 1
17:     end
18:
19:     if flag == 1                      # 選んだ曲を次の曲とする処理
20:         flag = 0
21:         [ オ ] = listNum
22:         listNum = listNum + 1
23:         print(listNum, ",", numberSong) # 再生番号と次の曲の曲番号を
24:                                         # 出力(改行あり)
25:     end
26: end
```

プログラム3-4．プログラム3-3 の初期値を変更し判定処理を簡単化したもの

キ の解答群:

【 (0) (-1) * range　(1) (-1) * range + 1　(2) -2　(3) 0　(4) 1　(5) 2
(6) range - 1　(7) range　(8) range + 1　】

〔 Ⅳ 〕 以下の文章を読んで、設問(A)と設問(B)に答えなさい。

じゃんけんの対戦のシミュレーションについて考える。じゃんけんの基本的なルールは以下である。

a) グー、チョキ、パーの 3 つの手がある。
b) 手の強さとして、グーはチョキに勝ち、チョキはパーに勝ち、パーはグーに勝つ。
c) 2 人の対戦者が同時にそれぞれの手を 1 つ出す。2 人が異なる手を出した場合はルール b)に従って勝者と敗者が決まり、2 人が同じ手を出した場合は、勝敗が決まらない「あいこ」となる。

上記のルールに従って対戦者 1 と対戦者 2 がじゃんけんを繰り返す対戦シミュレーションのプログラムを作成した。プログラムでは、各対戦者が各手を出す確率の値を対戦者ごとに独立に設定できるようにした。つまり、各対戦者がグー、チョキ、パーを完全にランダムに出すのではなく、対戦者ごとに出しやすい手や出しにくい手がある状況のシミュレーションを行えるようにした。ただし、対戦者ごとに、3 つの手を出す確率の値の和は 1.00 になるように設定している。

このプログラムを用いて、各対戦者が各手を出す確率をある値に設定したまま変化させずに合計 1000 回の対戦シミュレーションを行った。下の表Ⅳ-1 は、その最初の 10 回の結果をまとめたものである。以降では、連続して行ったじゃんけん対戦シミュレーションにおける 1 回ごとの対戦を「対戦回」と呼び、1 回目から数えて何回目の対戦回であるかとともに表す。たとえば、表Ⅳ-1 の 1 回目の対戦回では対戦者 1 がグーを出し、対戦者 2 がパーを出して、対戦者 2 が勝った。9 回目の対戦回では、両対戦者ともグーを出し、あいことなった。さらに、ある条件にあてはまる対戦回が 1 回目から数えて各対戦回までに(その対戦回も含めて)合計何回生じたかを、「各対戦回までのその条件の総回数」のように表現する。たとえば、表Ⅳ-1 において「10 回目の対戦回までの対戦者 1 がチョキを出した総回数」は 4 回である。

表Ⅳ-1. 10 回目の対戦回までのじゃんけん対戦シミュレーションの結果

対戦回(回目)	1	2	3	4	5	6	7	8	9	10
対戦者 1 の手	グー	グー	パー	チョキ	グー	チョキ	グー	チョキ	グー	チョキ
対戦者 2 の手	パー	チョキ	パー	グー	パー	チョキ	パー	チョキ	グー	パー
勝者	2	1	-	2	2	-	2	-	-	1

「勝者」の 1 は対戦者 1 が勝ったことを表し、2 は対戦者 2 が勝ったことを表す。 - はあいこを表す。

プログラムでは、各対戦回で各対戦者が出した手や対戦者ごとの勝敗の結果などの対戦結果に関する様々な情報を、対戦回ごとに記録するようにした。この情報を用いて、今回の 1000 回の対戦シミュレーションにおいてグー、チョキ、パーの各手が勝った対戦回の総回数を、各対戦回までごとにそれぞれ調べた。その結果を、次ページの図Ⅳ-1 に示すグラフに表した。ただし、「各手が勝った対戦回の総回数」には、対戦者 1 が勝った対戦回と対戦者 2 が勝った対戦回の両方を区別せずに含めている。また、図Ⅳ-1 において、1000 回目の対戦回の時点での各手が勝った対戦回の総回数の縦軸の値は、グーが 155 回、チョキが 128 回、パーが 362 回であった。

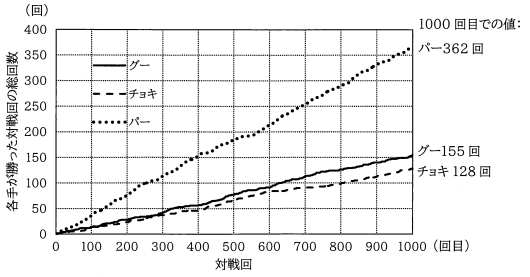
さらに、この 1000 回のシミュレーションのうち、パーが出た対戦回の中でのパーの勝ち、負け、あいこの割合(比率)を各対戦回までごとに調べた。その結果を、次ページの図Ⅳ-2 に示すグラフに表した。特に、1000 回目の対戦回の時点での図Ⅳ-2 のグラフの縦軸の値は、勝ちが 0.54、負けが 0.19、あいこが 0.27 であった。ここでも、パーを出したのが対戦者 1 か対戦者 2 かは区別せずに、合わせて集計している。

設問(A)

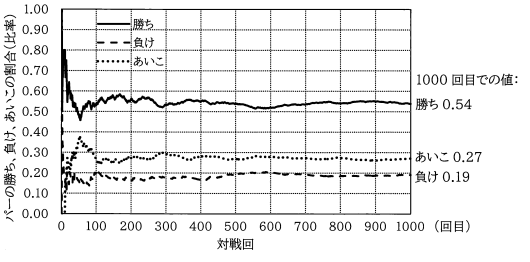
図Ⅳ-1 と図Ⅳ-2 から、1000 回目の対戦回までででパーが出た対戦回の総回数は、およそ［ ア ］回の範囲内と言える。また、1000 回目の対戦回までででパーであいこになった対戦回の総回数は、およそ［ イ ］回の範囲内と言える。空欄［ ア ］・［ イ ］に入る適切な選択肢を解答群からそれぞれ 1 つずつ選んで、その番号を解答欄にマークせよ。

［ ア ］・［ イ ］の解答群:

【 (0) 50～100
(1) 150～200
(2) 250～300
(3) 350～400
(4) 450～500
(5) 550～600
(6) 650～700
(7) 750～800 】

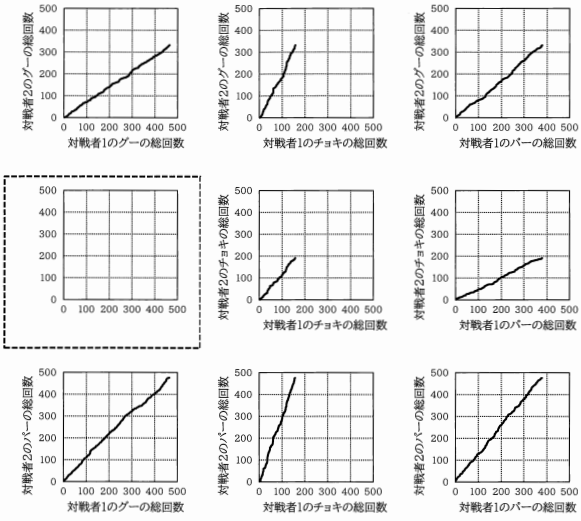


図Ⅳ-1. 1000 回の対戦シミュレーションにおける各手が勝った対戦回の総回数の変化



図Ⅳ-2. 1000 回の対戦シミュレーションにおけるパーの勝ち、負け、あいこの割合の変化

次に、この 1000 回の対戦において、各対戦回までに各対戦者が各手を出した総回数を、対戦者ごとに調べた。この結果を用いて、各対戦回までの「対戦者 1 がグーを出した対戦回の総回数と対戦者 2 がグーを出した対戦回の総回数」などのように、対戦者 1 と対戦者 2 で組み合わせた総回数どうしの関係を対戦回ごとに調べた。これらの関係を、手の組み合わせの総当たりで並べたグラフを、下の図Ⅳ-3 のように表した。図Ⅳ-3 内の各グラフは、2 人の対戦者それぞれが各手を出した総回数を同一対戦回で組にしたプロット点を、1 回目から 1000 回目までの対戦回に沿って線でつなげて表したものである。横軸の「対戦者 1 のグーの総回数」は、「対戦者 1 が各対戦回までにグーを出した対戦回の総回数」の意味である。その他横軸や縦軸の「対戦者 1 のチョキの総回数」や「対戦者 2 のグーの総回数」なども、同様の表現としている。



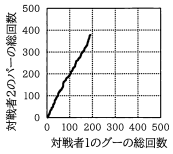
図Ⅳ-3. 1000 回の対戦における対戦者 1 と対戦者 2 が各手を出した対戦回の総回数どうしの関係

設問(B)

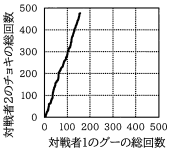
図Ⅳ-3 内の２行目１列目のグラフ（点線で囲ったグラフ）として正しいものを、下の［ ウ ］の解答群から1つ選び、その番号を解答欄にマークせよ。

［ ウ ］の解答群：

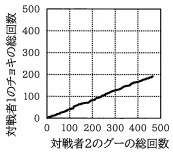
【 (0)



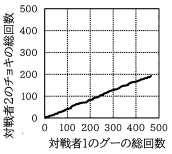
(1)



(2)



(3)



】

図Ⅳ-3 から、1000 回の対戦回全体の中で対戦者 1 と対戦者 2 がそれぞれどの手を出しやすかったかが分かる。この 1000 回の対戦シミュレーションを行った際にプログラムで設定していた、各対戦者が各手を出す確率の値の組み合わせとして適切なものを解答群から対戦者ごとに1つずつ選び、その番号をそれぞれの解答欄にマークせよ。

対戦者 1 が各手を出す確率の値の組み合わせ　　グー ： チョキ ： パー ＝ ［ エ ］

対戦者 2 が各手を出す確率の値の組み合わせ　　グー ： チョキ ： パー ＝ ［ オ ］

［ エ ］・［ オ ］の解答群：

【 (0) 0.15 : 0.60 : 0.25　(1) 0.33 : 0.33 : 0.34　(2) 0.35 : 0.15 : 0.45

(3) 0.35 : 0.20 : 0.45　(4) 0.45 : 0.15 : 0.40　(5) 0.50 : 0.30 : 0.30 】

数学(理系科目)（11/24実施分）

問題番号	設問	正解
[Ⅰ]	アイ	17
	ウエ	95
	オ	1
	カ	2
	キ	1
	ク	1
	ケ	2
	コ	1
	サ	1
	シ	1
	ス	2
	セ	3
	ソ	2
	タ	6
	チツ	35
	テトナ	729
	ニ	2
	ヌ	n
	ネ	2
	ノ	3
	ハ	n
	ヒ	1

問題番号	設問	正解
[Ⅱ]	ア	4
	イウ	−4
	エ	1
	オ	2
	カ	1
	キ	2
	ク	−
	ケ	2
	コ	2
	サ	2
	シ	2
	ス	1
	セ	2
	ソ	1
	タ	2
	チ	3
	ツ	4
	テ	1
	ト	8
	ナ	5
	ニ	2
	ヌ	7
	ネ	6
	ノハ	23
	ヒフ	16
	ヘ	2

問題番号	設問	正解
[Ⅲ]	ア	3
	イ	1
	ウ	2
	エ	2
	オ	4
	カ	9
	キ	4
	ク	2
	ケ	5
	コ	4
	サ	3
	シ	2
	ス	3
	セ	4
	ソ	7
	タ	4
	チ	1
	ツ	2
	テ	1
	ト	2
	ナ	2
	ニ	2
	ヌ	2
	ネ	3

情　報（11/22実施分）

問題番号	設問	正解
[Ⅰ]	ア	3
	イ	6
	ウ	9
	エ	8
	オ	4
	カ	1

問題番号	設問	正解
[Ⅱ]	ア	3
	イ	6
	ウ	5
	エ	0
	オ	8
	カ	6
	キ	4
	ク	1
	ケ	0

問題番号	設問	正解
[Ⅲ]	ア	3
	イ	3
	ウ	0
	エ	4
	オ	5
	カ	3
	キ	0
[Ⅳ]	ア	6
	イ	1
	ウ	3
	エ	4
	オ	3

国　語（11/22実施分）

問題番号	設問		正解
[一]	問一	Ⅰ	2
		Ⅱ	3
		Ⅲ	5
		Ⅳ	3
	問二		2
	問三	a	4
		b	3
		c	6
		e	5
	問四		4
	問五		2
	問六		5
	問七		3
	問八		4
	問九		3
	問十		5

問題番号	設問		正解
[二]	問一	ア	1
		イ	4
		ウ	2
		エ	3
		オ	4
	問二		3
	問三	a	2
		b	3
	問四	Ⅰ	1
		Ⅱ	5
	問五		1
	問六	甲	5
		乙	4
	問七		3
	問八		4
	問九		2
	問十		5
	問十一		1

国　語（11/23実施分）

問題番号	設問		正解
[一]	問一	Ⅰ	3
		Ⅱ	4
		Ⅲ	5
	問二	a	6
		b	1
		c	5
		f	2
	問三		5
	問四		1
	問五		4
	問六		3
	問七	d	1
		e	3
	問八		2
	問九		5
	問十		6

問題番号	設問		正解
[二]	問一	Ⅰ	2
		Ⅱ	3
		Ⅲ	5
	問二	ア	2
		イ	4
		ウ	1
		エ	1
		オ	3
	問三	a	4
		b	5
	問四		5
	問五		1
	問六		3
	問七		4
	問八		5
	問九		4
	問十		5
	問十一		3