

2025年度

慶應義塾大学入学試験問題

環境情報学部

情報および数学

注意事項1

1. 試験開始の合図があるまで、この問題冊子を開かないでください。
2. 問題冊子は全部で24ページです。
  - ・情報の問題Ⅰ～Ⅲは4ページから15ページです。
  - ・数学の問題Ⅳ～Ⅵは18ページから22ページです。
3. 試験開始の合図とともにすべてのページが揃っているか確認してください。  
ページの欠落・重複があった場合には、直ちに監督者に申し出てください。
4. 問題冊子の2ページに「注意事項2」があります。3ページに「注意事項3」があります。試験開始後必ず読んでください。
5. 問題冊子は、試験終了後必ず持ち帰ってください。
6. 受験番号と氏名は、解答用紙の所定の欄に必ず記入してください。
7. 解答用紙の「注意事項」を必ず読んでください。

## 注 意 事 項 2

問題冊子に数字の入った  $\square$  があります。それらの数字は解答用紙の解答欄の番号をあらわしています。対応する番号の解答欄の 0 から 9 までの数字または - (マイナスの符号) をマークしてください。

$\square$  が 2 個以上つながったとき、数は右詰めで入れ、左の余った空欄には 0 を入れてください。負の数の場合には、マイナスの符号を先頭の  $\square$  に入れてください。また、小数点以下がある場合には、左詰めで入れ、右の余った空欄には 0 を入れてください。

$$(例) \quad 12 \longrightarrow \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline \end{array}$$

$$-3 \longrightarrow \begin{array}{|c|c|c|} \hline - & 0 & 3 \\ \hline \end{array}$$

$$1.4 \longrightarrow \begin{array}{|c|c|c|} \hline 0 & 0 & 1 \\ \hline \end{array} . \begin{array}{|c|c|} \hline 4 & 0 \\ \hline \end{array}$$

$$-5 \longrightarrow \begin{array}{|c|c|c|} \hline - & 0 & 5 \\ \hline \end{array} . \begin{array}{|c|c|} \hline 0 & 0 \\ \hline \end{array}$$

分数は約分した形で解答してください。マイナスの符号は分母には使えません。

$$(例) \quad \frac{4}{8} \longrightarrow \frac{1}{2} \longrightarrow \frac{\begin{array}{|c|c|} \hline 0 & 1 \\ \hline \end{array}}{\begin{array}{|c|c|} \hline 0 & 2 \\ \hline \end{array}}$$

$$-\frac{6}{9} \longrightarrow -\frac{2}{3} \longrightarrow \frac{\begin{array}{|c|c|} \hline - & 2 \\ \hline \end{array}}{\begin{array}{|c|c|} \hline 0 & 3 \\ \hline \end{array}}$$

ルート記号の中は平方因子を含まない形で解答してください。

$$(例) \quad \sqrt{50} \longrightarrow \begin{array}{|c|c|} \hline 0 & 5 \\ \hline \end{array} \sqrt{\begin{array}{|c|c|} \hline 0 & 2 \\ \hline \end{array}}$$

$$-\sqrt{24} \longrightarrow \begin{array}{|c|c|} \hline - & 2 \\ \hline \end{array} \sqrt{\begin{array}{|c|c|} \hline 0 & 6 \\ \hline \end{array}}$$

$$\sqrt{13} \longrightarrow \begin{array}{|c|c|} \hline 0 & 1 \\ \hline \end{array} \sqrt{\begin{array}{|c|c|} \hline 1 & 3 \\ \hline \end{array}}$$

$$-\frac{\sqrt{18}}{6} \longrightarrow \frac{\begin{array}{|c|c|} \hline - & 1 \\ \hline \end{array} \sqrt{\begin{array}{|c|c|} \hline 0 & 2 \\ \hline \end{array}}}{\begin{array}{|c|c|} \hline 0 & 2 \\ \hline \end{array}}$$

数式については、つぎの例のようにしてください。分数式は約分した形で解答してください。

$$(例) \quad \sqrt{12a} \longrightarrow \begin{array}{|c|c|} \hline 0 & 2 \\ \hline \end{array} \sqrt{\begin{array}{|c|c|} \hline 0 & 3 \\ \hline \end{array}} a$$

$$-a^2 - 5 \longrightarrow \begin{array}{|c|c|} \hline - & 1 \\ \hline \end{array} a^2 + \begin{array}{|c|c|} \hline 0 & 0 \\ \hline \end{array} a + \begin{array}{|c|c|} \hline - & 5 \\ \hline \end{array}$$

$$\frac{4a}{2a-2} \longrightarrow \frac{-2a}{1-a} \longrightarrow \frac{\begin{array}{|c|c|} \hline 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|} \hline - & 2 \\ \hline \end{array} a}{1 - \begin{array}{|c|c|} \hline 0 & 1 \\ \hline \end{array} a}$$

選択肢の番号を選ぶ問題では、最も適切な選択肢を 1 つだけ選んでください。また、同じ選択肢を複数回選んでもかまいません。

### 注 意 事 項 3

問題文中のプログラムの表記は、大学入学共通テストで使用されている「共通テスト用プログラム表記」に準じます。以下にその概要を示します。

**文字列** 「"SFC"」のようにダブルクォーテーションで囲みます。複数の文字列を「+」で結合できます。

**配列** 「[1, 2, 3]」のように、要素の並びを角括弧で囲みます。

**変数** 変数名は英字で始まる英数字とアンダースコアの並びです。ただし、配列変数名は先頭文字が大文字です。配列の要素は、「Data[0]」のように、配列変数名の後に角括弧で囲んだ添字を書きます。添字は0から始まります。

**代入文** 「count = 1」のように等号の左辺に変数、右辺に式を書きます。変数には、数値だけでなく文字列や配列も代入できます。

例：name = "Keio " + "SFC"

例：Data = [1, 2, 3]

例：Data[0] = 4

**演算子** 加減乗除と不等号は数学と同じです。数学と異なるのは次の演算子です。

整数の除算：商（整数）「÷」、余り「%」

比較演算：等しい「==」、等しくない「!=」、以上「>=」、以下「<=」

論理演算：論理積「and」、論理和「or」、否定「not」

**関数** 関数呼び出しは、「平均 (Data)」のように、関数名の後に括弧で囲んだ引数を書きます。

「表示する」という関数は、引数の数値や文字列を画面に表示する関数です。それ以外の関数は問題文中に説明があります。

**制御文** 条件分岐や繰り返しの制御文の下にある罫線は、その制御文の制御範囲を表します。「└」が制御範囲の最後の文です。

例：もし  $n \% 2 == 0$  ならば：

┌  $n = n \div 2$

そうでなければ：

└  $n = 3 * n + 1$

例：i を 0 から 2 まで 1 ずつ増やしながら繰り返す：

└ sum = sum + Data[i]

## 情報 I

(ア) 2 進法表現および 10 進法表現による整数および小数について述べた次の文章の空欄 (1) (2) (3) ～ (15) (16) (17) (18) (19) (20) に入るもっとも適した数字を解答欄にマークしなさい。

計算機内部で 2 進法を使って整数を表現する場合、負の数を表すのに符号は用いずに 2 の補数表現を用いることが多い。たとえば、2 の補数表現を用いて 8 桁の数で正および負の整数と 0 を表す場合には、その範囲は  $(- (1) (2) (3))_{10}$  から  $((4) (5) (6))_{10}$  までになる。2 の補数表現を用いる理由の一つは、次の図のように、負の数の加算がそのまま 2 進法で表現された数の加算で計算できることである。

$$\begin{array}{r}
 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \\
 + \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \\
 \hline
 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0
 \end{array}$$

2 の補数表現で  $(00110101)_2$  は正の数であるが、 $(10010001)_2$  は負の数である。加算結果を表している  $(11000110)_2$  は 10 進法で表すと (7) (8) (9) であり、負の数の加算として正しく計算できていることがわかる。

次に、小数点を含む数について考える。 $(0.10101)_2$  は 10 進法的小数で表すと、0. (10) (11) (12) (13) (14) となる。ここで注意が必要なのは、10 進法で表したときに有限桁の小数でも、それを 2 進法での表現にすると循環小数になる場合があることである。たとえば、 $(0.3)_{10}$  は 2 進法では、0. (15) (16) (17) (18) (19) (20) ... となり、循環小数になってしまう（ここでは小数第 6 位までを示し、それ以下を ... で示してある）。

(イ) ハフマン符号とそれに準じた方法による情報の圧縮について述べた次の文章の空欄 (21) (22) (23)、(37) (38) (39) に入るもっとも適した数字を解答欄にマークしなさい。ただし、必要に応じて小数第 3 位を四捨五入して解答すること。また、(24) ～ (35) (36) に入るもっとも適した語を選択肢から選び、解答欄にマークしなさい。

4 種類の文字 A, B, C, D がある一定の出現確率で出現する文字列の符号化について考える。ここでは、符号は記号 0 と 1 を組み合わせたものとする。例えば、それぞれの文字に符号 0、10、110、111 を割り当てる。このような割り当てを以下では、次のように表現する。

$$\{A, B, C, D\} = \{0, 10, 110, 111\}$$

この割り当てでは、A, B, C, D に長さが 1, 2, 3, 3 の符号が割り当てられている。ここで、文字列を符号化したときの 1 文字あたりの符号の長さの期待値を平均符号長と呼ぶ。この符号化の例で、A, B, C, D の出現確率がすべて等しく 0.25 だった場合、平均符号長は (21) (22) (23) となる。

このような符号の割り当てでなく、各符号に同じ長さを割り当てて次のようにすると、平均符号長は 2 となり (21) (22) (23) より短くなる。

$$\{A,B,C,D\} = \{00,01,10,11\}$$

また、次のような符号の割り当てを考える。

$$\{A,B,C,D\} = \{0,01,11,10\}$$

この割り当てを使って元の文字列を符号化して 01101 という記号列が得られたとすると、元の文字列は 

|      |
|------|
| (24) |
|------|

|      |
|------|
| (25) |
|------|

|      |
|------|
| (26) |
|------|

 であることが分かり、一意に復元可能である。しかし、01001 という記号列が得られたとすると、ADB という文字列に復元できるが、他の文字列を符号化して同じ記号列が得られる場合があり、一意には復元できない。つまり、短い符号を割り当てることで平均符号長をいくらでも短くできるわけではなく、一意に復元できるためには、ある程度の長さが必要になる。

次に、より文字数の多い場合について、できるだけ平均符号長が短く、一意に復元できる符号の割り当てを考える。A, B, C, D, E, F の 6 文字があり、出現確率が次のように分かっているとする。A の出現確率 0.30 から、F の出現確率 0.05 までは順番に示されている。

| 文字   | A    | B    | C    | D    | E    | F    |
|------|------|------|------|------|------|------|
| 出現確率 | 0.30 | 0.22 | 0.17 | 0.16 | 0.10 | 0.05 |

このとき、平均符号長が最も短くなるような最適な符号の構成方法は複数あるが、その一つは次のようになる。ここでは同じ長さの符号について出現確率の大きいほうの符号が小さい符号よりも 2 進法の数として大きくなるように符号を割り当てている。たとえば、2 つの文字に 3 桁の符号 110 と 111 を割り当てる場合、出現確率の大きい文字のほうに 111 を割り当て、小さいほうに 110 を割り当てる。また、異なる長さの符号において、長さが短いほうの符号のうしろに、同じ長さにするのに必要なだけ 0 を付加して比較したときに、出現確率の大きいほうの符号が 2 進法の数として大きくなるように割り当てている。たとえば、110 と 1011 では、出現確率の大きいほうに割り当てる 110 に 0 を付け足し 1100 とした数と 1011を比較して 1100 のほうが大きいということである。

| 文字 | 符号   |      |
|----|------|------|
| A  | 11   |      |
| B  | (27) | (28) |
| C  | (29) | (30) |
| D  | (31) | (32) |
| E  | (33) | (34) |
| F  | (35) | (36) |

このとき、平均符号長は、 

|      |
|------|
| (37) |
|------|

 . 

|      |
|------|
| (38) |
|------|

|      |
|------|
| (39) |
|------|

 となる。

【(24)～(26)の選択肢】

(1) A (2) B (3) C (4) D

【(27)(28)～(35)(36)の選択肢】

|      |      |      |      |      |      |      |       |      |       |
|------|------|------|------|------|------|------|-------|------|-------|
| (11) | 00   | (12) | 01   | (13) | 10   | (14) | 11    | (15) | 000   |
| (16) | 001  | (17) | 010  | (18) | 011  | (19) | 100   | (20) | 101   |
| (21) | 110  | (22) | 111  | (23) | 0000 | (24) | 0001  | (25) | 0010  |
| (26) | 0011 | (27) | 0100 | (28) | 0101 | (29) | 0110  | (30) | 0111  |
| (31) | 1000 | (32) | 1001 | (33) | 1010 | (34) | 1011  | (35) | 1100  |
| (36) | 1101 | (37) | 1110 | (38) | 1111 | (39) | 00000 | (40) | 00001 |

## 情報Ⅱ

囲碁や将棋のように、2 人が交互に 1 手ずつ指し、不確定な要素がなく、局面に関するすべての情報が明らかで、片方の利得がもう片方の損失になるようなゲームを考える。

プレイヤーとしては、自分が次に指す局面でどの手を指すのが最善であるかを知りたい。どの手が最善かは、その後のすべての可能性を調べてみないとわからないが、一般には局面の数が多すぎて、すべて調べるのは不可能である。そこで、何手か先の局面を考えて、自分にもっとも有利な局面になるような手を最善手であると定義する。ここでは、2 手先の局面（自分が 1 手指し、その後に相手が 1 手指した状態）から最善手を決めることにする。

話を簡単にするため、すべての局面について、自分の番の時は A, B, C の 3 種類の手が指せ、相手の番の時は a, b, c の 3 種類の手が指せるものとする。また、現在の局面から自分が A という手を指した場合の局面を [A]、さらにその後に相手が a という手を指した場合の局面を [Aa] のように表す。

2 手先の局面のそれぞれに対して、何らかの基準で評価値を与えることができるものとする。評価値は、その局面が自分にどれだけ有利かを数値で表したもので、数値が大きいほど自分に有利（相手にとっては不利）である。複数の局面が同じ評価値になる場合の煩雑さを避けるため、評価値は 1 から 9 までの整数で、2 手先の局面のすべてに対して互いに異なる値が与えられるものとする。例えば、ある局面における 2 手先の局面の評価値は次の表のように与えられる。

| 局面  | [Aa] | [Ab] | [Ac] | [Ba] | [Bb] | [Bc] | [Ca] | [Cb] | [Cc] |
|-----|------|------|------|------|------|------|------|------|------|
| 評価値 | 3    | 6    | 5    | 4    | 1    | 7    | 2    | 8    | 9    |

最善手を選ぶためには、A, B, C のそれぞれに対して、その手を指した時に想定される 2 手先の局面の評価値を計算すればよい。なお、「A を指した時に想定される 2 手先の局面の評価値」を略して「[A] の評価値」と呼ぶことにする。

(ア) 空欄 、 に当てはまるもっとも適切な数字を解答欄にマークしなさい。

上の例で A を指した場合に 2 手先の局面がどうなるか考える。相手が b を指してくれれば局面 [Ab] になって評価値は 6 であるが、相手は評価値がもっとも低くなる（相手にとって有利になる）ような手を当然指すはずなので、a を指して局面 [Aa] となると考えるべきである。したがって、[A] の評価値は [Aa] の評価値である 3 とするのが妥当である。同様に考えると、[B] の評価値は 、[C] の評価値は  となるので、現在の局面では A を指すのが最善ということになる。

(イ) 空欄  ～  に当てはまるもっとも適切な選択肢を下から選び、その番号を解答欄にマークしなさい。

上で述べた考え方に従って最善手を選ぶプログラムは次のようになる。ただし、左端の数字は行の番号である。

```

(1) H1 = ["A", "B", "C"]
(2) H2 = ["a", "b", "c"]
(3) m1 = (42)
(4) i を 0 から 2 まで 1 ずつ増やしながら繰り返す :
(5)     m2 = (43)
(6)     j を 0 から 2 まで 1 ずつ増やしながら繰り返す :
(7)         v = 評価 ("[" + H1[i] + H2[j] + "]")
(8)         もし v < (44) ならば :
(9)             m2 = (45)
(10)     もし m2 > (46) ならば :
(11)         m1 = (47)
(12)         x = (48)
(13) 表示する (H1[x] + "が最善手")

```

ここで評価は次のような関数であるとする。

評価 (局面を表す文字列) : 局面の評価値を返す

例えば、評価値が上の例のようになっていれば、評価 (" [Aa] ") は 3 を返す。

【(42) ~ (48) の選択肢】

(1) H1 (2) H2 (3) i (4) j (5) m1  
 (6) m2 (7) v (8) x (9) 10 (0) 0

(ウ) 空欄 (49) ~ (50) に当てはまるもっとも適切な数字を解答欄にマークしなさい。

2 手先の局面は上の表に示したように 9 個あるが、そのすべての評価値を計算しなくても最善手が決まる場合がある。

表の左から順に計算していくとすると、まず [Aa], [Ab], [Ac] の評価値を計算し、[A] の評価値が 3 であるとわかる。次に [Ba], [Bb] を計算すると、[Bb] の評価値が 1 であることから [B] の評価値が 1 以下であることが確定し、[Bc] の評価値を計算しなくても、[B] の評価値は [A] の評価値より小さく、最善手には選ばれないことがわかる。同様に [C] についても評価値を計算しなくてもよい局面が存在する。評価値の計算が不要な局面を除くと、9 個の局面のうち (49) 個の局面の評価値を計算すれば最善手を決めることができる。

なお、何個の局面の評価値の計算が必要かは、計算の順序に依存する。表の右から順に計算していくとすると、(50) 個の局面の評価値を計算する必要がある。

(エ) この考え方に従って、前に述べたプログラムの 7 行目にある評価 ("[" + H1[i] + H2[j] + "]")



が不要な場合に実行しないようにするにはどうすればよいか、次の選択肢の中からもっとも適切なものを選び、その番号を (51) にマークしなさい。ただし「繰り返しを中断する」とは、現在実行中のもっとも内側の繰り返しをそこで中断し、繰り返しの次の文から引き続き実行することである。

【(51) の選択肢】

- (1) 5 行目の次に「もし  $m1 > m2$  ならば繰り返しを中断する」という文を挿入する。挿入した文は 4 行目から始まる繰り返しの範囲内に入る。
- (2) 7 行目の次に「もし  $v < m1$  ならば繰り返しを中断する」という文を挿入する。挿入した文は 6 行目から始まる繰り返しの範囲内に入る。
- (3) 8 行目の次に「もし  $j < x$  ならば繰り返しを中断する」という文を挿入する。挿入した文は 8 行目から始まる条件分岐の範囲内に入る。
- (4) 9 行目の次に「もし  $v < m1$  ならば繰り返しを中断する」という文を挿入する。挿入した文は 6 行目から始まる繰り返しの範囲内に入るが、8 行目から始まる条件分岐の範囲には入らない。
- (5) 11 行目の次に「もし  $m1 > m2$  ならば繰り返しを中断する」という文を挿入する。挿入した文は 10 行目から始まる条件分岐の範囲内に入る。
- (6) 12 行目の次に「もし  $j < x$  ならば繰り返しを中断する」という文を挿入する。挿入した文は 4 行目から始まる繰り返しの範囲内に入るが、10 行目から始まる条件分岐の範囲には入らない。

## 情報Ⅲ

1つの情報伝送路に複数の装置を接続して、装置間の情報通信を行うことを考える。ここで情報伝送路を「バス」と呼ぶ。バスには、図1のように装置A、装置B、装置C、装置Dの4つの機器が接続されている。各装置は送信部からバスに0か1を出力することで1ビットのデータの送信を行う。バスの状態は、各装置の出力値の論理和 (OR) となる。ただし、論理和の計算において0は偽、1は真を表しているものとする。各装置はバスの状態を読み出すことで1ビットのデータの受信を行う。また、同期信号発生装置が各装置に同期信号を送信している。

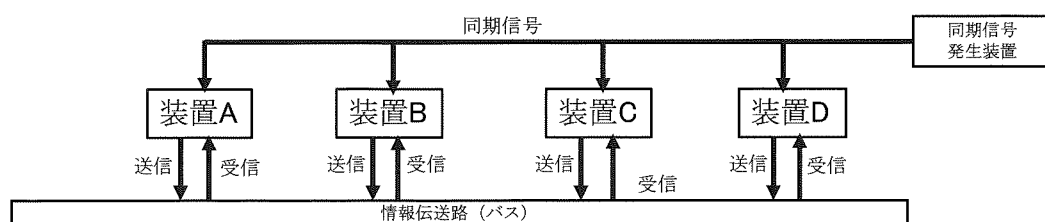


図1

装置間で送受信する情報は、IDとデータである。IDは、送信元の装置を識別するためのもので、各装置に事前に割り当てられた1～15の数値である。データは、各装置の状態や、装置のセンサーで計測した値などを想定している。そのようなデータは、大きなサイズでかつ時刻によって変化すると考えられるが、ここでは簡単化のため0～15の固定値とする。装置間では、上位4ビットを装置のID (2進法表記)、下位4ビットを装置のデータ (2進法表記) とした計8ビットの情報を送受信する。各装置から情報を送信する時は、上位ビットから送信を行うものとする。たとえば、IDが2 (10進法表記) でデータが5 (10進法表記) の場合、00100101を、0, 0, 1, 0, 0, 1, 0, 1の順に送信する。

装置A～Dが同じタイミングで情報の送受信を行うために、各装置は、同期信号発生装置から同一の同期信号を受信している。同期信号発生装置は、一定時間周期で、0, 1を交互に送信する。各装置は、プログラムの中で同期信号の変化を待つ処理を行うことで、同じタイミングでデータの送信や受信を行うことができる。なお、プログラムの実行速度は十分速く、同期信号の変化を待つ処理以外の処理の実行中に同期信号が変化することはないものとする。

装置のIDとデータから成る8ビットの情報を送受信する周期を1サイクルとして、サイクル番号0, 1, 2, ... を変数  $t$  で表記する。また、1ビットのデータの送受信を1ステップとし、ステップ番号を変数  $s$  で表記する。1サイクル内でステップを進めると、 $s$  の値は0, 1, 2, ..., 7と変化する。各装置のプログラムは同じタイミングで実行を開始し、同期信号に合わせて  $t, s$  を更新するものとする。すなわち、全ての装置で同じタイミングで  $(t=0, s=0), (t=0, s=1), (t=0, s=2), \dots (t=0, s=7), (t=1, s=0), (t=1, s=1), \dots$  と更新されることになる。

各装置において、データを送信する周期は、 $P$  サイクルに1回とし、 $P$  は装置により異なるものとする。

表 1 に各装置の ID、データ、送信周期 ( $P$ ) を 10 進法表記で示す。各装置内のプログラムの中で、各装置の ID、データ、送信周期 ( $P$ ) については、それぞれ、変数  $id$ 、 $data$ 、 $p$  に表 1 に示す値が初期値としてセットされるものとする。

表 1

|      | 装置の ID<br>(変数は $id$ ) | 装置のデータ<br>(変数は $data$ ) | データの送信周期 ( $P$ )<br>(変数は $p$ ) |
|------|-----------------------|-------------------------|--------------------------------|
| 装置 A | 6                     | 3                       | 3                              |
| 装置 B | 9                     | 6                       | 4                              |
| 装置 C | 4                     | 10                      | 6                              |
| 装置 D | 14                    | 2                       | 5                              |

以降では、各装置で同一のプログラム (ただし  $id$ 、 $data$ 、 $p$  の初期値は装置毎に異なる) により、各装置が送信する情報を他の装置で受信するためのプログラムを作成する。なお、取り扱う数値はすべて整数であるものとする。

まず、プログラムの同期信号に基づく、周期、ステップの更新処理、各ステップで行う処理、周期更新時に行う処理について考える。システムには、各装置におけるビットデータの送信・受信タイミングを同期するための関数、**同期待ち ()** が用意されている。この関数を実行すると、同期信号が変化するまで待機する。この関数を使い、通信プログラム P1 を以下のように作成する。ただし、プログラム P1 の中で以降にプログラムを作成する部分は『』で示されている。

## 【プログラム P1】

```

t = 0
s = 0
常に繰り返す :
    同期待ち ()
    もし s == 0 ならば :
        『送信判断・送信データ作成』
        『ビットデータの送信・受信』
        s = s + 1
    もし s > 7 ならば :
        『受信した ID・データの表示』
        t = t + 1
        s = 0

```

(ア) 空欄 (52) ～ (60) に当てはまるもっとも適切なものを選択肢から選び、その番号を解答欄にマーク

しなさい。

P1 中の『送信判断・送信データ作成』の部分のプログラムを作成する。ここでは、そのサイクルにおいて、情報を送信する・しないを変数  $t$  の値により決定し、送信データを 0, 1 の数値で変数に格納するプログラムを作成する。この処理は、 $s$  の値が 0 の時に行うので、各サイクルの最初に行う処理となる。具体的な処理については、以下のようにする。

- $t$  が  $p$  の倍数になる時 ( $t$  が 0 の場合も含む) に、情報を送信すると決定し、変数 `send_ok` に 1 (送信する) を代入する。それ以外の場合は、`send_ok` に 0 (送信しない) を代入する。
- 上位 4 ビットを ID、下位 4 ビットをデータとした 1 バイトを変数 `send_byte` に代入する。`send_byte` を 2 進数で表記し、各桁の値をステップ  $s$  の時に送信するビットデータ (0 あるいは 1) として、配列変数 `Send_bit[s]` に代入する。ビットデータの送信は上位ビットから行うため、`Send_bit[0]` に最上位ビットの値、`Send_bit[7]` に最下位ビットの値を代入する。

上記を実現するプログラムとして、以下のようなプログラム P2 を作成した。

#### 【プログラム P2】

```
もし  $t$  (52)  $p == 0$  ならば :
|   send_ok = 1
そうでなければ :
|   send_ok = 0
send_byte =  $id$  (53) 16 (54) data
 $i$  を 7 から 0 まで 1 ずつ減らしながら繰り返す :
|   Send_bit[i] = send_byte (55) 2
|   send_byte = send_byte (56) 2
```

次にプログラム P1 中の『ビットデータの送信・受信』の部分のプログラムを作成する。具体的な処理については、以下のようにする。

- システムには、ビットデータをバスに送信するための関数として**送信** ( $n$ ) が用意されている。この関数を実行すると、 $n$  (0 または 1) をバスに送信できるものとする。`send_ok` が 1 の場合、『送信判断・送信データ作成』で `Send_bit[s]` にセットした値 (0 または 1) を送信し、`send_ok` が 0 の場合は 0 を送信する。バスの状態は各装置から送信されたビットデータの論理和となるため、0 の送信はバスの状態に影響を与えない。
- システムには、バスの状態を読みだす関数として**受信** () が用意されている。この関数を実行すると、バスの状態 (0 または 1) を返り値として取得できる。ステップ  $s$  の時に読み出したバスの状態を配列変数 `Receive_bit[s]` に代入する。
- ステップ内で全ての装置からのビットデータの送信が完了し、十分に時間が経った後に**受信** () に

よるバスの状態の読み出しを行うために、同期信号が変化するまで待機してから受信 ( ) を実行する。

上記を実現するプログラムとして、以下のようなプログラム P3 を作成した。

#### 【プログラム P3】

もし send\_ok == 1 ならば :

    | 送信 (Send\_bit[s])

そうでなければ :

    | 送信 (0)

同期待ち ( )

Receive\_bit[s]= 受信 ( )

最後にプログラム P1 中の『受信した ID・データの表示』の部分のプログラムを作成する。この部分は、s > 7 の時に実行されるため、Receive\_bit[0]～Receive\_bit[7] により、そのサイクルにおける受信データを得ることが可能である。プログラムでは、上位 4 ビットが ID、下位 4 ビットがデータとなる 1 バイトの情報を格納する変数 receive\_byte を用意し、これを使い ID、データをそれぞれ変数 receive\_id, receive\_data に代入して t と共に表示する。

上記を実現するプログラムとして、以下のようなプログラム P4 を作成した。

#### 【プログラム P4】

receive\_byte = 0

i を 0 から 7 まで 1 ずつ増やしながら繰り返す :

    | receive\_byte = receive\_byte (57) 2

    | receive\_byte = receive\_byte (58) Receive\_bit[i]

receive\_id=receive\_byte (59) 16

receive\_data=receive\_byte (60) 16

表示する ("t=", t)

表示する ("装置の ID=", receive\_id)

表示する ("装置のデータ=", receive\_data)

#### 【(52) ～ (60) の選択肢】

(1) + (2) - (3) ÷ (4) \* (5) %

(イ) 空欄 (61) ～ (66) に当てはまるもっとも適切なものを選択肢から選び、その番号を解答欄にマークしなさい。

ここまで作成したプログラムを装置 A～D で実行して通信を行う場合、1 サイクルの中で、1 つの装

置のみがデータを送信した場合は、バスの状態は、その装置の送信ビットデータと同じになり、各装置は送信元の情報を正しく受信できる。一方で、1 サイクルの中で複数の装置がデータを送信した場合、バスの状態は論理和となる（複数の装置から送信された送信ビットデータの中に 1 が 1 つでもあれば 1 となる）ため、各装置では正しく情報を受信できない。また、1 サイクルの中でどの装置もデータを送信しない場合、各装置で受信・表示される ID は 0、データは 0 となる。

例えば、 $t$  の値が 1 の時に各装置で受信・表示される ID は  $\boxed{(61)}$ 、データは  $\boxed{(62)}$  となる。 $t$  の値が 3 の時に各装置で受信・表示される ID は  $\boxed{(63)}$ 、データは  $\boxed{(64)}$  となる。 $t$  の値が 12 の時に各装置で受信・表示される ID は  $\boxed{(65)}$ 、データは  $\boxed{(66)}$  となる。

### 【 $\boxed{(61)}$ ～ $\boxed{(66)}$ の選択肢】

- (1) 0   (2) 2   (3) 3   (4) 4   (5) 6  
 (6) 9   (7) 10   (8) 11   (9) 14   (0) 15

(ウ) 空欄  $\boxed{(67)}$  ～  $\boxed{(70)}$  に当てはまるもっとも適切なものを選択肢から選び、その番号を解答欄にマークしなさい。

1 サイクルに 2 つ以上の装置が情報を送信し、送信情報が衝突することを防止するために、「ビットデータの送信・受信」の部分で改良したプログラム P3 改を作成した。このプログラムでは、バスの状態と自分が送信したビットデータが異なる場合に、send\_ok を 0 にし、同一サイクル内の以降のステップで 0 を出力する（情報の送信をしない）ことで、送信情報の衝突を回避する。さらに、変数 send\_again(初期値 0) を用意し、送信情報の衝突回避のために send\_ok を 0 にした場合、send\_again に 1 を代入し次のサイクルでデータを送信行うようにしている。

P3 を P3 改に修正したプログラムを装置 A～D で実行して通信を行う場合、 $t$  の値が 12 の時に各装置で受信・表示される ID・データの送信元は  $\boxed{(67)}$ 、 $t$  の値が 13 の時に各装置で受信・表示される ID・データの送信元は  $\boxed{(68)}$ 、 $t$  の値が 14 の時に各装置で受信・表示される ID・データの送信元は  $\boxed{(69)}$ 、 $t$  の値が 15 の時に各装置で受信・表示される ID・データの送信元は  $\boxed{(70)}$  である。

## 【プログラム P3 改】

```
もし s == 0 ならば :  
  |  もし send_again == 1 ならば :  
  |  |  send_ok = 1  
  |  |  send_again = 0  
  |  
もし send_ok == 1 ならば :  
  |  送信 (Send_bit[s])  
そうでなければ :  
  |  送信 (0)  
同期待ち ()  
Receive_bit[s] = 受信 ()  
もし send_ok == 1 ならば :  
  |  もし Receive_bit[s] != Send_bit[s] ならば :  
  |  |  send_ok = 0  
  |  |  send_again = 1
```

## 【(67) ～ (70) の選択肢】

- (1) 装置 A   (2) 装置 B   (3) 装置 C   (4) 装置 D
- (5) 装置 A～装置 D のいずれでもない

(計算用紙)



(計算用紙)