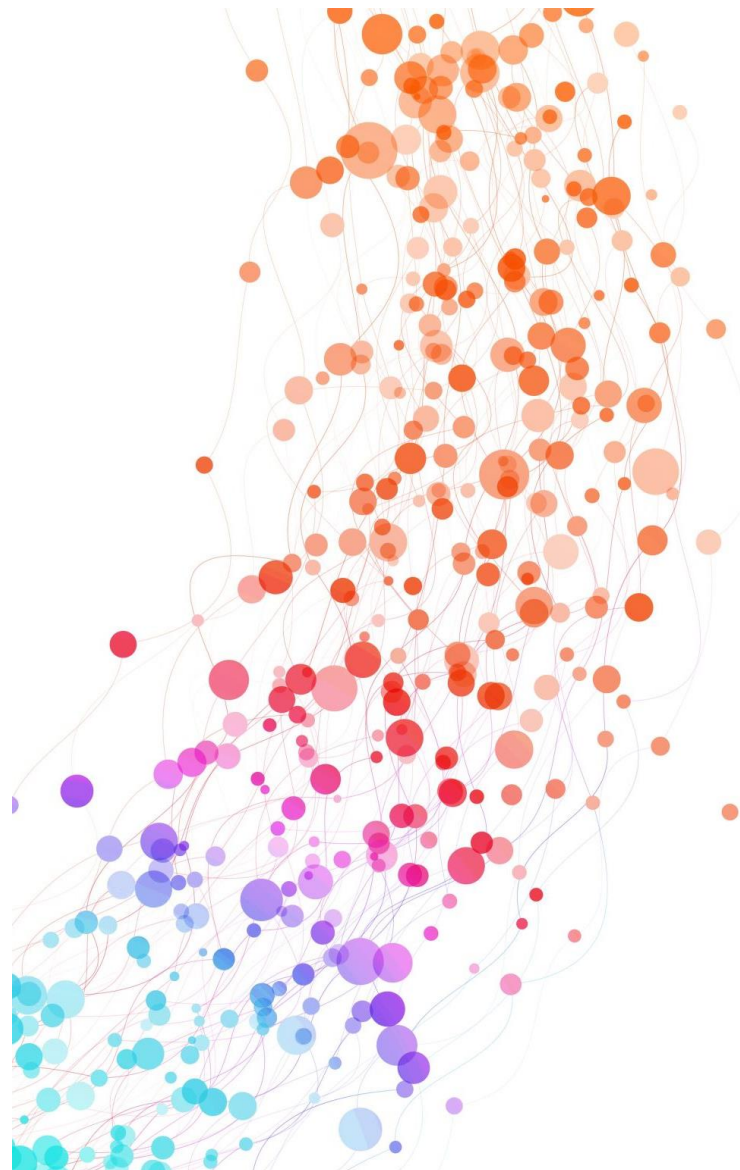


プログラミング 問題

3年情報



時給1500円の仕事について、累積の勤務時間が100時間以上になると全勤務の時給が1割増えるプログラム

(1) tanka=1500、zikan=【外部からの入力】

(2) もしzikan ならば：

(3) tanaka=

(4) 表示する（“時給”、tanak、“ 円”）

(1) tanka=1500、zikan=【外部からの入力】

(2) もしzikan ならば：

(3) tanaka=

(4) 表示する（“時給”、tanak、“” 円”）

(1) tanka=1500、zikan=【外部からの入力】は
tanakという変数の箱に1500が入っている。

【外部からの入力】は200を入れる（数字はなんでもいい）

(1) tanka=1500、zikan=【外部からの入力】

(2) もし zikan 100ならば:

(3) tanaka=

(4) 表示する (“時給”、tanak、“ 円”)

(2) 問題文は勤務時間が100以上になると単価が増えると言っている
なのでもしzikan(ここでは勤務時間)が100以上になれば単価が増える
今回の場合200をいれているので下のイの方に進む

答え ① > =

(1) tanka=1500、zikan=【外部からの入力】

(2) もしzikan 100ならば：

(3) tanaka=

(4) 表示する（“時給”、tanak、“円”）

(3) 上のアで100以上になった場合単価が1割増える。

答え ⑤tanka*1.1

外部から入力された数が偶数か奇数かを判定して表示するプログラム

(1) kazu=【外部からの入力】

(2) もしkazu== 0ならば：

(3) | 表示する（“偶数”）

(4) そうでなければ

| 表示する（“奇数”）

(1) kazu=【外部からの入力】

(2) もしkazu== 0ならば：

(3) | 表示する（“偶数”）

(4) そうでなければ

| 表示する（“奇数”）

(1)kazuに適当に偶数になる4を入れる

(1) kazu=【外部からの入力】

(2) もし kazu== 0ならば：

(3) | 表示する（“偶数”）

(4) そうでなければ

| 表示する（“奇数”）

(2) 4になにかをして0と等しければ偶数と判定される

① 2で割る(÷)(/)と2なので0にならない

② 2で割ると余りが0になる

答え ② kazu%2

- 料金に応じて映画料金を表示するプログラムで外部からの年齢として「16」が入力された時画面に表示されるものを、解答群から一つ選べ。

- (1) `nenrei` = 【外部からの入力】
- (2) もし `nenrei` < 12 ならば:
 - (3) | 表示する (“子供料金”)
- (4) そうでなくもし `nenrei` < 60 ならば:
 - (5) | 表示する (“大人料金”)
- (6) そうでなければ:
 - (7) | 表示する (“シニアd料金”)

- (1) nenrei=【外部からの入力】
- (2) もしnenrei <12 ならば:
- (3) | 表示する(“子供料金”)
- (4) そうでなくもしnenrei<60ならば:
- (5) | 表示する(“大人料金”)
- (6) そうでなければ:
- (7) | 表示する(“シニア料金”)

●外部からの入力としてnenreiに「16」が入る

上の条件に当てはまるのはそうでなくもしnenrei<60ならば:である

答え ① 大人料金

- 次の数当てゲームのプログラムで正答が「12」のとき、外部から正答より小さい数が入力されたら「もっと上」、大きい数が入力されたら「もっと下」、正答が入力されたら「当たり」と画面に表示するプログラム

(1) kotae=12, kazu=【外部からの入力】

(2) もしkazu kotaeならば:

(3) | 表示する(“もっと上”)

(4) そうでなくもしkazu kotaeならば:

(5) | 表示する(“もっと下”)

(6) そうでなければ:

(7) | 表示する(“当たり”)

(1) kotae=12, kazu=【外部からの入力】

(2) もしkazu kotaeならば:

(3) | 表示する(“もっと上”)

(4) そうでなくもしkazu kotaeならば:

(5) | 表示する(“もっと下”)

(6) そうでなければ:

(7) | 表示する(“当たり”)

(1) 問題文より外部からの入力が小さいと「もっと上」となる



kazu (外部からの入力) がkotaeより小さいと「もっと上」となる 答え ③ <

(1) kotae=12, kazu=【外部からの入力】

(2) もしkazu kotaeならば:

(3) | 表示する(“もっと上”)

(4) そうでなくもしkazu kotaeならば:

(5) | 表示する(“もっと下”)

(6) そうでなければ:

(7) | 表示する(“当たり”)

(1) 問題文より外部からの入力が大きいと「もっと下」となる



kazu (外部からの入力) がkotaeより大きいと「もっと下」となる

答え ① >

2進数→10進数の変換

15

① 1001

桁の重み (2^3) (2^2) (2^1) (2^0)

×

×

×

×

1

0

0

1



8

+

0

+

0

+

1

答え 9

10進数→2進数の変換

16

① 29 2) 29

2) 14

2) 7

2) 3

2) 1

0

• • • 1
• • • 0
• • • 1
• • • 1
• • • 1



答え

11101

(2)



0 1 2 3 4 5 6 7 8 9 A B C D E F の16個の数を使用。

16 ^{くらい} 倍ずつ位が上がる。

- 「9」の次は「10」ではなく、「A」を用いる
- 1つのケタの最大の数「F」の次にケタが上がり、「10」となる

10進数



16進数



16進数→10進数の変換

18

①A3

桁の重み (16¹) (16⁰)

×

×

A

3



$$16^1 \times 10 + 16^0 \times 3$$

答え 163

16進数→10進数の変換

19

① 173

$$16 \overline{) 173}$$

$$16 \overline{) 10} \quad \cdot \quad \cdot \quad \cdot$$

0

• • •

13

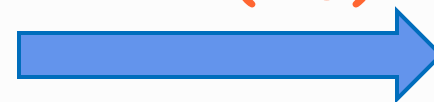
10

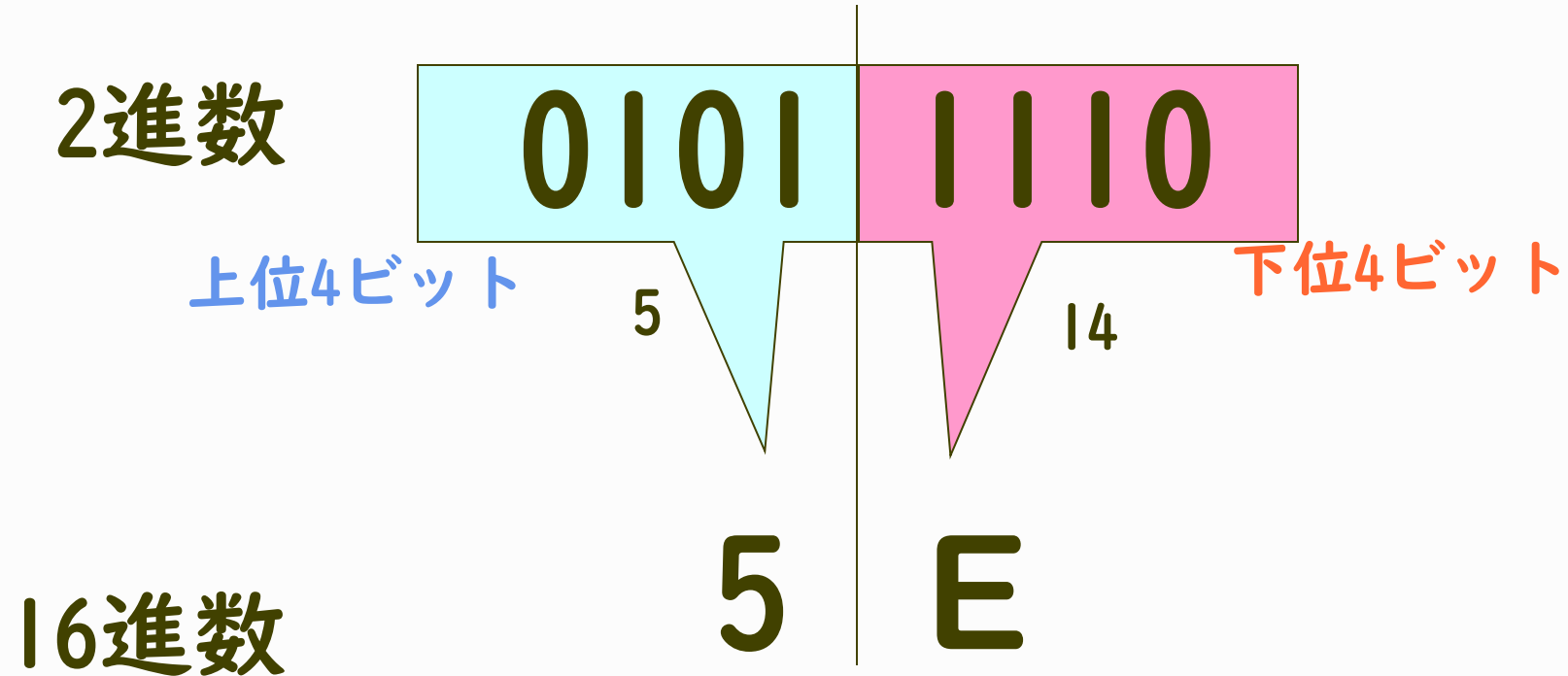


答え

AD

(16)





- ポイントは4ビットずつ分ける

次の中でコンピューターが計算できるものはどれ？

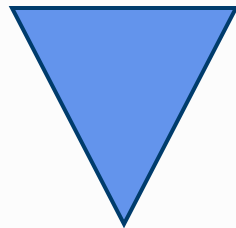
21

① $3+6$

② 4×3

③ $7-2$

④ $10\div 5$



① $3+6$ のみ

コンピューターは足し算しかできない

- 引き算や掛け算を全て足し算になおしてから計算しています。
- コンピュータの計算は論理回路の組み合わせで実現されています。

論理回路を複雑にすると計算スピードが落ちるので、究極のシンプルな形を追い求めこうなりました

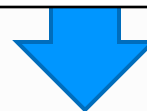
- 補数とは・・・元の数を足したときに桁上がりする最小の数
のことを指しています

例) 10進法における4に対する10の補数は6、
23に対する10の補数は77

$$\textcircled{1} 7 - 2$$

2に対する
補数は8

$$\text{10進法：} \underbrace{7}_{\text{補数}} - 2 = \underbrace{7}_{\text{補数}} + 8 = \cancel{1}5$$

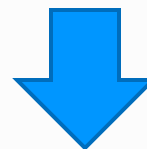


最上位である1(桁上がり部分)
を取り除き5

$$\textcircled{1} 5 - 3$$

3に対する
10の補数は7

足し算を使った式： $5 + 7 = 12$

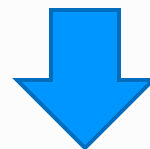


最上位である1を取り
除き2

$$\textcircled{2}8 - 4$$

4に対する
10の補数は6

足し算を使った式： $8+4=14$



最上位である1を取り
り除き4

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

Diagram illustrating binary addition. The first row shows the numbers 011 and 010 being added. The second row shows the result 1100. Blue arrows indicate carries: from the first column (0+0=0), from the second column (1+1=10, carry 1), and from the third column (1+0+1=10, carry 1). The final result is 1100.

☆ポイント
 $1 + 1 = 10$

補数の求め方について

27

例) $7 - 2$

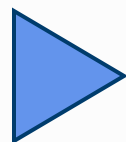
$$10\text{進法} : 7 - 2 = 7 + \underset{\text{補数}}{8} = 15$$

10進法で2の補数は8になります

10進法で補数を求める方法は $10 - 2$ をすれば求まります。

2進法でも同じように引き算をすれば求まります。

$7 - 2$ を2進法にすると



$$2\text{進法} : 0111_{(2)} - \underline{0010}_{(2)}$$

2進法で補数を求める方法は $10000_{(2)} - 0010_{(2)}$ をすれば求まります。

でもこれっておかしくないですか？

2の補数の求め方

28

例) 0 1 0 1 の場合

0 0 0 1

1 1 1 0

1 1 1 1

手順①
ビットを反転
(1の補数)

手順②
1を加算

この場合
下位4ビットだけとる(桁上がりは無視)

① 0 0 1 1 の場合

0 0 1 1 ⁽²⁾

1 1 0 0

1 1 0 1



手順①
ビットを反転
(1の補数)

手順②
1を加算

②00111000の場合

0 0 1 1 1 0 0 0 ⁽²⁾

1 1 0 0 0 1 1 1

1 1 0 0 1 0 0 0



手順①
ビットを反転
(1の補数)

手順②
1を加算

2の補数表現使った足し算で求める方法

31

$$\textcircled{1} 0100_{(2)} - \underline{0011}_{(2)}$$

手順①

右側の2進法の補数を求める

0011の補数を求める

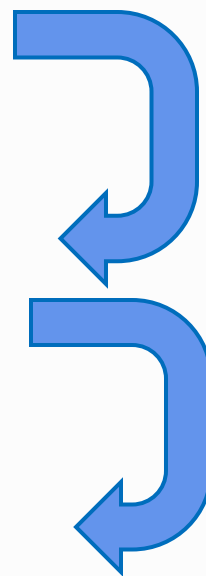
32

●0011の場合

0 0 1 1 ₍₂₎

1 1 0 0 ₍₂₎

1 1 0 1 ₍₂₎



①ビットを反転
(1の補数)

②1を加算

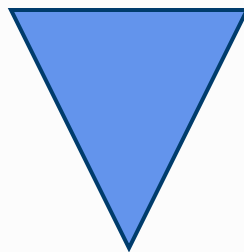
2の補数表現使った足し算で求める方法

33

$$\textcircled{1} 0100_{(2)} - \underline{0011}_{(2)}$$

手順①

右側の2進法の補数を求める



$$1101_{(2)}$$

2の補数表現使った足し算で求める方法

34

$$\textcircled{1} 0100_{(2)} + 1101_{(2)}$$

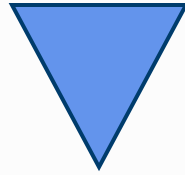
$$= 10001_{(2)}$$

手順③

下位4ビットだけとる

$$= 0001_{(2)}$$

$$\textcircled{2}0111_{(2)} - \underline{0100}_{(2)}$$



補数にすると・・・

$$1100_{(2)}$$

$$\textcircled{2} 0111_{(2)} + 1100_{(2)}$$

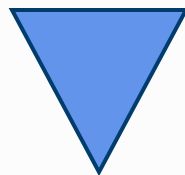
$$= 10011_{(2)}$$

手順③

下位4ビットだけとる

$$= 0011_{(2)}$$

$$\textcircled{2}0110_{(2)} - \underline{0001}_{(2)}$$



補数にすると・・・

$$1111_{(2)}$$

$$\textcircled{2}0110_{(2)} + 1111_{(2)}$$

$$= 10101_{(2)}$$

手順③

下位4ビットだけとる

$$= 0101_{(2)}$$

4 ビットでは・・・

1 番上位のビット（先頭のビット）が

0 のとき→ ②正

1 のとき→ ③負

4 ビットで表される

最大値→ 7

最小値→ -8

この先頭のビットを

①符号ビット

- 3ビットの場合もある

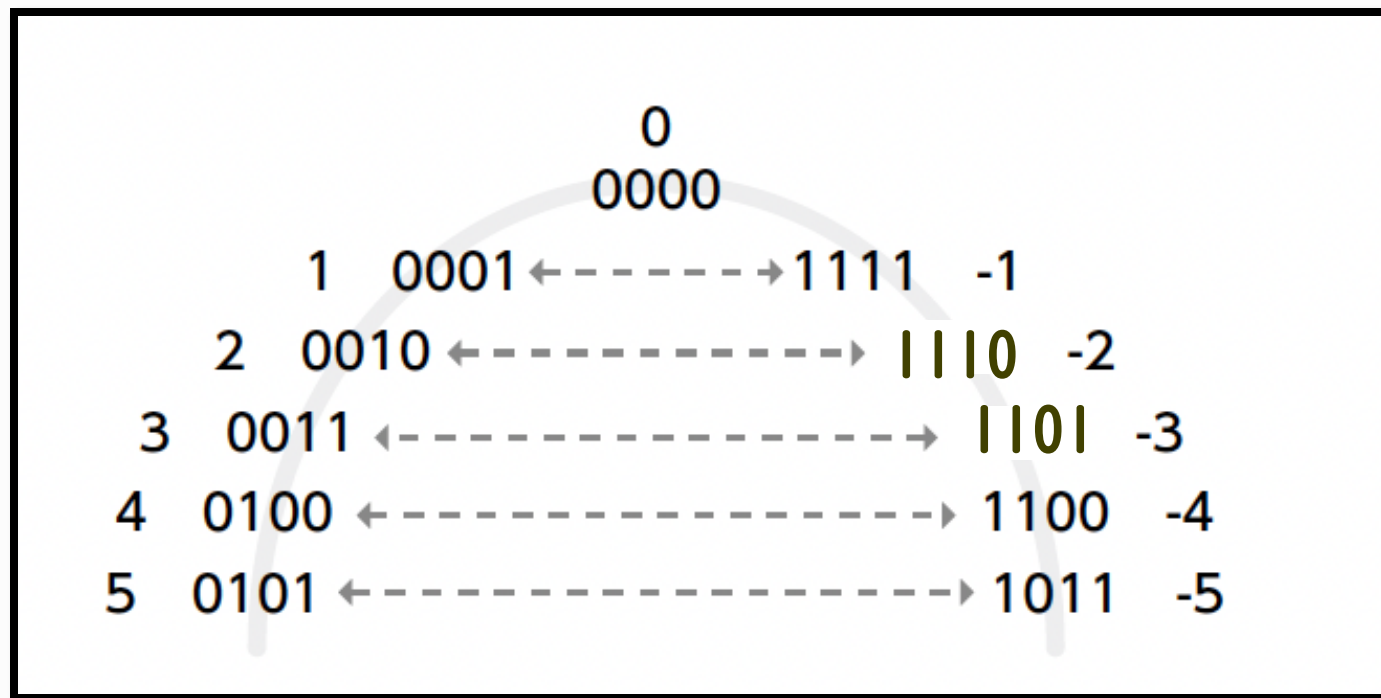
2進法表現	2の補数表現での数値	符号なし整数での数値
5 4321		
1 0111	7	7
1 0110	6	6
1 0101	5	5
1 0100	4	4
1 0011	3	3
1 0010	2	2
1 0001	1	1
1 0000	0	0
0 1111	-1	15
0 1110	-2	14
0 1101	-3	13
0 1100	-4	12
0 1011	-5	11
0 1010	-6	10
0 1001	-7	9
0 1000	-8	8

先頭のビット

表5 整数の2の補数表現

なぜ符号ビットを使う？

40



1 を補数変換をして

-1 にするには

0 と 1 を反転して

1 を足す

「1110」の表現するとこれが「-2」か「14」を表す数なのかわからない。そこで「符号付きビットで表現（2の補数表現）」のように断り書き付くことが多い。

注意点

- 10進数から2進数に変換をして答えを出しても〇〇ビットでと
指定が入る時がある

- 例 29を2進数に直すと11101

これは5ビット(1と0が5つ並んでいる)


これを7ビットで表現しなさいと言われるとどうするか？



先頭に0を付け加えて7ビットにする

0011101 (2)

●1の2進数表記は？

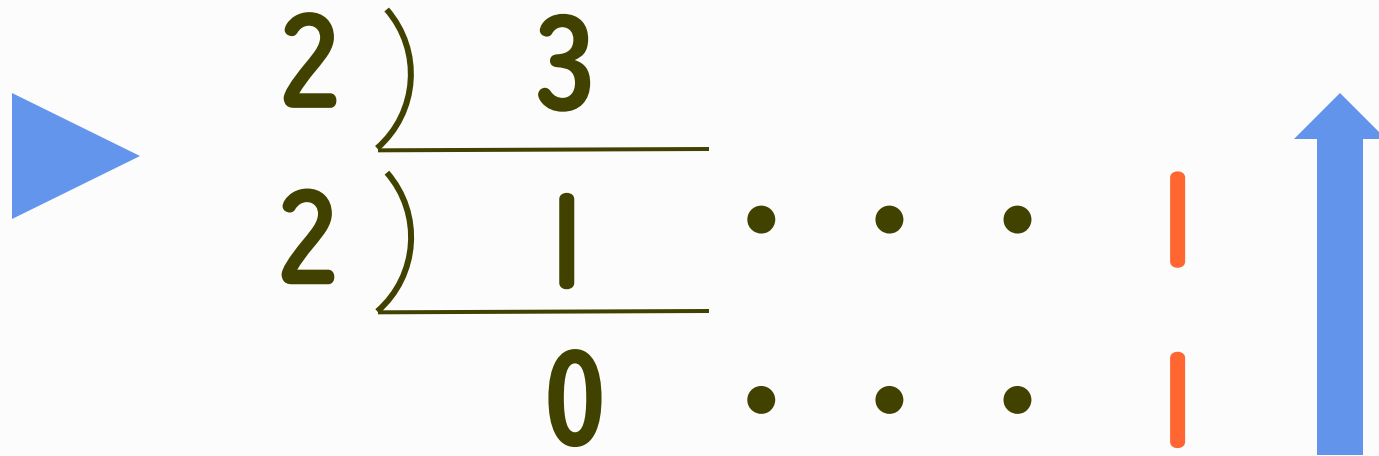


$$\begin{array}{r} 2 \overline{) 1} \\ 0 \end{array} \dots$$

●答えは1だが、3ビット表されるとあるので
無理やり0を1の前に足して3ビットにする

答え ①001

●3の2進数表記は？

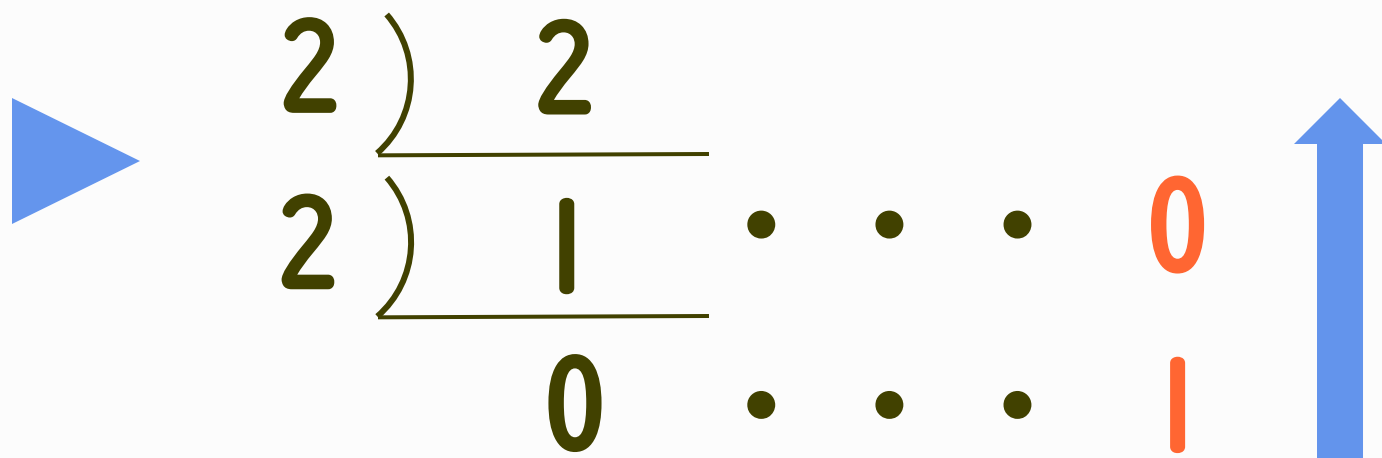


$$\begin{array}{r} 2 \overline{) 3} \\ 2 \overline{) 1} \dots \dots \dots \\ 0 \dots \dots \end{array}$$

●力同様に無理やり0を1の前に足して3ビットにする

答え ③ 011

- 手順としてはまず2を3ビットの2進数にする


$$\begin{array}{r} 2 \overline{) 2} \\ 2 \overline{) 1} \end{array} \begin{array}{c} \cdot \cdot \cdot \\ \cdot \cdot \cdot \end{array} \begin{array}{c} 0 \\ 1 \end{array}$$

- カ、キ同様に無理やり0を1の前に足して3ビットにする

答え 010

●次に010 (2) を-2にする方法を考える



2を-1にするには

① 0と1を反転

② 1を足す

答え 110

- 110は符号ビットを使ってーを表現している
ここで問われているのは符号ビットを使わずに
10進数に変換する

桁の重み (2^2) (2^1) (2^0)

× × ×

1 1 0



4 + 2 + 0

答え 6

- 4ビット目と3ビット目がオンである。
コンピュータではオフを0、オンを1と表現する。
- 別の解き方として
4ビット目と3ビット目が生きていて、それぞれの数字が8と4。
合計すると12である。これを2進数になおす。

答え 1100

- まず1から16までの数字を思い浮かべてと指示している
- 実際にカードにあるのは1から15までの数字



- カードにならないとなる数字となると16が当てあまる

答え ②

- 今回のカードゲームでは4枚のカードを用意している。
その上で1～16までの数字を当てる。
一番大きい数字は $2^4=16$ と考える
- 同じ考えで 2^7 で一番大きい数字を求めることができる

答え ⑤

- まず32ビットあるものを8ビットずつ区切る。



10101100

00010000

00001010

10110100



- 区切ったものを10進数に変換する。



172.

16.

10.

180

答え ③

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

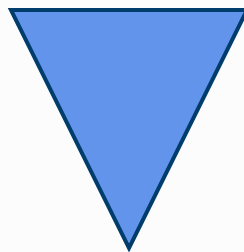
☆ポイント
 $1 + 1 = 10$

答え ⑤

$$11001_{(2)} - \underline{10101}_{(2)}$$

手順①

右側の2進法の補数を求める

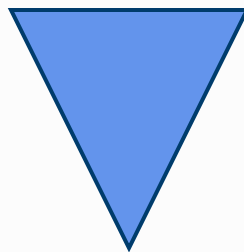


$$01011_{(2)}$$

$$11001_{(2)} - \underline{10101}_{(2)}$$

手順①

右側の2進法の補数を求める



$$01011_{(2)}$$

$$11001_{(2)} + 01011_{(2)}$$

$$= 100100_{(2)}$$

手順③

下位4ビットだけとる

$$= 00100_{(2)}$$

コ 補数とは・・・元の数を足したときに桁上がりする最小の数
のことを指しています

- 10進法における4に対する10の補数は6、
23に対する10の補数は77

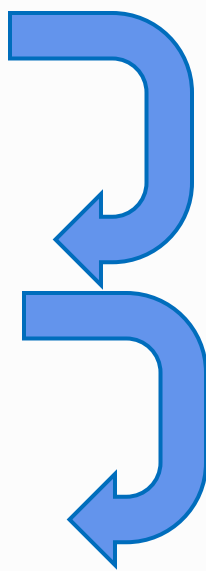
答え ②

●2の補数の求め方は

0 0 0 1

1 1 1 0

1 1 1 1



手順①
ビットを反転
(1の補数)

手順②
1を加算

答え ⑦

●10011100の補数の求め方は

10011100

01100011

01100100

手順①
ビットを反転
(1の補数)

手順②
1を加算

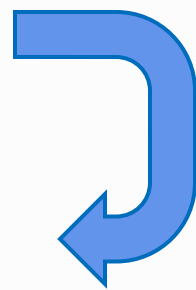
答え ④

●1011の補数の求め方は

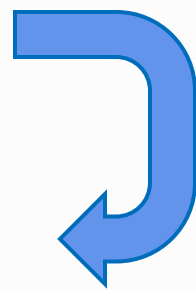
1011

0100

0101



手順①
ビットを反転
(1の補数)



手順②
1を加算

答え ②

$$\begin{array}{r} 0101 \\ +) 1111 \\ \hline 10100 \end{array}$$

答え ⑥

The diagram illustrates the propagation of a carry in binary addition. It shows two rows of bits: the top row is 0 1 0 1 and the bottom row is 1 1 1 1. Above the top row, four blue curved arrows point from right to left, representing the carry moving through the columns. Below the bottom row, a horizontal line separates it from the result row. The result row shows the first bit as a blue '1' with a diagonal slash through it, followed by four orange bits: 0, 1, 0, 0.

答え ①