

問1 次の各設問に答えよ。(34点)

設問1 変数を次のように宣言、初期化した。(各1点)

```
boolean b0, b1, b2, b3, b4, a=true, b=false, c=true;
```

各文を実行した後の変数 b0~b4 の値 (true/false) を答えよ。

(1) b0 = !!a; (※演算子の優先順位は2枚目に印刷)

(2) b1 = !a | b;

(3) b2 = a | b && c;

(4) b3 = (!a && b) | c;

(5) b4 = !(a | b) && (c ? false : true);

答(1) true (2) false (3) true (4) true (5) false

設問2 次は10個の整数値が格納された1次元配列から最小値を変数 min に求めるコードである。空欄を適切に埋めてコードを完成させなさい。(各1点)

```
int[] data={5,3,2,8,12,9,7,1,15,13};
int i, min;
min=data[ ];
for(i=1;i< ; ){
    if(data[ ]<min)min=data[ ];
}
```

答(1) 0 (2) data.length 又は 10 (3) i++ (4) i (5) i

※その他、目的が適切に達成される場合は正解とする。

設問3 次は2次元配列の初期化を行うコードである。下の図に数値を埋め、この配列を図示せよ。さらに、配列要素の無い場所は × (バツ) を記入せよ。(5点)

```
int[][] ary={{1,2,3},{4,5,6,7},{8,9},{10}};
```

答	1列目	2列目	3列目	4列目
1行目	1	2	3	×
2行目	4	5	6	7
3行目	8	9	×	×
4行目	10	×	×	×

設問4 次は3次元配列の初期化を行うコードである。下に示す各配列要素に代入される値を答えよ。配列要素が確保されない場合は × (バツ) を記入せよ。(各1点)

```
int[][][] ary={{1,2},{3},{4,5}},{6,7},{8,9,10}};
```

(1) ary[0][0][0] (2) ary[1][1][2]

(3) ary[1][2][0] (4) ary[0][2][1]

(5) ary[2][0][0]

答(1) 1 (2) 10 (3) × (4) 5 (5) ×

設問5 次は基本型変数と参照型変数の違いを確認するコードである。A地点とB地点での各変数の値を答えよ。(各1点)

```
int var_a=1, var_b;
int[] ary_a={2,3,1}, ary_b;
var_b = var_a;
ary_b = ary_a;
var_a = 2;
ary_a[2] = 5;
// ← (A地点)
ary_a[1] = ary_b[2];
ary_a[0] = ary_b[1];
ary_a[2] = ary_b[0];
// ← (B地点)
```

答 (A地点)

変数 var_a : 2, 変数 var_b : 1

配列 ary_a [0] : 2, ary_a [1] : 3, ary_a [2] : 5

配列 ary_b [0] : 2, ary_b [1] : 3, ary_b [2] : 5

(B地点)

配列 ary_a [0] : 5, ary_a [1] : 5, ary_a [2] : 5

配列 ary_b [0] : 5, ary_b [1] : 5, ary_b [2] : 5

問2 次の説明文の空欄に選択肢から適切な語句を選び、説明文を完成させよ。同じ番号の欄には同じ語句が入る。(26点)

説明文1 制御構造に関する説明文 (各2点)

Javaで (1) 構造は、for文とwhile文、do-while文を用いて作成する。これらは (1) 条件の評価のタイミングにより2つに分類される。for文とwhile文は (2) であり、do-while文は (3) である。 (2) では処理の本体 (ブロック) 部分が (4) 。一方、 (3) では (5) 。

[空欄(1)(2)(3)の選択肢]

A. 順次、B. 分岐、C. 繰返し、D. ネスト、E. break文、F. continue文、G. 前判定ループ、H. 中判定ループ、I. 後判定ループ、J. 無限ループ

[空欄(4)(5)の選択肢]

K. 一度も実行されない場合がある

L. 常に一度だけ実行される

M. 少なくとも一度は実行される

N. 常に一度も実行されない

答(1) C (2) G (3) I (4) K (5) M

説明文2 配列に関する説明文 (各2点)

配列は (1) 型の複数の変数を一括し管理する機能をもつ。配列を使用するには、まず (2) を宣言する。 (2) は、 (3) 変数の1つである。例えば、

```
int[][] ary;
```

とすればよい。ここで、aryは (4) 配列の (2) として宣言される。次に (5) を確保する。 (5) は値を格納するための領域であり、これを準備することである。例えば、

```
ary = (6) int[3][4];
```

とすればよい。このとき、この配列は (7) 個の (5) を持つことになる。各 (5) には (8) を用いて場所を指定し、ary[0][0]のように記述すれば値の読み書きができる。

[空欄(1)の選択肢] A. 同じ、B. 異なる

[空欄(3)の選択肢] C. 基本型、D. 参照型

[空欄(6)の選択肢] E. new、F. int、G. class

[空欄(7)の選択肢] (適切な数値を答えてください)

[その他の空欄の選択肢]

H. 配列変数、I. 配列要素、J. クラス変数、K. オブジェクト、L. 1次元、M. 2次元、N. 3次元、O. 添え字、P. 初期化、Q. .length 修飾子

答(1) A (2) H (3) D (4) M

(5) I (6) E (7) 12 (8) O

問 3 次の for 文に関する設問に答えよ。(8 点)

```
int i, sum;
for(i=0, sum=0; i<20; i+=2)
{
    sum+=i;
}
```

設問 1 上のコードを実行した直後の変数 sum の値を答えよ。(3 点)

答 90

設問 2 上のコードを次のように while 文を用いて書き直したい。上のコード中の下線で示したそれぞれの式、または文(下の選択肢に示す)を while 文の適切な位置に配置せよ。答えは一通りとは限りません。(各 1 点)

```
int i, sum;
(1);
(2);
while( (3) )
{
    (4);
    (5);
}
```

[選択肢]
A. i=0 B. sum=0 C. i<20 D. i+=2 E. sum+=i

答(1)A (又は B) (2)B (又は A) (3) C (4) E (5) D

問 4 コードの動作に関する設問に答えよ。(22 点)

設問 1 次の switch 文において変数 i の値を 0~4 で初期化した場合の画面出力を正確に答えよ。何も出力されない場合は×(バツ)を記入せよ。(各 2 点)

```
int i=初期化の値;
switch(i)
{
    case 1: System.out.println('a');
    case 2: System.out.println('b');
        break;
    case 3: System.out.println('c');
        break;
    case 4: System.out.println('d');
    default: System.out.println('e');
}
```

初期化の値	0	1	2	3	4
画面出力	e	a b	b	c	d e

設問 2 次の continue; 文を含むコードにおいて変数 c の値を 1~3 で初期化した場合の画面出力を正確に答えよ。何も出力されない場合は×(バツ)を記入せよ。(各 4 点)

```
int c=初期化の値;
for(int i=1; i<=6; i++)
{
    if(i%c==0)
    {
        continue;
    }
    System.out.println(i);
}
```

初期化の値	1	2	3
画面出力	×	1 3 5	1 2 4 5

問 5 8 ビットの 2 進数を 10 進数に変換するコードを作りたい。int[] 型の変数 bits に与えられた 2 進数の数値を、10 進数の数値として int 型の変数 dec に求めるコードを空欄を埋めて完成させよ。必要な変数は各自宣言して使用すること。上位ビット ← -- → 下位ビット (10 点)

```
int[] bits={0, 1, 1, 0, 0, 1, 0, 1};
int dec=0;
```

```
int multi=1;
for(int i=bits.length-1; i>=0; i--)
{
    dec += bits[i] * multi;
    multi *= 2;
}
<その他の解答例>
for(int i=0; i<bits.length-1; i++)
{
    dec += bits[i];
    dec *= 2;
}
dec += bits[bits.length-1];
```

System.out.println("10 進数で"+dec+"です");

《参考資料》 演算子の優先順位表

主な演算子の優先順位			
演算子	名前	結合規則	
++	後置インクリメント	左	↑ 同じ優先度 ↓
--	後置デクリメント	左	
!	論理否定	右	
~	1 の補数 (反転)	右	
+	プラス	右	
-	マイナス	右	
++	前置インクリメント	右	
--	前置デクリメント	右	
()	キャスト	右	
*	乗算	左	
/	除算	左	↑ 優先度高い ↓ 優先度低い
%	剰余	左	
+	加算 (文字列連結)	左	
-	減算	左	
<<	左シフト	左	
>>	右シフト	左	
>>>	符号なし右シフト	左	
>	より大きい	左	
>=	以上	左	
<	未満	左	
<=	以下	左	
==	等価	左	
!=	非等価	左	
&	ビット論理積	左	
^	ビット排他的論理和	左	
	ビット論理和	左	
&&	論理積	左	
	論理和	左	
?:	条件	右	
=	代入	右	
+=, -= など	複合代入演算	右	

お疲れ様でした。

情報工学部	情報工学科	年次	組	学籍 番号	氏名
					評点

福岡工業大学