

```

println("Result:");
for (int i = 0; i < 4; i++) {
    println(answer_route[i]);
}
}

int calcTotalDistance() {
    int total_distance = 0;
    for (int i = 0; i < 3; i++) {
        total_distance += distance(route[i], route[i+1]);
    }
    total_distance += distance(route[3], route[0]);
    return total_distance;
}

boolean checkConflict(int n, int c) {
    for (int i = 0; i < n; i++) {
        if (route[i] == c) return true;
    }
    return false;
}

// 以下に関数 checkRoute()ほか必要な関数の定義を記せ

```

## 2022年度 大学院入学試験問題

先端数理科学研究科 先端メディアサイエンス専攻(博士前期課程)

科目: 専門科目

## 注意事項

- 試験時間は90分です。
- 外国人留学生志願者のみ、英語による解答を認めます。
- 問題は、[I] 数学、[II] 情報、[III] プログラミングの3分野から出題されています。3分野のうち2分野を選択し解答しなさい。
- 選択した2分野の解答用紙の選択欄に「○」を、選択しなかった1分野の解答用紙の選択欄に「×」を、それぞれ必ず記入しなさい。選択欄にこれら以外の記載がある場合や、空欄がある場合、すべての答案を無効とします。
- 選択しなかった分野を含め、すべての解答用紙および問題用紙に受験番号と氏名を記入しなさい。
- 問題用紙は11ページあります(表紙は含まず)。
- 解答は、解答用紙に記入しなさい。
- 問題用紙、解答用紙は試験終了後すべて回収します。

受験番号 \_\_\_\_\_

氏 名 \_\_\_\_\_

2022

前期

二期

# [I] 数学

【問題1】行列  $A$  を

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

とする。  $P^{-1}AP$  が対角行列となるとき、そのときの  $P$  と  $P^{-1}AP$  を求めよ。

【問題2】

(1)  $D = \{(x, y) | x \leq 0, x^2 + y^2 \leq 3\}$  のとき、次の2重積分の値を求めよ。

$$\iint_D \frac{x}{(x^2 + y^2)(x^2 + y^2 + 1)} dx dy$$

(2) 関数  $f(x)$  を次で定める。

$$f(x) = \begin{cases} \frac{\sin x}{x} & (x \neq 0) \\ 1 & (x = 0) \end{cases}$$

(i)  $f'(x)$  を求め、  $f'(0)$  と  $\lim_{x \rightarrow 0} f'(x)$  とを比較せよ。

(ii)  $f(x)$  の  $x=0$  におけるテイラー展開を2次の項まで求めよ。

【問題3】6点からなる連結な単純グラフで3-正則グラフは同型を除いて2つあることが知られる。それらを図示せよ。ただし、3-正則グラフとはすべての点の次数が3であるグラフである。

# [II] 情報

【問題1】

スタックとキューに関する以下の問いに答えよ。

(1) スタックについて説明する以下の(ア)、(イ)、(ウ)に入る言葉を答えよ。

スタックは、格納されたデータを取り出す際、最後に格納したデータが最初に取り出される。すなわち、入出力形式として(ア)型のデータ構造を有する。スタックにデータを格納する操作を(イ)、取り出す操作を(ウ)と呼ぶ。

(2) すでに A, A, C, B の順序でデータが格納されているキューに、デキュー (Dequeue) 2回、B, C の順でエンキュー (Enqueue) した。その後4回デキュー (Dequeue) した際に、データはどの順序で取り出されるか答えよ。

(3) テキストエディタの実装において Undo 機能を実装する場合、スタックとキューのどちらが適しているか、理由を含めて答えよ。必要に応じて図表を用いても良い。

【問題2】

ハフマン符号化に関する以下の問いに答えよ。

(1) 文字列が A, B, C, D からなる4種類の文字で構成され、文字列の中での出現確率が以下の表に示すとおり A, B, C, D それぞれで 50%, 40%, 7%, 3% であった。ここで、各文字に割り当てる符号語 (ビット列) をそれぞれ答えよ。なお、符号化プロセスに用いる木構造も同時に示すこととする。

文字の種類	A	B	C	D
出現確率 (%)	50	40	7	3
符号語 (ビット列)				

(2) (1) で求めた符号語で 1000 文字の文字列の符号化を試みた。ここで、1000 文字の文字列の出現確率を算出したところ、A, B, C, D それぞれで 50%, 40%, 8%, 2% とズレがあった。このズレが符号化効率にどのような影響を与えるか、理由を含めて答えよ。

2027

工  
期

工  
期

【問題3】

入力ビット  $a, b$  とキャリー  $c$  を足して和  $s$  とキャリー  $c'$  を出力する全加算器 FA を、 $(s, c') = FA(a, b, c)$  で表す。

- (1)  $FA(0, 1, 1)$  を求めよ。
- (2)  $c'$  の論理式を NAND だけを用いて構成せよ (回路図を示せ)。
- (3) 2つの全加算器 FA1, FA2 を用いて、2ビットの加算器  $A(a_1, a_0, b_1, b_0) = (s_2, s_1, s_0)$  を構成せよ (ブロック図や式を用いて記述する)。ただし、 $s_0, a_0, b_0$  を下位 (LSB) とする。

【問題4】

生体認証の結果が次の表で与えられている。

- (1) B についての他人受入率 FAR (False Acceptance Rate) を求めよ。
- (2) A についての本人拒否率 FRR (False Rejection Rate) を求めよ。
- (3) C の FRR を下げるにはどうしたら良いか述べよ。

被験者 \ 認証結果	A	B	C
A	8	2	0
B	1	9	0
C	3	3	4

【III】プログラミング

この問題では Processing という言語でプログラムが記述されている。Processing は Java を基にしたプログラミング言語である。以下のプログラムの仕様を前提に各設問に答えよ。ただし、すべての関数が設問および解答で使われるとは限らない。

- 関数 `setup()` はプログラム実行開始時に一度だけ呼び出される。
- 関数 `println(x)` は引数  $x$  で指定された内容を標準出力 (コンソール) に表示した後、改行する。
- 関数 `sqrt(x)` は  $x$  の平方根を求める。
- 関数 `floor(x)` は  $x$  の小数点以下を切り捨てた整数値を出力する。
- 整数同士の演算では、その計算結果の小数点以下の値は切り捨てられる。

Processing における主な算術演算子

- + 足す (加算)
- 引く (減算)
- \* 掛ける (乗算)
- / 割る (除算)
- % 割った数の余り (剰余)

### 【問題1】

長さ  $N$  の整数型配列  $a$  に、1 以上の整数からなる、重複のないデータ列が格納されている。この配列の中で 2 番目に大きい値を見つけてその値を出力するプログラムを作りたい。以下のプログラムの空欄①～③を埋め、これを完成させよ。ただし  $N$  は 2 以上とする。

```
int first = 0;
int second = 0;
for (int i = 0; i < N; i++) {
    if (a[i] > first) {
        ①
    } else if ( ② ) {
        ③
    }
}
println(second);
```

### 【問題2】

図 a のように 2 次元平面領域内に  $N$  個の点および 5 個の楕円がある。最も小さい楕円を  $e_1$  とし、以降小さい順に  $e_2, e_3, e_4, e_5$  とする。  $e_1$  で囲まれた領域側にある領域を  $sec1$  とし、  $e_1$  の外側かつ  $e_2$  の内側にある領域を  $sec2$ 、  $e_2$  の外側かつ  $e_3$  の内側の領域は  $sec6$  とする。図 b のように各楕円の長軸と短軸の交点は原点  $(0, 0)$  である。

ここで、各点が存在する領域の番号を標準出力に表示する関数 `dividePoints()` を作りたい。解答用紙の空白部分を埋めプログラムを完成させよ。ただし各点の  $x$  座標と  $y$  座標はそれぞれ配列  $px$  と  $py$  により与えられるものとする。

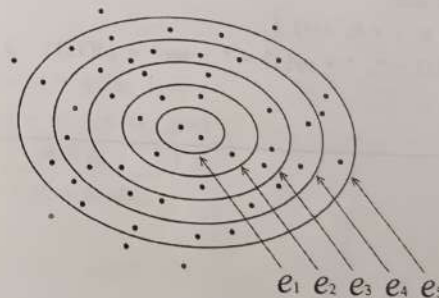


図 a

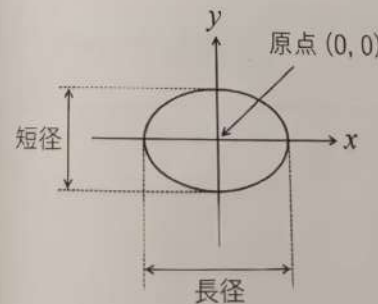


図 b

表：楕円のサイズ

楕円	長径	短径
$e_1$	150	100
$e_2$	300	200
$e_3$	450	300
$e_4$	600	400
$e_5$	750	500

```
// 配列 px, py にはそれぞれ N 個の値が入っているものとする
int[] px = {100, -23, ..., 390, -240};
int[] py = {-290, 142, ..., 15, 153};
```

```
void dividePoints() {
    int[] id = new int[N];
```

```
// 領域番号の表示
```

```
for (int i = 0; i < N; i++) {
    println(px[i] + ", " + py[i] + " : sec" + id[i]);
}
```

### 【問題 3】

以下はある模様を描かれた画像を生成するプログラムである。プログラム中に示される空欄に(A)～(H)の条件式を当てはめた際のそれぞれの実行結果を(ア)～(ク)の中から選べ。

なお, map は縦 10 マス・横 16 マスの模様データを格納した 2 次元配列である。また, 関数 createPattern() は map の内容をもとに模様を描かれた画像を生成する関数であり, 配列要素が 1 のときに白いマスを, 0 のときに黒いマスを画像上に描画する。

ただし, 実行結果の図ではマスがわかりやすいようにマスの境界線を黒で描画し, 黒いマスを濃いグレーで描画した。

```
int WHITE = 1, BLACK = 0;
int[][] map = new int[10][16];
```

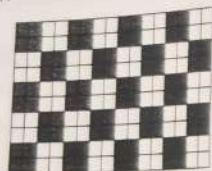
```
void setup() {
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 16; j++) {
            if (  ) {
                map[i][j] = BLACK;
            } else {
                map[i][j] = WHITE;
            }
        }
    }
    createPattern();
}
```

条件式

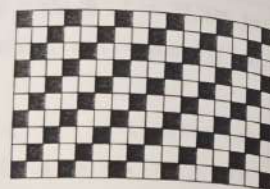
- |     |   |
|-----|---|
| (A) | $i \% 2 == 0$   |
| (B) | $j \% 2 == 0$   |
| (C) | $(i + j) \% 2 == 0$                                     |
| (D) | $(i + j) \% 3 == 0$                                     |
| (E) | $(\text{floor}(i / 2) + \text{floor}(j / 2)) \% 2 == 0$ |
| (F) | $(i * j) \% 2 + (i * j) \% 3 == 0$                      |
| (G) | $((i * j) \% 2 + (i * j) \% 3) \% 2 == 0$               |
| (H) | $((i + j) \% 2 + (i + j) \% 3) \% 2 == 0$               |

実行結果

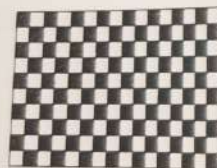
(ア)



(オ)



(イ)



(カ)



(ウ)



(キ)



(エ)



(ク)



#### 【問題4】

整数からなる十分に長い系列データ  $x_1, x_2, \dots$  に対し、最大で最新 1000 件までのデータの移動平均を求めるとき、データをリングバッファに格納して効率的に求めることを考える。

- (1) いま、関数 `retrieve()` は初回の呼び出しでは  $x_1$  を、2 回目の呼び出しでは  $x_2$  を、というように、呼び出すたびに最新の系列データを 1 件返すとする。リングバッファは配列 `buffer` と変数 `head` を用いて実装する。関数 `push()` は引数として与えられた値をリングバッファに格納した上で、バッファ中の値の総和を計算し、変数 `sum` に格納する。関数 `update()` は最新の系列データを取得した後に `push()` を用いてリングバッファを更新した後、最新 1000 件分のデータの平均値を `sum` を用いて計算し、変数 `moving_average` に格納する。次のページにそのプログラムを示す。

`push()` 中の空欄①では `head` の更新処理を行う。空欄①を埋めて `push()` を完成させよ。なお、`buffer` の各要素は 0 で初期化されているものとする。

```

int buffer_size = 1000;
int[] buffer = new int[buffer_size];
int head = 0;
int sum = 0;
int data_size = 0;
float moving_average;

void update() {
    push(retrieve());
    moving_average = (float) sum / data_size;
}

void push(int v) {
    buffer[head] = v;
    ①
    if (data_size < buffer_size) {
        data_size = data_size + 1;
    }
    sum = 0;
    for (int i = 0; i < data_size; i++) {
        sum = sum + buffer[i];
    }
}

```

- (2) (1)のプログラムはupdate() を呼び出すたびに1000件分のループを実行して総和を求めており、計算効率が悪い。push() を、ループを含まない、効率的なものに書き換えよ。解答欄にはpush()の実装のみ記すこと。

# 2021年度 先端数理科学研究科 過去問題 先端メディアサイエンス専攻 【閲覧用】

※ケータイ・スマホ等での撮影禁止

2021

工  
期

正  
期