

令和9年度 豊橋技術科学大学大学院工学研究科博士前期課程
入学者選抜学力検査問題

基 礎 科 目 （ 情 報 ・ 知 能 工 学 専 攻 ）

注 意 事 項

- 1 試験開始の合図まで、この問題冊子と解答用紙を開いてはいけません。
- 2 問題冊子の枚数は、表紙、草稿用紙を含めて10枚です。
- 3 問題冊子とは別に解答用紙が3枚あります。
- 4 問題は3問あります。全問解答してください。
- 5 試験開始の合図の後すぐに、すべての解答用紙の所定の箇所に受験番号を記入してください。
- 6 解答は必ず各問題別の解答用紙の所定の欄に記入してください。解答用紙の裏面には解答を記入しないでください。
- 7 落丁、乱丁、印刷不鮮明の箇所などがあれば、ただちに申し出てください。
- 8 問題冊子の余白は草稿用として使用しても構いません。
- 9 試験終了時刻まで退出してはいけません。
- 10 問題冊子は持ち帰ってください。

(草 稿 用 紙)

[1] 関数

$$f(x,y) = x^2 + 4xy + 4y^2 - 2x + 6y + 6$$

と、この関数による xy 平面上の 2 次曲線

$$f(x,y) = 0$$

を考える。以下の問いに答えよ。

- (1) 関数 $f(x,y)$ について、偏微分 $\frac{\partial f(x,y)}{\partial x}$, $\frac{\partial f(x,y)}{\partial y}$ をそれぞれ求めよ。
- (2) 直線 $x = -2y$ 上の点 P における関数 $f(x,y)$ の勾配ベクトル $\nabla f(P)$ を求めよ。
- (3) 2 次曲線 $f(x,y) = 0$ について、ベクトル $\mathbf{r} = \begin{pmatrix} x \\ y \end{pmatrix}$, 2×2 の対称行列 $\mathbf{Q}$, 2 次元ベクトル $\mathbf{s}$, スカラー C を用いて,

$$\mathbf{r}^T \mathbf{Q} \mathbf{r} + \mathbf{s}^T \mathbf{r} + C = 0$$
と表すとき、行列 $\mathbf{Q}$, ベクトル $\mathbf{s}$, スカラー C を求めよ。ここで記号 T はベクトルや行列の転置を表す。
- (4) 行列 $\mathbf{Q}$ の 2 つの固有値 λ_1, λ_2 とそれらに対応する固有ベクトル $\mathbf{u}_1, \mathbf{u}_2$ を求めよ。ただし、 $\lambda_1 \geq \lambda_2$ とし、固有ベクトルは第 1 要素を正とする単位ベクトルで求めよ。
- (5) 求めた固有ベクトル $\mathbf{u}_1, \mathbf{u}_2$ を並べた行列 $\mathbf{P} = (\mathbf{u}_1 \mathbf{u}_2)$ を作る。この行列 $\mathbf{P}$ を使って行列 $\mathbf{Q}$ を対角化し、対角化した行列を求めよ。
- (6) 2 次曲線 $f(x,y) = 0$ 上の点 $\mathbf{r}$ について、問(5)の行列 $\mathbf{P}$ を用いて $\tilde{\mathbf{r}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \mathbf{P}^T \mathbf{r}$ と変換する。この変換後の点 $\tilde{\mathbf{r}}$ が描く曲線を $\tilde{f}(\tilde{x}, \tilde{y}) = 0$ とするとき、元の曲線 $f(x,y) = 0$ から変換後の曲線 $\tilde{f}(\tilde{x}, \tilde{y}) = 0$ への変換について、以下の選択肢から最も適切なものを一つ選び、記号(ア)から(エ)で答えよ。

(ア) 平行移動, (イ) 直交変換, (ウ) 拡大縮小, (エ) せん断

- (7) この 2 次曲線 $f(x,y) = 0$ の種類について、以下の選択肢から最も適切なものを一つ選び、記号(カ)から(ク)で答えよ。

(カ) 楕円, (キ) 双曲線, (ク) 放物線

[2] 赤，白，青の3チームがゲームで総当たり戦を行い，2勝したチームを優勝とする。赤チームが白チームに勝つ確率は $\frac{2}{3}$ ，赤チームが青チームに勝つ確率は $\frac{1}{2}$ ，白チームが青チームに勝つ確率は $\frac{3}{4}$ であり，各試合の勝敗はその後の勝敗に影響しないものとする。引き分けはなく，毎試合必ず勝敗がつくとして，以下の問いに答えよ。ただし，答えが分数になるときは既約分数で答えよ。

- (1) 1回の総当たり戦において赤チームが1勝1敗となる確率を求めよ。
- (2) 1回の総当たり戦における白チームの勝数を確率変数 X とする。 $X=0$, $X=1$, $X=2$ の場合の確率 $P(X)$ をそれぞれ求めよ。
- (3) 1回の総当たり戦における白チームの勝数の期待値を求めよ。
- (4) 1回の総当たり戦において，すべてのチームが1勝1敗となる確率を求めよ。
- (5) 1回の総当たり戦において，すべてのチームが1勝1敗となった場合は再度総当たり戦を行うこととする。総当たり戦を最大3回としたとき，青チームが優勝する確率を求めよ。
- (6) 総当たり戦の回数を制限せず，優勝チームが決まるまで総当たり戦を行うとき，青チームが優勝する確率を求めよ。

[3] ハッシュテーブルに関する操作のC言語(ANSI C)による実装をプログラム1に示す。プログラム1の81行目を として、以下の問いに答えよ。

- (1) プログラム1の に、以下のリスト1を挿入する。実行時の標準出力への出力について、 , , に入る文字列を答えよ。

リスト1

```
initialize_table();
store_key_value("a", 10);
store_key_value("b", 20);
store_key_value("c", 30);
print_tables();
```

標準出力

```
0: [0] / key: a, val: 
1: [1] / key: b, val: 
2: [2] / key: c, val: 
3: [--] / key: ##, val: ##
4: [--] / key: ##, val: ##
```

- (2) プログラム1の に、以下のリスト2を挿入する。実行時の標準出力への出力について、 から に入る文字列を答えよ。

リスト2

```
initialize_table();
store_key_value("aa", 10);
store_key_value("ee", 40);
store_key_value("ac", 20);
print_tables();
```

標準出力

```
0: [0] / key: aa, val: 10
1: [0] / key: ac, val: 20
2: [] / key: , val: 
3: [] / key: , val: 
4: [] / key: , val: 
```

- (3) プログラム1の に、以下のリスト3を挿入する。実行時の標準出力への出力について、 から に入る文字列を答えよ。

リスト3

```
initialize_table();
store_key_value("aa", 10);
store_key_value("ee", 40);
store_key_value("ac", 20);
store_key_value("bb", 50);
store_key_value("ek", 30);
store_key_value("ef", 60);
print_tables();
```

標準出力

```
0: [0] / key: aa, val: 10
1: [0] / key: ac, val: 20
2: [] / key: , val: 
3: [] / key: , val: 
4: [] / key: , val: 
```

- (4) プログラム1の に、問(3)のリスト3に続けて、以下のリスト4を挿入する。実行時に問(3)の標準出力への出力に続いて、以下の標準出力が出力されるとき、 , に入る文字列を答えよ。

リスト4

```
int val;
val = get_value_by_key("ee");
printf("key:ee, value = %d\n", val);
val = get_value_by_key("eee");
printf("key:eee, value = %d\n", val);
```

標準出力

```
key:ee, value = 
key:eee, value = 
```

- (5) プログラム1では、テーブルの配列に空きがあっても新しい要素を入れられないことがある。そこで、テーブルの配列に空きがあったら必ず入れ、かつ、配列に空きがないときは、既に挿入されたデータを消さずにテーブルが表示されるように、プログラム1の39行目から41行目、および、プログラム1の60行目から62行目に対し同じ修正を行い、問(4)のプログラムを実行する。適切な修正を以下の表より選び、記号(A)から(D)で答えよ。

(A)	<pre>if(++index >= MAX_TABLE index == hash) { return -2; }</pre>
(B)	<pre>if(++index >= MAX_TABLE) index = 0; if(index == hash) { return -2; }</pre>
(C)	<pre>if(++index >= MAX_TABLE) index = 0;</pre>
(D)	<pre>if(++index != hash) { break; } if(index >= MAX_TABLE) { return -2; }</pre>

(6) プログラム1の ※ に，以下のリスト5を挿入する。

リスト5

```
initialize_table();
store_key_value("aa", 10);
store_key_value("ab", 20);
store_key_value("ac", 30);
store_key_value("ad", 40);
store_key_value("ae", 50);
int val;
val = get_value_by_key("ae");
```

ア．リスト5内の `get_value_by_key` 関数の実行中に `strcmp` 関数は何回呼び出されるか答えよ。

イ．先頭の文字が同じで，異なる文字列をキーとしたデータ N 個に対して `store_key_value` 関数を実行したとき，`get_value_by_key` 関数によりデータを取り出すときの計算量を以下の表より選び，記号(A)から(G)で答えよ。

(A) $O(1)$	(B) $O(\log N)$	(C) $O(N \log N)$	(D) $O(N!)$
(E) $O(N)$	(F) $O(N^2)$	(G) $O(2^N)$	

- (7) プログラム1の20行目を削除し、リスト6を挿入する。MAX_TABLE(プログラム1の12行目)を1000とし、a から z の26文字をランダムに組み合わせた長さ1以上、MAX_KEYLEN(プログラム1の11行目)以下の文字列をキーとしたときの、calc_hash_value関数にあてはまる特徴について、あてはまるものすべてを以下の表より選び、記号(A)から(F)で答えよ。

リスト6

```
unsigned int sum = 1;
for(int i = 0; i < len; i++) {
    sum = (sum * 4567 + (key[i] - 'a')) % MAX_TABLE;
}
return sum;
```

(A)	同じ入力文字列に対して、同じハッシュ値を返す。
(B)	異なる入力文字列に対して、必ず異なるハッシュ値を返す。
(C)	計算時間は文字列の長さに依存していない。
(D)	計算時間は文字列の長さに依存し、指数関数的に増える。
(E)	ある入力文字列では、MAX_TABLE以上の値を返す。
(F)	特定のハッシュ値に値がかたよる。

- (8) ハッシュ値が一様に分布する理想のcalc_hash_value関数を実装したとする。この関数を用いて、異なる文字列をキーとしたデータ N 個に対してstore_key_value関数を実行したとき、get_value_by_key関数によりデータを取り出すときの計算量を以下の表より選び、記号(A)から(G)で答えよ。

(A)	$O(1)$	(B)	$O(\log N)$	(C)	$O(N \log N)$	(D)	$O(N!)$
(E)	$O(N)$	(F)	$O(N^2)$	(G)	$O(2^N)$		

プログラム 1

```

001:#include <stdio.h>
002:#include <stdlib.h>
003:#include <string.h>
004:
005:typedef struct item
006:{
007:    char *key;
008:    unsigned int value;
009:    int hash;
010:} hash_item;
011:#define MAX_KEYLEN 10
012:#define MAX_TABLE 5
013:
014:hash_item hash_table[MAX_TABLE];
015:
016:unsigned int calc_hash_value(char *key)
017:{
018:    int len = strlen(key);
019:    if(len > 0 && len <= MAX_KEYLEN) {
020:        return (key[0] - 'a') % MAX_TABLE;
021:    }
022:    return MAX_TABLE;
023:}
024:void initialize_table()
025:{
026:    for(int i = 0; i < MAX_TABLE; i++) {
027:        hash_table[i].key = NULL;
028:        hash_table[i].hash = -1;
029:    }
030:}
031:int store_key_value(char *key, unsigned int value)
032:{
033:    unsigned int index, hash;
034:    index = hash = calc_hash_value(key);
035:    if(index >= MAX_TABLE) {
036:        return -1;
037:    }
038:    while(hash_table[index].key != NULL) {
039:        if(++index >= MAX_TABLE) {
040:            return -2;
041:        }
042:    }
043:    hash_table[index].key = (char *)malloc(strlen(key) + 1);
044:    strcpy(hash_table[index].key, key);
045:    hash_table[index].value = value;
046:    hash_table[index].hash = hash;
047:    return index;
048:}

```

プログラム1(前ページより続き)

```

049:int get_value_by_key(char *key)
050:{
051:    unsigned int index, hash;
052:    index = hash = calc_hash_value(key);
053:    if(index >= MAX_TABLE) {
054:        return -1;
055:    }
056:    while(hash_table[index].key != NULL) {
057:        if( strcmp(hash_table[index].key, key) == 0 ) {
058:            return hash_table[index].value;
059:        }
060:        if(++index >= MAX_TABLE) {
061:            return -2;
062:        }
063:    }
064:    return -3;
065:}
066:void print_tables()
067:{
068:    for(int i = 0; i < MAX_TABLE; i++) {
069:        if(hash_table[i].key != NULL) {
070:            printf("%d: [%d] / key: %s, val: %d\n",
071:                i, hash_table[i].hash,
072:                hash_table[i].key, hash_table[i].value);
073:        } else {
074:            printf("%d: [--] / key: ##, val: ##\n", i);
075:        }
076:    }
077:}
078:
079:int main()
080:{
081:    ※
082:}

```

注：関数 `size_t strlen(const char *s);`

文字列の長さを返す

注：関数 `int strcmp(const char *s1, const char *s2);`

文字列を比較し同じ文字列なら0を返す

注：関数 `char *strcpy(char *dst, const char *src);`

srcからdstへ文字列をコピーする

注：'a' は文字列内のaに対するASCII値 97 であり, 'b' は 98, 'c' は 99,
とアルファベット順に連続して値が増加し, 'z' は 122 である