知能システム学Ｉ

次の３問すべてを解答せよ．問題毎に別の答案用紙を用いること．答案用紙の追加は認めない．

1

以下の問い 1)〜4) に答えよ．

1) 以下の a)，b) の極限を求めよ．

a) $\displaystyle\lim_{x\to 0}\frac{\tan^2(\pi x)-\sin^2(\pi x)}{x^4}$

b) $\displaystyle\lim_{x\to\infty} x\left(2^{\frac{1}{x}}-1\right)$

2) 以下の a)，b) の重積分を求めよ．

a) $\displaystyle\iint_D y^2 dxdy,\quad D=\{(x,y)\mid x^2+y^2\le a^2, a>0\}$

b) $\displaystyle\iint_D xe^{xy}dxdy,\quad D=\left\{(x,y)\mid \frac{1}{2}\le y\le 1, 1\le x\le \frac{1}{y}\right\}$

3) 以下の実数値関数 $x_i=x_i(t)$ $(i=1,2,3)$ に関する連立微分方程式を考える．

$$\begin{cases}\frac{dx_1}{dt} &= -x_1+x_3\\ \frac{dx_2}{dt} &= ax_2+2x_3\\ \frac{dx_3}{dt} &= -2x_2+ax_3\end{cases}$$

この連立微分方程式の解 $x_1(t)$，$x_2(t)$，$x_3(t)$ が，任意の初期条件に対して
$\displaystyle\lim_{t\to\infty}x_1(t)=\lim_{t\to\infty}x_2(t)=\lim_{t\to\infty}x_3(t)=0$ となる実定数 a の条件を求めよ．

4) 実数値関数 $x=x(t)$，$y=y(t)$ を考える．初期条件を $x(0)=1$，$y(0)=1$ としたとき，以下の連立微分方程式の解が $t\to\infty$ で漸近する軌道 $(\bar{x}(t),\bar{y}(t))$ を求めよ．

$$\begin{cases}\frac{dx}{dt} &= (1-\sqrt{x^2+y^2})x-y\\ \frac{dy}{dt} &= x+(1-\sqrt{x^2+y^2})y\end{cases}$$

2

以下の問い 1)〜3) に答えよ．

1) 以下の行列 A, B の行列式を求めよ．ただし x は実数とする．

$$A = \begin{bmatrix} 1+x^2 & x & 0 & 0 \\ x & 1+x^2 & x & 0 \\ 0 & x & 1+x^2 & x \\ 0 & 0 & x & 1+x^2 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 1 & x+1 & 1 & 1 \\ 1 & x+1 & 1 & x+1 & 1 \\ x+1 & 1 & x+1 & 1 & x+1 \\ 1 & x+1 & 1 & x+1 & 1 \\ 1 & 1 & x+1 & 1 & 1 \end{bmatrix}$$

2) 一辺の長さが 1 の正六角形 ABCDEF（図 1）に関して以下の小問 a), b) に答えよ．

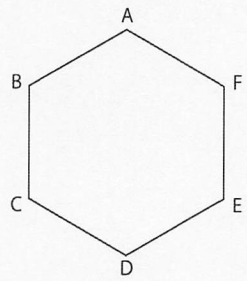

図 1: 正六角形 ABCDEF

a) 頂点 A と C を結ぶベクトルを $\boldsymbol{u}$，頂点 A と E を結ぶベクトルを $\boldsymbol{v}$ とする．ベクトル $\overrightarrow{AB}$ と $\overrightarrow{AF}$ を $\boldsymbol{u}$，$\boldsymbol{v}$ で表せ．

b) 点 P が辺 AB 上を，点 Q が辺 CD 上をそれぞれ独立して動くとき，線分 PQ を $z : 1-z$ $(0 \leq z \leq 1)$ に内分する点を R とする．この点 R が動く領域の面積が最大になるときの z とその面積を求めよ．

3) 点 $P((a\cos\theta + b\sin\theta)\cos\theta,\ (a\cos\theta + b\sin\theta)\sin\theta)$ $(0 \leq \theta < 2\pi)$ と点 $Q(2a,\ 2b)$ に関して以下の小問 a), b) に答えよ．ただし a，b は定数で $a > 0$，$b > 0$ とする．

a) 点 A を (a, b)，点 R を $(\cos\theta,\ \sin\theta)$ として，ベクトル $\overrightarrow{OP}$ を $\overrightarrow{OA}$ と $\overrightarrow{OR}$ で表せ．ただし点 O は原点とする．

b) 点 P と点 Q の距離が最小となるときの θ とその距離を求めよ．

3

以下の問い 1)，2) に答えよ．

1) ハッシュ表では，同じバケットに複数のキーが割り当てられる衝突が発生する場合がある．衝突解決法の一つとしてオープンアドレス法があり，その代表的な手法にダブルハッシュ法がある．ダブルハッシュ法では，2 つの異なるハッシュ関数を用いて衝突時の再探索位置を決定する．第 1 ハッシュ関数 h_1 は空いているバケットの初期探索位置を決め，第 2 ハッシュ関数 h_2 は，衝突発生後の再探索間隔を決定する．キー key に対して，衝突が i $(i = 0, 1, 2, ...)$ 回発生したときにアクセスされるバケットの index_i は，次の式で表される．

$$\text{index}_i = (h_1(key) + i \cdot h_2(key)) \mod \text{TABLE\_SIZE}$$

ここで，TABLE_SIZE はハッシュ表のバケット数を表す．インデックスが表の範囲外にならないよう，計算結果は TABLE_SIZE で割った余り（mod）を取る．プログラム 1 は，このダブルハッシュ法を用いたハッシュ表を実装した C 言語のプログラムである．各関数の役割は以下の通りである．`insert` 関数は，初期位置から開始し，衝突が発生した場合には，第 2 ハッシュ関数により決定される間隔でバケットをずらしながら，空きバケットまたは削除済みのバケットに値を挿入する．`search` 関数は初期位置から順にバケットを探索し，目的の値が見つかった場合はその位置（インデックス）を返す．見つからない場合，あるいは空きバケットに到達した場合は `-1` を返す．`delete_key` 関数は，`search` 関数で探索した結果の位置に `DELETED` (-2) を記録し，削除を行う．このプログラムに関して，以下の小問 a)~d) に答えよ．

a) `insert` および `search` 関数が正しく動作するようにするため，プログラム 1 中の [1] および [2] の部分に適切なコードを補え．なお，[2] は複数行の記述でもよいものとする．

b) プログラム 1 の第 30 行目の実行が終了した時点での `table` の状態を，以下の形式の表に記入せよ．解答用紙にはこの表を転記し，空欄をすべて埋めること．また，`EMPTY` は `-1`，`DELETED` は `-2` を用いて数字で表すこと．

バケットのインデックス	0	1	2	3	4	5	6	7	8	9	10
バケットの値											

c) プログラム 1 の第 31 行目が実行された際，`search` 関数内で調べられるバケットのインデックスを，探索される順に答えよ．

d) プログラム 1 の第 31 行目の直前に，`table[3]=EMPTY;` という行が挿入されたとする．その後に（元の）第 31 行目が実行されたとき，`search` 関数の返り値は何になるか答えよ．

ーーー 次ページに続く ーーー

ーーー 問題 3 の続き ーーー

プログラム 1

```
1  #include <stdio.h>
2  #define TABLE_SIZE 11
3  #define EMPTY -1
4  #define DELETED -2
5  int h1(int key) { return key % TABLE_SIZE; }
6  int h2(int key) { return 1 + (key % (TABLE_SIZE - 1)); }
7  void insert(int table[], int key) {
8      for (int i = 0; i < TABLE_SIZE; ++i) {
9          int index = (h1(key) + i * h2(key)) % TABLE_SIZE;
10         if ( [ 1 ] ) { table[index] = key; return; }
11     }
12 }
13 int search(int table[], int key) {
14     for (int i = 0; i < TABLE_SIZE; ++i) {
15         int index = (h1(key) + i * h2(key)) % TABLE_SIZE;
16         [ 2 ]
17     }
18     return -1;
19 }
20 void delete_key(int table[], int key) {
21     int index = search(table, key);
22     if (index != -1) table[index] = DELETED;
23 }
24 int main() {
25     int table[TABLE_SIZE];
26     for (int i = 0; i < TABLE_SIZE; ++i) { table[i] = EMPTY;}
27     insert(table, 33);
28     insert(table, 22);
29     insert(table, 47);
30     delete_key(table, 22);
31     int idx = search(table, 47);
32     if (idx != -1) { printf("Found 47 at index %d\n", idx); } else { printf("47 not
           found\n"); }
33     printf("Current table:\n");
34     for (int i = 0; i < TABLE_SIZE; ++i) { printf("[%2d]: %d\n", i, table[i]); }
35     return 0;
36 }
```

ーーー 次ページに続く ーーー

ーーー 問題 3 の続き ーーー

2) プログラム 2 は，BM 法（Boyer-Moore 法）を用いて，テキスト中からパターンを効率的に探索する C 言語のプログラムである．BM 法では，検索の効率化のために skip テーブルを用いる．このテーブルは，各文字に対してパターン内での最後の出現位置を記録し，不一致が発生した際にパターンを右に何文字シフトすべきかを決定する．不一致文字がパターン中に存在しない場合は，パターン全体の長さをスキップ量として設定する．プログラム 2 に含まれる関数は以下の通りである．str_len 関数は文字列の長さを返す．build_skip_table 関数は，上記の skip テーブルを構築する関数である．まずテーブルの全要素をパターン長で初期化し，次にパターン中の各文字についてその文字の最後の出現位置から末尾までの距離をスキップ量として記録する．bm_search 関数は，テキストとパターンを比較し，一致する位置を探索する．比較はパターンの末尾から行い，不一致が発生した場合は，skip テーブルに基づいてパターンを右にシフトすることで，比較回数を削減する．パターンが一致した場合は開始位置のインデックスを返し，一致しなかった場合は -1 を返す．このプログラムに関する以下の小問 a)～c) に答えよ．

a) build_skip_table および bm_search 関数が正しく動作するようにするため，プログラム 2 中の [3] および [4] の部分に適切なコードを補え．

b) プログラム 2 の実行において，pattern と text の最初の 3 回の比較位置を順に記述せよ．

c) プログラム 2 の main 関数が実行されたときの戻り値を答えよ．

プログラム 2

```
1  #include <stdio.h>
2  #define MAX_CHAR 256
3  typedef unsigned char uchar;
4  int str_len(uchar *s) { int len = 0; while (s[len] != '\0') len++; return len; }
5  void build_skip_table(uchar *pattern, int skip[], int m) {
6      int i; for (i = 0; i < MAX_CHAR; i++) { skip[i] = m; }
7      for (i = 0; i < m - 1; i++) { [          3          ] }
8  }
9  int bm_search(uchar *text, uchar *pattern) {
10     int skip[MAX_CHAR]; int m = str_len(pattern); int n = str_len(text); int i, j;
11     build_skip_table(pattern, skip, m); i = 0;
12     while (i <= n - m) {
13         j = m - 1; while (j >= 0 && pattern[j] == text[i + j]) { j--; }
14         if (j < 0) { return i; } else { [          4          ] }
15     }
16     return -1;
17 }
18 int main() {
19     uchar text[] = "ABC ABCDAB ABCDABCDABDE";
20     uchar pattern[] = "ABCDABD";
21     int pos = bm_search(text, pattern);
22     return pos;
23 }
```