北海道大学　大学院情報科学院

情報科学専攻　修士課程

情報理工学コース

専門科目１

１０：００～１２：００

受験上の注意

● 本冊子内の５問，問１（基礎数学），問２（情報数学），問３（確率・統計），問４（コンピュータ基礎工学），および問５（プログラミング）から３問を選択し解答すること．

● 選択問題チェック票に受験番号および，選択した科目に印を記入すること．

●すべての解答用紙に，受験番号，選択した問題番号(例えば，問３など)を必ず記入すること．

●解答用紙は３枚である．この他に下書き用の草案紙３枚を配付する．

●解答は，問題ごとに別々の解答用紙に記入すること(裏面を使用してもよい．解答用紙が不足したり，破損したりした場合には試験監督員に申し出ること)．

●解答が複数枚にわたる時は，1/2，2/2 のように解答用紙にページ番号を必ず付すこと，及び受験番号，選択した問題番号を各ページに記入すること．

●問題冊子，草案紙は持ち帰り，選択問題チェック票とすべての解答用紙を提出すること．

●机の上に置いてよいものは，筆記用具（黒鉛筆，消しゴム，鉛筆削り），時計，および特に指示があったもののみである．時計は計時機能のみを使用し，アラームの使用を禁ずる．携帯電話，スマートフォン，タブレット，コンピュータ等は電源を切ってかばんの中にしまうこと．電卓，電子辞書などは使用を禁ずる．

専門科目１

選択問題チェック票

１．　受験番号を記入すること．

２．　問１から問５のうち選択した３問について，以下の表中に○を記入し，選択した問題番号と解答用紙に記入した問題番号が一致していることを確かめること．

３．　本チェック票は解答用紙と一緒に提出すること．

受験番号	

問１　基礎数学		選択した３問に○を記入すること
問２　情報数学		
問３　確率・統計		
問４　コンピュータ基礎工学		
問５　プログラミング		

問 1. 基礎数学

［1］2変数関数に関する以下の問いに答えよ．

集合 D を $\mathbb{R}^2$ の開集合とし，関数 $f : D \to \mathbb{R}$ が2回連続微分可能であるとする．さらに，点 $\mathbf{x}_0 = (x_0, y_0)'$（$'$ は転置）で，勾配は零，つまり，$\nabla f(\mathbf{x}_0) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})'|_{x=x_0,\, y=y_0} = (0, 0)'$ であるとする．また，点 $\mathbf{x}_0$ でのヘッセ行列を

$$H(\mathbf{x}_0) = \left(\begin{array}{cc} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{array} \right) \Bigg|_{x=x_0,\, y=y_0}$$

とする．

(1) 関数 f がある点で極値（極大値あるいは極小値）を取るならばその点で f の勾配が零となること，しかし，逆は成り立たないことを示せ．

(2) 点 $\mathbf{x}_0$ でのヘッセ行列 $H(\mathbf{x}_0)$ が正定値であれば，f は $\mathbf{x}_0$ で（狭義）極小値を取ることを証明せよ．ここで，行列 A が「正定値」であるというのは，任意の非零ベクトル $\mathbf{x}$ に対して，$\mathbf{x}'A\mathbf{x} > 0$ なることである．

［2］ベクトルと行列に関する以下の問いに答えよ．

$$\mathbf{a} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 2 \\ 8 \\ 10 \end{pmatrix}, \mathbf{c} = \begin{pmatrix} 3 \\ 10 \\ 13 \end{pmatrix}, \mathbf{d} = \begin{pmatrix} -2 \\ 0 \\ 1 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

(1) 3次元ベクトル $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ の中から重複を許して3つ選んで横に並べ，階数（ランク）が1，2，3となる正方行列 A_1, A_2, A_3 をそれぞれつくれ．

(2) それら3つの行列 A_1, A_2, A_3 のうちどれが方程式 $A_i\mathbf{x} = \mathbf{0}$ $(i = 1, 2, 3)$ において非自明な解 $\mathbf{x} \neq \mathbf{0}$ を持つか，理由とともに述べよ．

(3) 3つのベクトル $\mathbf{a}, \mathbf{b}, \mathbf{c}$ を横に並べてできる正方行列 $B = (\mathbf{a}\,\mathbf{b}\,\mathbf{c})$ に対して B の零空間 $\mathrm{Ker}(B) = \{\mathbf{x} \in \mathbb{R}^3 \mid B\mathbf{x} = \mathbf{0}\}$ の非零要素を一つ挙げよ．さらに，$\alpha, \beta \in \mathbb{R}$ に対して，$\mathbf{x}, \mathbf{y} \in \mathrm{Ker}(B) \to \alpha\mathbf{x} + \beta\mathbf{y} \in \mathrm{Ker}(B)$ を示せ．

(4) 前問と同じ正方行列 B に対して，方程式 $B\mathbf{x} = \begin{pmatrix} -1 \\ -2 \\ -3 \end{pmatrix}$ を解け（一般解を求めよ）．

問2. 情報数学

[1] 命題論理に関する以下の問いに答えよ. ただし, $\vee$ は論理和, $\wedge$ は論理積, $\neg$ は否定, $\Rightarrow$ は含意を表わす.

(1) 命題論理における論理式は, 以下のように再帰的に定義される:

i) 原子式は論理式である.

ii) A, B を論理式とすると, $(A \wedge B), (A \vee B), (A \Rightarrow B), (\neg A)$ はそれぞれ論理式である.

iii) 上の i) と ii) によって定められるもののみが論理式である.

P, Q が原子式であるとき, $((\neg(P \wedge Q)) \Rightarrow Q)$ が論理式であることを示せ.

(2) P, Q, R は, 常に真 (1)·偽 (0) いずれかの真理値をもつとする. $(((P \Rightarrow Q) \wedge (Q \Rightarrow R)) \Rightarrow (P \Rightarrow R))$ が命題の内容によらず, 真になる恒真命題であることを, $(P \Rightarrow Q)$, $(Q \Rightarrow R)$, $((P \Rightarrow Q) \wedge (Q \Rightarrow R))$, $(P \Rightarrow R)$, $(((P \Rightarrow Q) \wedge (Q \Rightarrow R)) \Rightarrow (P \Rightarrow R))$ すべての真理値表を作成することで示せ.

(3) 3つの論理結合子 $\neg, \wedge, \vee$ を含む論理式によって, すべての真理関数 (n 個の真理値 $\{1,0\}^n$ を入力とし真理値 $\{1,0\}$ を出力する関数) を表せる (十全である) ことがわかっている. これを利用して, 以下の真理値表で表現される論理結合子 $|$ 1つだけで十全であることを示せ.

P	Q	$P\|Q$
1	1	0
1	0	1
0	1	1
0	0	1

[2] 実数の組の全体 $\mathcal{R}^2$ において, $\mathcal{R}^2$ の 2 点, $A = (a_1, a_2)$, $B = (b_1, b_2)$ に対して関数 $d : \mathcal{R}^2 \times \mathcal{R}^2 \to \mathcal{R}$ を以下のように定義する.

$$d(A, B) = \max\{|a_1 - b_1|, |a_2 - b_2|\}$$

このとき以下の問いに答えよ.

(1) $\mathcal{R}^2$の任意の 3 点, $A = (a_1, a_2)$, $B = (b_1, b_2)$, $C = (c_1, c_2)$ に対して, $d(A, C) \leq d(A, B) + d(B, C)$ が成り立つことを示せ.

(2) 関数 d が距離関数であることを示せ. ただし, 距離関数とは任意の 3 点 A, B, C に対して以下の i)〜iii) を満たす関数である.

i) $d(A,B) \geq 0$ であり, $d(A,B) = 0 \Leftrightarrow A = B$,

ii) $d(A,B) = d(B,A)$,

iii) $d(A,C) \leq d(A,B) + d(B,C)$.

(3) 新たに関数 $\hat{d} : \mathcal{R}^2 \times \mathcal{R}^2 \to \mathcal{R}$ を以下のように定義する.

$$\hat{d}(A,B) = \min\{|a_1 - b_1|, |a_2 - b_2|\}$$

この $\hat{d}$ は距離関数であるか, 距離関数である場合にはそれを示し, 距離関数でない場合にはその理由を述べよ.

問 3. 確率・統計

[1] 離散型の確率変数 X が以下の確率分布にしたがう場合の期待値 $\mathrm{E}[X]$ を求めよ.

(1) 成功確率 p $(0 < p < 1)$, 試行回数 n の二項分布:

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x} \qquad (x = 0, 1, 2, \ldots, n).$$

ただし, $\binom{n}{x} = \dfrac{n!}{x!(n-x)!}$ である.

(2) パラメータ λ $(\lambda > 0)$ のポアソン分布:

$$p(x) = \mathrm{e}^{-\lambda} \frac{\lambda^x}{x!} \qquad (x = 0, 1, 2, \ldots).$$

[2] X と N はそれぞれ離散型の確率変数であり, 0 以上の整数全体 $\{0, 1, 2, \ldots\}$ に値をとるものとする. また, X, N の同時確率関数を $p(x, n) := \mathrm{P}(X = x, N = n)$, X の周辺確率関数を $p_X(x) := \mathrm{P}(X = x)$, N の周辺確率関数を $p_N(n) := \mathrm{P}(N = n)$, $N = n$ という条件が与えられた下での X の条件付き確率関数を $p_{X|N}(x|n) := \mathrm{P}(X = x|N = n)$ と表すことにする.

(1) 以下の式が成り立つことを示せ:

$$p_X(x) = \sum_{n=0}^{\infty} p_N(n) p_{X|N}(x|n).$$

(2) 以下の式が成り立つことを示せ:

$$\mathrm{E}[X] = \mathrm{E}[\mathrm{E}[X|N]].$$

ただし, $\mathrm{E}[X]$ は X の期待値, $\mathrm{E}[\mathrm{E}[X|N]]$ の外側の $\mathrm{E}[\cdot]$ は N に関する期待値である. また, $\mathrm{E}[X|N]$ は N で条件付けられた X の条件付き期待値であり $\mathrm{E}[X|N] := \sum_{x=0}^{\infty} x\, p_{X|N}(x|N)$ と定義される.

(3) 確率変数 N がパラメータ λ $(\lambda > 0)$ のポアソン分布にしたがい, $N = n$ で条件付けられた確率変数 X の条件付き分布が, 成功確率 p $(0 < p < 1)$, 試行回数 n の二項分布であるとする. つまり, N の周辺確率関数と, N が与えられた下での X の条件付き確率関数が以下のように与えられるとする:

$$p_N(n) = \mathrm{e}^{-\lambda} \frac{\lambda^n}{n!} \qquad (n = 0, 1, 2, \ldots),$$

$$p_{X|N}(x|n) = \binom{n}{x} p^x (1-p)^{n-x} \qquad (x = 0, 1, 2, \ldots, n).$$

このとき, X の期待値 $\mathrm{E}[X]$ を求めよ.

問 4. コンピュータ基礎工学

［1］ コンピュータの数値表現に関する以下の問いに答えよ.

（1） 10 進数の 100 を 2 進数で表現せよ.
（2） 10 進数の 111 を 8 ビット，2 の補数表現で表せ.
（3） 10 進数の −86 を 8 ビット，2 の補数表現で表せ.
（4） 10 進数の計算 111−86 を 8 ビット，2 の補数表現で計算した場合の計算過程と結果を示せ.

［2］ 浮動小数点形式の数値の演算に関する次の記述中の括弧 a〜d に入れるべき適切な数値および用語を答えよ.

浮動小数点形式で表現された数値の演算では，浮動小数点形式の表現規則により，「桁落ち」と「情報落ち」と呼ばれる誤差が発生する．たとえば，仮数部の有効桁数が 3 桁，指数部の有効桁数が 2 桁の 10 進浮動小数点数で，$0.123\times10^{2}+0.456\times10^{-2}$ という演算を行う場合，実際には，$0.123\times10^{2}+$（ a ）という演算が行われることになる．このような場合に発生する誤差を（ b ）と呼ぶ．一方，$0.789\times10^{2}-0.788\times10^{2}$ という演算では，演算結果が正規化され（ c ）となり，有効桁数が減少する．このような有効桁数の減少を（ d ）と呼ぶ．なお，ここでは仮数部の有効桁の先頭が小数第 1 位に来るように正規化されている．

［3］ パイプライン制御に関する以下の問いに答えよ.

パイプライン制御とは，CPU の内部動作をいくつかのステージに分割して，ステージ単位で並行処理を行う仕組みである．ここでは，1 つの命令を以下の 4 つのステージに分割する.

- 命令取り出し（Instruction Fetch）
- 命令解読（Instruction Decode）
- 命令実行（Execute）
- 結果書き込み（Write-back）

(1) 命令を分割するステージ数を 4，ステージ毎の実行時間を 10ms，実行する命令数を 5 とするとき，すべての処理を終了するまでの時間を計算せよ.
(2) 命令を分割するステージ数を D，ステージ毎の実行時間を P 秒とするとき，I 個の命令をパイプラインで実行するのに要する時間を D，P，I と数値を用いて表せ．なお，パイプラインの各ステージは 1 ピッチで処理されるものとし，パイプラインハザードについては考慮しなくてよい.

［4］仮想記憶方式に関する次の記述を読んで設問に答えよ．

仮想記憶方式とは，オペレーティングシステム（OS）の主記憶管理において，OS が提供する論理的な記憶領域（仮想記憶）のアドレスと主記憶上の物理的なアドレスを対応付けて管理する方式である．仮想記憶方式では，補助記憶装置を仮想記憶として用いるので，仮想記憶上に主記憶の容量を超えるプログラムを格納することができる．仮想記憶上のアドレス空間を仮想アドレス空間，主記憶上のアドレス空間を物理アドレス空間と呼び，それぞれの空間の記憶場所を仮想アドレスと物理アドレスで指定する．

図 1 に，仮想記憶方式の一つであるページング方式における仮想アドレス空間のページと物理アドレス空間のページの対応例を示す．**図 1** では，補助記憶装置に格納されているプログラム A は，a1，a2，a3，a4，a5 に分割されている．仮想ページと物理ページの対応は，ページテーブルで管理されている．ページテーブルでは，仮想ページの内容が物理アドレス空間にも存在しているかどうかを示すビット（存在ビット）が管理されており，ページが存在するときは 1，存在していないときは 0 とする．

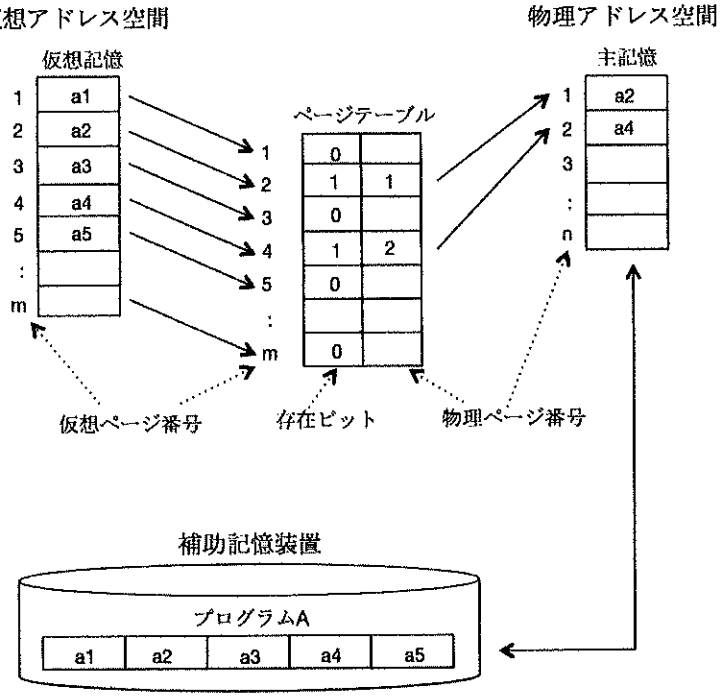

図 1 仮想アドレス空間のページと物理アドレス空間のページの対応例

(1) **図 1** の状態から，主記憶の物理ページ番号 3 に a3，物理ページ番号 4 に a 1 が読み込まれ，さらに物理ページ番号 2 が a5 に置き換えられたとする．このときのページテーブルの値を，**図 1** にならって表記せよ．

(2) プログラム実行過程で，実行に必要なページが主記憶にないとき，ページフォールトが発生し，ページアウトやページインという割込みが発生する．

いま物理ページの個数が 3 しか使えないと仮定し，以下の順で仮想ページを参照するとき，ページ置換えアルゴリズムが FIFO (First-In First-Out)の場合と LRU (Least Recently Used)の場合で発生するページフォールトの回数を比較せよ．

[仮想ページの参照順（仮想ページ番号の並び）] 2→4→3→1→3→4→5→4→3→1→4

問 5. 計算機プログラミング

［1］ 次のような確率に従う事象のシミュレーションを行いたい.

- 50%の確率で空が曇る.
- 空が曇っているとき，10%の確率でスプリンクラーが稼動する.
- 空が曇っていないとき，50%の確率でスプリンクラーが稼動する.
- 空が曇っているとき，80%の確率で雨が降る.
- 空が曇っていないとき，20%の確率で雨が降る.
- スプリンクラーが稼動していて，かつ雨が降っているとき，99%の確率で芝生が濡れる.
- スプリンクラーが稼動していて，かつ雨が降っていないとき，90%の確率で芝生が濡れる.
- スプリンクラーが稼動していなくて，かつ雨が降っているとき，90%の確率で芝生が濡れる.
- スプリンクラーが稼動していなくて，かつ雨が降っていないとき，芝生が濡れる確率は 0% である.

ここでは，空が曇っているかどうかが与えられたとき，スプリンクラーが稼動することと，雨が降ることは条件付独立であると考えている．ソースコード 1 は，シミュレーションを行ってサンプルを生成し，芝生が濡れているときに，空が曇っている確率を計算する C 言語プログラムである．ここで，「空が曇っている」「スプリンクラーが稼動している」「雨が降っている」「芝生が濡れている」といった事象が成立しているときを整数 1，そうでないときを 0 で表している．ただし，エラー処理については省略している場合がある．なお，srand は乱数の種を設定する関数，rand は 0 から RAND_MAX までの範囲で一様分布する整数の乱数を返す関数，time は現在時刻をある定まった時刻からの経過秒数として返す関数である．以下の問いに答えよ

（1） 関数 generate_samples は指定された数のサンプルを生成する．(ア)，(イ) を適切に埋めて完成させよ.

（2） (ウ)，(エ)，(オ) を適切に埋めて main 関数を完成させよ.

ソースコード 1

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define NSAMPLES 5000

const double pcloudy = 0.5,
             psprinkler_cloudy1 = 0.1, psprinkler_cloudy0 = 0.5,
             prain_cloudy1 = 0.8, prain_cloudy0 = 0.2,
             pwet_sprinkler1_rain1 = 0.99, pwet_sprinkler1_rain0 = 0.9,
             pwet_sprinkler0_rain1 = 0.9, pwet_sprinkler0_rain0 = 0.0;

struct sample {
    int cloudy;
    int sprinkler;
    int rain;
    int wet;
}
```

```
double rand1(void) {
    return rand()/(RAND_MAX+1.0);
}

void generate_samples (struct sample s[], int n) {
    int j;
    srand((unsigned int) time(NULL));
    for (j = 0; j < n; j++){
        if (rand1() < pcloudy){
            [ (ア) ];
            if (rand1() < psprinkler_cloudy1) s[j].sprinkler=1;
            if (rand1() < prain_cloudy1) s[j].rain=1;
        } else {
            if (rand1() < psprinkler_cloudy0) s[j].sprinkler=1;
            if (rand1() < prain_cloudy0) s[j].rain=1;
        }
        if ((s[j].sprinkler==1) && (s[j].rain==1)){
            if (rand1() < pwet_sprinkler1_rain1) s[j].wet=1;
        } else if ((s[j].sprinkler==1) && (s[j].rain==0)){
            if (rand1() < pwet_sprinkler1_rain0) s[j].wet=1;
        } else if ([ (イ) ]){
            if (rand1() < pwet_sprinkler0_rain1) s[j].wet=1;
        } else {
            if (rand1() < pwet_sprinkler0_rain0) s[j].wet=1;
        }
    }
}

int main(void){
    int i, ncloudy=0, nwet=0;
    [ (ウ) ];
    for (i = 0; i < NSAMPLES; i++) {
        samples[i].cloudy=0;
        samples[i].sprinkler=0;
        samples[i].rain=0;
        samples[i].wet=0;
    }
    generate_samples ([ (エ) ]);
    for (i = 0; i < NSAMPLES; i++){
        if (samples[i].wet==1){
            nwet++;
            if ([ (オ) ]) ncloudy++;
        }
```

```
    }
    printf("%f¥n",(double)ncloudy/nwet);
}
```

［2］ ソースコード2は，標準入力から文字列を読み込み，長さ SUBLEN の部分文字列を列挙し，その頻度とともに標準出力に表示するC言語プログラムである．ただし，エラー処理については省略している場合がある．部分文字列はその頻度とともに二分木に格納される．なお，strlen(s)は文字列 s の長さを返す関数，strcpy(s,t)は文字列 t を s へコピーする関数，strcmp(s,t)は文字列 s と t を辞書順で比較し，s が t より小さければ負の値を，s と t が等しければ0を，s が t より大きければ正の値を返す関数である．malloc は指定されたサイズの新しい記憶領域へのポインタを返す関数である．以下の問いに答えよ．

（1） 関数 add_tree は部分文字列がすでに二分木に格納されていれば，頻度を 1 増やし，そうでなければ新しいノードを作成し格納する．（カ）と（キ）を適切に埋めて完成させよ．

（2） 関数 tree_print は二分木に格納された部分文字列を辞書順にその頻度とともに表示する．（ク）を適切に埋めて完成させよ．

（3） （ケ）と（コ）を適切に埋めて main 関数を完成させよ．

ソースコード2

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAXLEN 100
#define SUBLEN 3

struct t_node {
    char *substring;
    int count;
    struct t_node *left;
    struct t_node *right;
};

struct t_node *t_alloc(void){
    return (struct t_node *) malloc(sizeof(struct t_node));
}

struct t_node *add_tree(struct t_node *p, char *w){
    int cond;
    if (      (カ)      ){
        p=t_alloc();
        p->substring = (char *) malloc(SUBLEN+1);
        strcpy(p->substring,w);
        p->count=1;
        p->left=p->right=NULL;
```

```
    }else if ((cond=strcmp(w,p->substring))==0)
        p->count++;
    else if (cond < 0)
        p->left=add_tree(p->left,w);
    else
        [          (キ)          ];
    return p;
}

void tree_print(struct t_node *p){
    if (p != NULL) {
        tree_print(p->left);
        printf("%s %d¥n", p->substring, p->count);
        [        (ク)        ];
    }
}

int main(void){
    int i,j;
    char string[MAXLEN];
    int length;
    char sub[SUBLEN+1];
    struct t_node *root;

    root=NULL;
    scanf("%s",string);
    length=strlen(string);
    if (SUBLEN>length){
        return 0;
    }else{
        for (i=0; i < length-SUBLEN+1; i++){
            for (j=0; j<SUBLEN; j++){
                [        (ケ)        ];
            }
            sub[SUBLEN]='¥0';
            root=add_tree([        (コ)        ]);
        }
    }
    tree_print(root);
    return 0;
}
```