筆答専門試験科目（午前）
情報工学系

２０２４　大修

時間　９：３０～１２：００

注　意　事　項

１．各答案用紙の受験番号欄に受験番号を記入し，試験科目名欄に「情報工学系」と記入せよ．
２．次の１番～３番の３題すべてに解答すること．
３．解答は１題ごとに別々の答案用紙に，問題番号を明記した上で記入せよ．必要であれば，答案用紙の裏面に記入して良いが，答案用紙の表面にその旨を明記すること．
４．１枚の答案用紙に２題以上の解答を記入した場合はそれらの解答を無効とすることがある．
５．１題の解答を２枚以上の答案用紙に記入した場合はその解答を無効とすることがある．
６．電子式卓上計算機等の使用は認めない．

1. 以下の問いに答えよ．

1) 次の実行列 $\mathbf{P}$ の行列式 $|\mathbf{P}|$ を計算せよ．ただし，$c \neq 0$ かつ $d \neq 24$ とする．

$$\mathbf{P} = \begin{pmatrix} 2 & 0 & 0 \\ 13 & 5 & a \\ 25 & 8 & b \end{pmatrix} \begin{pmatrix} c & 3 & 4 \\ 0 & 9 & 27 \\ 0 & 24 & 3d \end{pmatrix}^{-1}$$

2) 2×2 実対称行列 $\mathbf{Q}$ および 2 次元実ベクトル $\mathbf{c}$ を用いて定義される実関数

$$h(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{c}^\top \mathbf{x} + 3$$

について，次の問いに答えよ．ただし，$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2$ であり，$\mathbb{R}$ は実数全体の集合，$\top$ は転置を表す．

a) $\mathbf{Q} = \begin{pmatrix} 2 & -1 \\ -1 & 4 \end{pmatrix}$，$\mathbf{c} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ のとき，h の偏導関数 $\dfrac{\partial h}{\partial x_1}$ および $\dfrac{\partial h}{\partial x_2}$ をそれぞれ求めよ．

b) $\mathbf{Q} = \begin{pmatrix} 2 & -1 \\ -1 & 4 \end{pmatrix}$，$\mathbf{c} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ のとき，$\dfrac{\partial h}{\partial x_1} = \dfrac{\partial h}{\partial x_2} = 0$ となる全ての (x_1, x_2) の組み合わせを求めよ．

c)「$\dfrac{\partial h}{\partial x_1} = \dfrac{\partial h}{\partial x_2} = 0$ となる (x_1, x_2) が存在しかつ一意に定まる」ための $\mathbf{Q}$ に関する必要十分条件を答えよ．

d)「$\dfrac{\partial h}{\partial x_1} = \dfrac{\partial h}{\partial x_2} = 0$ となる (x_1, x_2) が存在しかつ一意に定まり，さらにそのような (x_1, x_2) において h が最小値をとる」ための $\mathbf{Q}$ に関する必要十分条件を答えよ．

3) 確率変数 X の確率分布が区間 $[-1, 1]$ 上の一様分布であるとする．次の問いに答えよ．

a) 確率変数 X の確率密度関数 f_X を求めよ．

b) 確率変数 $Y = \dfrac{1}{1 + e^{-X}}$ の確率密度関数 f_Y を求めよ．ただし，e はネイピア数を表す．

2. 以下の問いに答えよ．

1) 文脈自由文法 $G = (N, \Sigma_1, P, E)$ について考える．ここで，$N = \{E, T, F\}$ は非終端記号の集合，$\Sigma_1 = \{0, 1, \times, +, (,)\}$ は終端記号の集合，P は以下に示す生成規則の集合，E は開始記号(出発記号ともいう)となる非終端記号を表す．

$$P = \{E \to T,\ E \to E + T,\ T \to F,\ T \to T \times F,\ F \to (E),\ F \to 0,\ F \to 1\}$$

a) 図 2.1 が，文法 G による正しい導出木であるか否かを答えよ．正しい導出木であるならば，導出している文字列を書け．そうでないならば，その誤りを説明せよ．

b) 図 2.2 が，文法 G による正しい導出木であるか否かを答えよ．正しい導出木であるならば，導出している文字列を書け．そうでないならば，その誤りを説明せよ．

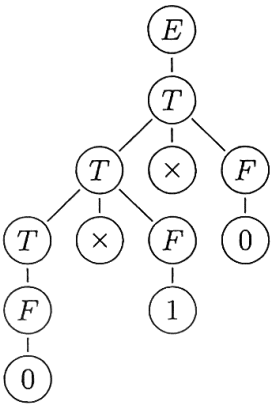

図 2.1

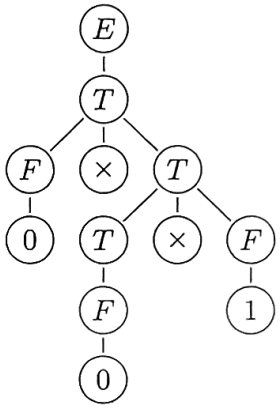

図 2.2

c) 文字列 $(1 + 0) \times 1$ は，文法 G により導出されるだろうか．導出されるのであれば，導出木を書け．導出されないのであれば，その理由を書け．

(次ページにつづく)

(前ページのつづき)

2) 以下に述べる非決定性有限オートマトンと正規文法について，空欄 (ア) から (キ) を埋めよ．ただし，空欄 (ア)〜(オ) には a もしくは b のいずれかがあてはまり，空欄 (カ)，(キ) には U, A, B のいずれかがあてはまる．

次に示す非決定性有限オートマトン $(Q, \Sigma_2, \delta, q_1, F')$ を考える．ここで，$Q = \{q_1, q_2, q_3, q_4\}$ は状態の集合，$\Sigma_2 = \{a, b\}$ はアルファベット，δ は図 2.3 の遷移図により与えられる遷移関数である．また，q_1 は初期状態であり，$F' = \{q_4\}$ は受理状態 (最終状態ともいう) の集合である．

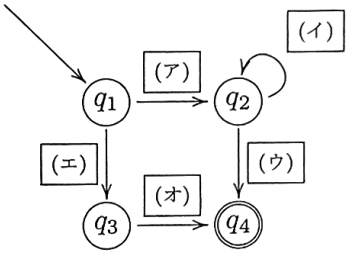

図 2.3

この非決定性有限オートマトンが受理する文字列の集合と，次の正規文法 G' で生成される文字列の集合は一致する．ここで，$G' = (N', \Sigma_2', P', U)$ であり，$N' = \{U, A, B\}$ は非終端記号の集合，$\Sigma_2' = \{a, b\}$ は終端記号の集合，P' は以下に示す生成規則の集合，U は開始記号 (出発記号ともいう) となる非終端記号を表す．

$$P' = \{U \to aA,\ U \to b\boxed{(カ)},\ A \to a\boxed{(キ)},\ \boxed{(キ)} \to a,\ \boxed{(カ)} \to b\}$$

(次ページにつづく)

(前ページのつづき)

3) 命題論理の自然演繹について考える．p_0, p_1, p_2, p_3 を命題記号 (命題変数ともいう) とする．

a) 命題の集合 $\Gamma_1 = \{p_0,\ p_0 \to p_1,\ p_1 \to p_2,\ p_1 \wedge p_2 \to p_3,\ p_3 \to \neg p_0\}$ は矛盾している．すなわち，Γ_1 の要素を仮定とし，$\bot$ を結論とする，次のような導出 (証明木，証明図ともいう) が存在する．空欄 (ク)〜(ス) にあてはまる命題を答えよ．

$$\dfrac{\dfrac{\dfrac{p_0 \quad p_0 \to p_1}{\boxed{\text{(ケ)}}}\to E \quad \dfrac{p_0 \quad p_0 \to p_1}{\boxed{\text{(ク)}}}\to E}{\boxed{\text{(サ)}}}\wedge I \quad \dfrac{p_1 \to p_2}{\boxed{\text{(コ)}}}\to E}{\dfrac{\dfrac{p_1 \wedge p_2 \to p_3}{\boxed{\text{(シ)}}}\to E \quad p_3 \to \neg p_0}{\boxed{\text{(ス)}}}\to E \quad p_0}\neg E$$
$$\bot$$

ただし，$\to E$ は含意の消去規則 (除去規則ともいう)，$\wedge I$ は連言の導入規則，$\neg E$ は否定の消去規則 (除去規則ともいう) である．

b) 命題の集合 $\Gamma_2 = \{p_0 \to p_1,\ p_0 \to p_2,\ p_1 \wedge p_2 \to p_3,\ p_3 \to \neg p_0\}$ は矛盾しているだろうか．矛盾しているならば，Γ_2 の要素を仮定とし，$\bot$ を結論とするような導出を書け．矛盾していない，すなわち，無矛盾であるならば，その理由を示せ．

4) 一階述語論理について考える．$\prec$ を 2 引数の述語記号とし，論理式の集合 Δ を以下のように定める．

$$\Delta = \{\forall x \forall y \forall z(x \prec y \wedge y \prec z \to x \prec z),\ \forall x \forall y \exists z(x \prec y \to x \prec z \wedge z \prec y)\}$$

a) 整数全体の集合 $\mathbb{Z}$ をユニバースとし，述語記号 $\prec$ を整数の大小関係 $<$ と解釈する．その解釈は Δ のすべての要素を真とするか否かを答え，その理由を示せ．

b) 有理数全体の集合 $\mathbb{Q}$ をユニバースとし，述語記号 $\prec$ を有理数の大小関係 $<$ と解釈する．その解釈は Δ のすべての要素を真とするか否かを答え，その理由を示せ．

c) 集合 $\{rock, paper, scissors\}$ をユニバースとし，述語記号 $\prec$ を関係

$$\{(rock, paper), (paper, scissors), (scissors, rock)\}$$

と解釈する．その解釈は Δ のすべての要素を真とするか否かを答え，その理由を示せ．

3.

1）アルゴリズムの計算量に関して，次の問いに答えよ．

a）次の自然数 n の関数について n を十分大きくしたときの n に対する増加量が最も小さいものから大きいものの順に並べよ．

① $n/(\log n)$，② $\log n$，③ $(\log n)^3$，④ 2^n，⑤ $n^{1/2}$，⑥ $n!$

b）自然数 n の関数 f(n)に対するビッグオー記法 $f(n)=O(g(n))$を考える．ここで，g(n)も自然数 n の関数である．f(n)として以下に示す関数（ア），（イ），（ウ）を考えたとき，それぞれに対する g(n)のうち「n を十分大きくしたときの n に対する増加量が最も小さい」ものを図 3.1 の選択肢群から選び番号で答えよ．

（ア） $2n^3+3(\log n)^2+n^{1.6}\log n$

（イ） $n^{20}+n^{\log n}+\log(n!)$

（ウ） $n^2(\sin^2 n)\,2^n$

c）表 3.1 の整列アルゴリズムの長さ n の配列に対する最悪時間計算量と平均時間計算量をビッグオー記法を用いて表す場合，空欄 [あ]～[く] に当てはまる「n を十分大きくしたときの n に対する増加量が最も小さい」関数を図 3.1 の選択肢群からそれぞれ選び番号で答えよ．同じ選択肢を複数回選んでもよい．

① n^3	② $(\log n)^2$	③ $n^{1.6}\log n$	④ $n^{1.6}$
⑤ n^{20}	⑥ $n^{\log n}$	⑦ $n \log n$	⑧ n^2
⑨ $\sin^2 n$	⑩ 2^n	⑪ $n^2(\sin^2 n)$	⑫ $n^2\,2^n$

図 3.1

表 3.1

	最悪時間計算量	平均時間計算量
クイックソート	O([あ])	O([い])
マージソート	O([う])	O([え])
バブルソート	O([お])	O([か])
ヒープソート	O([き])	O([く])

（次ページにつづく）

（前ページのつづき）

2）二分探索木を構築するための C 言語のプログラムを考える．最初に，プログラム 3.1 に示す構造体 node を定義する．図 3.2 にこれに対応する木構造の様子を示す．プログラム 3.2 に構造体 node を用いて二分探索木を生成し，最後に得られた二分探索木の高さを出力する main 関数を示す．この main 関数では配列 data を定義し，空欄 ア の中には 7 個の整数が入るものとする．配列 data の要素はループの中で二分探索木に挿入される．ただし，二分探索木の高さを計算する関数 height をプログラム 3.3，二分探索木の節点を生成する関数 create をプログラム 3.4，二分探索木に節点を挿入する関数 insert をプログラム 3.5 に示す．

a）プログラム 3.4 と 3.5 の空欄 イ ～ キ に当てはまる最も適切なものを図 3.3 の選択肢群からそれぞれ選び番号で答え，プログラムを完成させよ．同じ選択肢を複数回選んでもよい．

b）プログラム 3.2 の空欄 ア の中に 7 個の整数 0, 2, 3, 4, 6, 1, 5 を入れた場合，printf 関数の呼び出しにより出力される二分探索木の高さを答えよ．

c）プログラム 3.2 の空欄 ア の中に 7 個の整数 4, 2, 2, 2, 0, 4, 8 を入れた場合，printf 関数の呼び出しにより出力される二分探索木の高さを答えよ．

```
struct node {
  int data;
  struct node* left;
  struct node* right;
};
```

プログラム 3.1

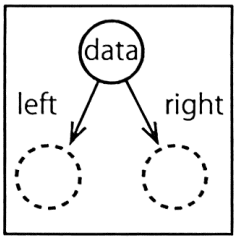

図 3.2

① data	② left	③ right	④ NULL
⑤ new_node	⑥ *new_node	⑦ **new_node	⑧ p->data
⑨ p->left	⑩ p->right	⑪ left->data	⑫ right->data

図 3.3

（次ページにつづく）

（前ページのつづき）

```
int main(void) {
  int data[7] = {      ア      };
  struct node* root = NULL;
  for (int i = 0; i < 7; i++) {
    root = insert(root, data[i]);
  }
  printf("%d\n", height(root));
}
```

プログラム 3.2

```
int height(struct node* p) {
  if (p == NULL) return -1;
  int left = 1 + height(p->left);
  int right = 1 + height(p->right);
  if (left > right) return left;
  else return right;
}
```

プログラム 3.3

```
struct node* create(int data) {
  struct node* new_node = (struct node*) malloc(sizeof(struct node));
  new_node->  イ  =  ウ  ;
  new_node->left = NULL;
  new_node->right = NULL;
  return  エ  ;
}
```

プログラム 3.4

```
struct node* insert(struct node* p, int data) {
  if (p ==  オ  ) p = create(data);
  else if (data > p->data)
     カ  = insert(p->right, data);
  else
     キ  = insert(p->left, data);
  return p;
}
```

プログラム 3.5

（次ページにつづく）

（前ページのつづき）

節点数 n の二分探索木の高さが最大の場合でも O(log n)となるようにすると挿入・削除の最悪時間を O(log n)に抑えることができる．このような二分探索木を平衡探索木と呼ぶ．特に，図 3.4 に示す回転を用いて「どの節点においても，その左右の部分木の高さの差が 1 以下」になるように生成された平衡探索木を AVL 木と呼ぶ．

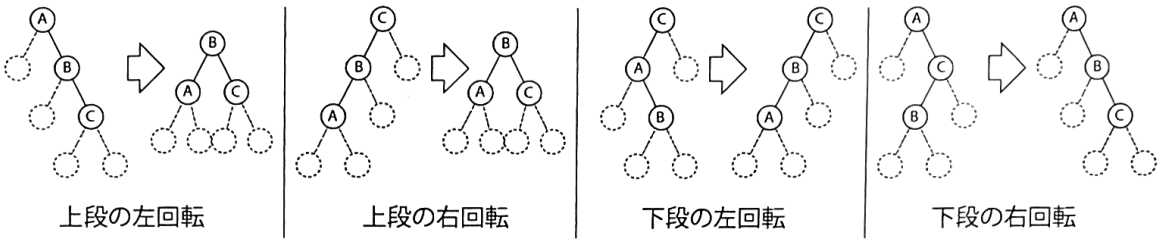

図 3.4

AVL 木への要素の挿入を行う際には，左右の部分木の高さの差が挿入後も 1 以下であることを保証する必要がある．具体的には，挿入により高さの差が 2 になる場合には要素の大小関係に従い図 3.4 中の 1 つまたは 2 つの適切な回転を行うことで AVL 木にすることができる．AVL 木を生成する main 関数をプログラム 3.6 に示す．空欄 [ク] の中には 7 個の整数が入るものとする．AVL 木への要素の挿入を行う関数 insert_avl をプログラム 3.7 に示す．ただし，関数 rotate_left は上段の左回転及び下段の左回転を行うものであり，その定義をプログラム 3.8 に示す．また，関数 rotate_right はプログラム 3.8 で定義した関数 rotate_left と逆向きの回転を行う．関数 height はプログラム 3.3 で定義したものとする．

d）プログラム 3.7 の空欄 [ケ] ～ [セ] に当てはまる最も適切なものを図 3.3 の選択肢群からそれぞれ選び番号で答え，プログラムを完成させよ．同じ選択肢を複数回選んでもよい．

e）プログラム 3.8 の空欄 [ソ] ～ [ツ] に当てはまる最も適切なものを図 3.3 の選択肢群からそれぞれ選び番号で答え，プログラムを完成させよ．同じ選択肢を複数回選んでもよい．

（次ページにつづく）

（前ページのつづき）

```
int main(void) {
  int data[7] = {      ク      };
  struct node* root = NULL;
  for (int i = 0; i < 7; i++) {
    root = insert_avl(root, data[i]);
  }
  printf("%d\n", height(root));
}
```

プログラム 3.6

```
struct node* insert_avl(struct node* p, int data) {
  if (p == NULL) p = create(data);
  else if (data > p->data) {
    p->right = insert_avl(p->right, data);
    if (height(p->right) - height(p->left) == 2) {
      if (data > p->    ケ    )
        p = rotate_left(p);
      else {
        コ    = rotate_right(    サ    );
        p = rotate_left(p);
      }
    }
  }
  else {
    p->left = insert_avl(p->left, data);
    if (height(p->left) - height(p->right) == 2) {
      if (data > p->    シ    ) {
        ス    = rotate_left(    セ    );
        p = rotate_right(p);
      }
      else
        p = rotate_right(p);
    }
  }
  return p;
}
```

プログラム 3.7

（次ページにつづく）

（前ページのつづき）

```
struct node* rotate_left(struct node* p) {
  struct node* right_child = p->[ソ];
  p->[タ] = right_child->[チ];
  right_child->[ツ] = p;
  return right_child;
}
```

プログラム 3.8

f）プログラム 3.6 の空欄 ク の中に 7 個の整数 0, 2, 3, 4, 6, 1, 5 を入れた場合，printf 関数の呼び出しにより出力される AVL 木の高さを答えよ．

g）プログラム 3.6 の空欄 ク の中に 7 個の整数 4, 2, 2, 2, 0, 4, 8 を入れた場合，printf 関数の呼び出しにより出力される AVL 木の高さを答えよ．

h）プログラム 3.6 の空欄 ク の中に 7 個の整数 2, 5, 3, 0, 1, 6, 4 を入れた場合，printf 関数の呼び出しにより出力される AVL 木の高さを答えよ．