平成３０年度

筑波大学大学院博士課程
システム情報工学研究科
コンピュータサイエンス専攻

博士前期課程（一般入学試験　２月期）
試験問題　基礎科目（数学，情報基礎）
Mathematics／Fundamentals of Computer Science

[注意事項][Instructions]

1. 試験開始の合図があるまで，問題の中を見てはいけません．また，筆記用具を手に持ってはいけません．
   Do NOT open this booklet before the examination starts. In addition, do not have a pen in hand before the examination starts.
2. 解答用紙（罫線有り）を３枚，下書き用紙（白紙）を１枚配布します．
   You are given three answer sheets (ruled paper) and one draft sheet (blank paper).
3. 試験開始の合図のあとで，全ての解答用紙の定められた欄に，研究科，専攻，受験番号を記入すること．
   Fill in the designated spaces on each answer sheet with the name of the graduate school, the name of main field (department), and your examination number after the examination starts.
4. この問題は全部で 14 ページ（表紙を除く）です．1〜7 ページは日本語版，8〜14 ページは英語版です．
   This booklet consists of 14 pages, excluding the cover sheet. The Japanese version is shown on pages 1–7 and the English version on pages 8–14.
5. 問題は全部で５問あります．このうち，３問選択すること．問題ごとに解答用紙を分けて記入すること．
   There are five problems. Select three problems. Write your answer to each problem in a different answer sheet.
6. 解答用紙に解答を記述する際に，問題番号を必ず明記すること．
   When writing the answers, clearly label the problem number on each answer sheet.

平成３０年２月１日

問題 I の解答に使用する解答用紙の先頭には「問題 I」と明記すること．この解答用紙には問題 I に対する解答以外を記述しないこと．

問題 I　y_1，y_2 が x に関して微分可能な関数であるとき，連立方程式

$$\begin{aligned} y_1' &= y_1 + 2y_2 \\ y_2' &= 2y_1 - 2y_2 \end{aligned}$$

について，以下の問いに答えなさい．

(1) 連立方程式が $\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = A \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ と表されるとき，行列 A を求めなさい．

(2) $D = P^{-1}AP$ を満たす正則行列 P と対角行列 D を一組求めなさい．

(3) w_1，w_2 が x に関して微分可能な関数であり，$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = P^{-1} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ であるとき，$\begin{pmatrix} w_1' \\ w_2' \end{pmatrix} = D \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ であることを示しなさい．

(4) 設問 (3) の結果を用いて，y_1，y_2 を求めなさい．ただし，$(e^{ax})' = ae^{ax}$（a は定数）が成り立つことを使ってよい．

問題 II の解答に使用する解答用紙の先頭には「問題 II」と明記すること．この解答用紙には問題 II に対する解答以外を記述しないこと．

問題 II

(1) $x^2 + xy + y^2 = 1$ とする．以下の問いに答えなさい．

(a) $\dfrac{dy}{dx}$ を求めなさい．

(b) $\dfrac{d^2y}{dx^2}$ を求めなさい．

(2) 次の定積分の値を求めなさい．

$$\int_{-\infty}^{\infty} \frac{dx}{4x^2 + 6x + 3}$$

問題 III の解答に使用する解答用紙の先頭には「問題 III」と明記すること．この解答用紙には問題 III に対する解答以外を記述しないこと．

問題 III　$\mathbb{N}$ を（0 を含む）全ての自然数からなる集合とし，$S = \{x \in \mathbb{N} \mid 0 \leq x \leq 4\}$ とする．S の直積 $S \times S$ 上の関数 f と g_n（ただし $n \in \mathbb{N}$）を，それぞれ以下のように定義する．

$$\begin{aligned} f(\langle x, y\rangle) &= \langle (x+1) \bmod 5,\ (y+2) \bmod 5\rangle \\ g_n(\langle x, y\rangle) &= \begin{cases} \langle x, y\rangle & (n = 0 \text{ のとき}) \\ f(g_{n-1}(\langle x, y\rangle)) & (n \geq 1 \text{ のとき}) \end{cases} \end{aligned}$$

ここで，0 以上の整数 a と正の整数 b に対し，$a \bmod b$ は a を b で割った余りを表す．また，$S \times S$ 上の二項関係 R_m（ただし $m \in S$）を以下のように定義する．

$$\langle x, y\rangle R_m \langle x', y'\rangle \iff \exists k \in S\ (k \leq m \wedge \langle x', y'\rangle = g_k(\langle x, y\rangle))$$

このとき，以下の問いに答えなさい．

(1) 集合 $T = \{\langle x, y\rangle \in S \times S \mid x = y\}$ に対して，関数 f による T の像 $f(T)$ を求めなさい．

(2) $g_3(\langle 1, 2\rangle)$ を求めなさい．計算過程を示すこと．

(3) 関数 g_2 の逆関数を求めなさい．

(4) 反射律・対称律・推移律を全て満たす二項関係を，一般に同値関係という．R_4 が同値関係となることを示しなさい．

(5) R_3 は $S \times S$ 上で同値関係ではない．そのことを反例を 1 つ挙げて示しなさい．

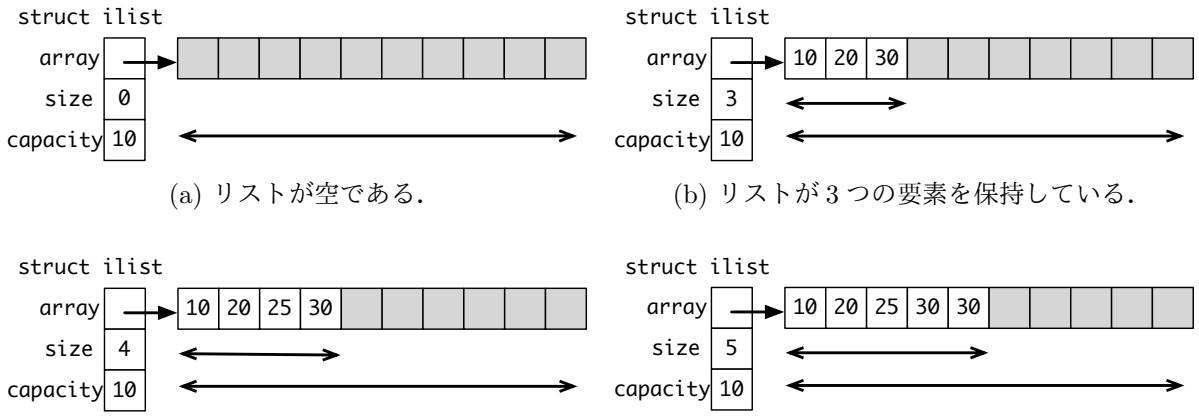
問題 IV の解答に使用する解答用紙の先頭には「問題 IV」と明記すること．この解答用紙には問題 IV に対する解答以外を記述しないこと．

問題 IV

小さい順 (昇順，ascending order) に並んだ整数を保持するリストを，C 言語の配列で実装する．図 1 に，そのための構造体 `ilist` を示す．メンバ `array` は，配列の先頭，メンバ `size` は，リストの要素数，メンバ `capacity` は，配列の容量を保持する．図 2 に，構造体 `ilist` の例を示す．構造体 `ilist` は，表 1 に示した関数で操作される．

```
struct ilist {
  int *array;
  int  size;
  int  capacity;
};
```

図 1: 整数のリストを保持する構造体 `ilist` の定義．



(a) リストが空である．

(b) リストが 3 つの要素を保持している．

(c) リストが 4 つの要素を保持している．

(d) リストが 5 つの要素を保持している．

図 2: 整数のリストを保持する構造体 `ilist` の例．

表 1: 構造体 `ilist` を操作する関数．

関数	説明
`struct ilist *new_ilist()`	構造体と配列のためにメモリを割り当て，構造体のメンバを初期化し，構造体へのポインタを返す．
`void free_ilist(struct ilist *l)`	構造体，および，その配列のメモリを解放する．
`void expand(struct ilist *l)`	構造体の配列を拡張し，容量を増やす．
`void insert(struct ilist *l, int x)`	構造体の，要素が小さい順に並んだ配列に整数を挿入する．挿入後も配列は小さい順に並んでいる．
`void append(struct ilist *l, int x)`	構造体の配列の末尾に整数を追加する．この関数は配列の容量が不足していた場合，関数 `expand()` を呼び容量を増やす．

次の問いに答えなさい．

(1) 次の関数 make_list() は，空のリストを割り当て，そのリストに 3 つの整数を加え，図 2(b) のようなリストを返す．表 1 に示した関数を用いて空欄を埋めて，関数を完成させなさい．

```
struct ilist *make_list() {
  struct ilist *l = new_ilist();
  (A) ( l, (B) );
  (C) ( l, 10 );
  append( l, (D) );
  return( l );
}
```

(2) 次の関数 uniq() は，引数として与えられたリストを読み，連続して現れる要素を比較し，重複を排除したリスト (ユニークな要素を含むリスト) を返す．たとえば，この関数は，図 2(d) のようなリストを取ると，図 2(c) のようなリストを返す．空欄を埋めて，関数を完成させなさい．

```
struct ilist *uniq( struct ilist *l ) {
  struct ilist *u;
  int x, i ;
  u = new_ilist();
  if( (E) )
    return( u );
  x = (F) ;
  append( u, x );
  for( (G) ) {
    if( (H) ) {
      x = (I) ;
      append( u, x );
    }
  }
  return( u );
}
```

(3) 次の関数 merge() は，2 つのリストを引数として取り，それらを 1 つのリストへマージする．ただしこの関数は，引数の各リストが小さい順に並んでいることを前提としている．空欄を埋めて，関数を完成させなさい．

```
struct ilist *merge( struct ilist *a,
                     struct ilist *b ) {
  struct ilist *l;
  int i, j;
  l = new_ilist();
  i=0; j=0;
  while( (J) ) {
    if( (K) ) {
      append( l, (L) );
      i++;
    }
    else {
      append( l, (M) );
      j++;
    }
  }
```

```
  if( (N) ) {
    for( ; i<a->size; i++ ) {
      append( l, (O) );
    }
  }
  else {
    for( ; j<b->size; j++ ) {
      append( l, (P) );
    }
  }
  return( l );
}
```

右上に続く。

(4) 次の関数 insert() は，関数 expand() と find_pos() を利用し，整数 x をリスト l に挿入する．関数 insert() は，そのリストが小さい順に並んでいることを前提としている．関数 expand() は必ず成功するものとする．関数 find_pos() は，二分探索法を用いている．空欄を埋めて，関数を完成させなさい．

```
void insert( struct ilist *l, int x ) {
  int i, j;
  if( l->size == l->capacity )
    expand( l );
  i = find_pos( l, x );
  for( [     (Q)     ] ) {
    l->array[j+1] = l->array[j];
  }
  l->array[i] = x;
  l->size ++;
}

void expand( struct ilist *l ) {

    コード省略
}
```

```
int find_pos( struct ilist *l, int x ) {
  int lower, upper, i;
  lower  = 0;
  upper  = l->size;
  while( [     (R)     ] ) {
    i = [     (S)     ];
    if( x == l->array[i] ) {
      return( [     (T)     ] );
    }
    else if( [     (U)     ] ) {
      upper = i;
    }
    else {
      [     (V)     ] ;
    }
  }
  return( lower );
}
```

右上に続く。

(5) 設問 (4) の関数 find_pos() の計算の複雑さをオーダー記法 $O()$ で表記しなさい．ただし，リストの要素数を n とする．

(6) 設問 (4) の関数 insert() には，その実行時間がリストの要素数が増えるに従い増大するという問題がある．リストの要素数を n とする．関数 expand() の計算の複雑さが $O(1)$ である時，関数 insert() の計算の複雑さをオーダー記法 $O()$ で表記しなさい．

(7) 関数 expand() の計算の複雑さが $O(1)$ である時，設問 (4) の関数 insert() とは異なり，関数 append() の実行時間は，リストの要素数が増えても増大しない．その理由を説明しなさい．

(8) 設問 (6) の問題が深刻な場合，データ構造として配列を使うべきではない．配列の代替となるより良いデータ構造を示しなさい．その代替データ構造が元のものより良い理由を説明しなさい．

問題 V の解答に使用する解答用紙の先頭には「問題 V」と明記すること．この解答用紙には問題 V に対する解答以外を記述しないこと．

問題 V　以下の問いに答えなさい．解答に論理記号を用いる場合は次の論理記号を用いること．

表 1: 論理記号一覧

名称	記号
論理積 (AND)	$\cdot$
論理和 (OR)	$+$
排他的論理和 (XOR)	$\oplus$
否定 (NOT)	$-$ (a の否定なら $\bar{a}$)

(1) 2 進数 3 ビットのデータ $\boldsymbol{x}=(x_3, x_2, x_1)$ がある．このとき「論理関数 $f(\boldsymbol{x})$ は，$\boldsymbol{x}$ のビット列に含まれる 1 の個数が偶数ならば 0，奇数ならば 1 を返す」とする．例えば，$\boldsymbol{x}=(1,1,1)$ のとき，$f(\boldsymbol{x})$ は 1 を返す．

(a) $f(\boldsymbol{x})$ の真理値表を示しなさい．

(b) $f(\boldsymbol{x})$ を論理和標準形（積和標準形，最小項展開とも呼ぶ）で示しなさい．

(2) 2 進数 4 ビットのデータ $\boldsymbol{t}=(t_4, t_3, t_2, t_1)$ と 2 進数 3 ビットのデータ $\boldsymbol{p}=(p_3, p_2, p_1)$ があり，

$$\boldsymbol{p} = G(\boldsymbol{t}) = (t_3 \oplus t_2 \oplus t_1,\ t_4 \oplus t_3 \oplus t_2,\ t_4 \oplus t_2 \oplus t_1)$$

と定める．また，$\boldsymbol{t}$ と $\boldsymbol{p}$ のビット列を結合した，2 進数 7bit のデータ $\boldsymbol{w}$ を次のように定める．

$$\boldsymbol{w} = (w_7, w_6, w_5, w_4, w_3, w_2, w_1) = (t_4, t_3, t_2, t_1, p_3, p_2, p_1)$$

このとき $\boldsymbol{w}$ は，データ通信やデータ保存などの過程において，何らかのエラーにより任意の 1 ビットが反転しても元のデータを再現できるという性質を持っている．

(a) $\boldsymbol{t}=(1,0,1,1)$ の場合の $\boldsymbol{p}$ を求めなさい．

(b) エラーにより $\boldsymbol{w}$ のある 1 ビットが反転したデータを $(0,0,1,0,1,0,0)$ とする．このとき，元の $\boldsymbol{w}$ を求めなさい．また，導出過程も示しなさい．

(3) 設問 (2) の $\boldsymbol{t}$，$\boldsymbol{p}$，$\boldsymbol{w}$ に加えて，2 進数 4 ビットのデータ $\boldsymbol{t}'=(t_4, t_3, t_2, \overline{t_1})$，$\boldsymbol{t}''=(\overline{t_4}, \overline{t_3}, t_2, t_1)$ がある．

(a) $G(\boldsymbol{t}')$ を，$t_4,\ t_3,\ t_2,\ t_1,\ 1,\ 0,\ \oplus$ のみを用いて示しなさい．

(b) $G(\boldsymbol{t}'')$ を，$t_4,\ t_3,\ t_2,\ t_1,\ 1,\ 0,\ \oplus$ のみを用いて示しなさい．

(c) エラーにより $\boldsymbol{w}$ において最大 2 ビットが反転する場合，エラーの混入したデータから $\boldsymbol{w}$ を特定することができない場合がある．このことを論理式または真理値表を用いて説明しなさい．

Write the answers to Problem I on one answer sheet, and clearly label it at the top of the page as "Problem I." Do not write answers to other problems on the answer sheet.

Problem I When y_1 and y_2 are differentiable functions with respect to x, answer the following questions about simultaneous equations

$$\begin{aligned} y_1' &= y_1 + 2y_2, \\ y_2' &= 2y_1 - 2y_2. \end{aligned}$$

(1) When the simultaneous equations are expressed in the form $\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = A \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$, find the matrix A.

(2) Find a pair of a regular matrix P and a diagonal matrix D so that $D = P^{-1}AP$ holds.

(3) When w_1 and w_2 are differentiable functions with respect to x and $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = P^{-1} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$, show that $\begin{pmatrix} w_1' \\ w_2' \end{pmatrix} = D \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$.

(4) Find y_1 and y_2 by using the result of Question (3).
You may use the fact that $(e^{ax})' = ae^{ax}$ (a is constant) holds.

Write the answers to Problem II on one answer sheet, and clearly label it at the top of the page as "Problem II." Do not write answers to other problems on the answer sheet.

Problem II

(1) Let $x^2 + xy + y^2 = 1$. Answer the following questions.

(a) Find $\dfrac{dy}{dx}$.

(b) Find $\dfrac{d^2y}{dx^2}$.

(2) Find the following definite integral

$$\int_{-\infty}^{\infty} \frac{dx}{4x^2 + 6x + 3}.$$

Write the answers to Problem III on one answer sheet, and clearly label it at the top of the page as "Problem III." Do not write answers to other problems on the answer sheet.

Problem III Let $\mathbb{N}$ be the set of all natural numbers, including 0, and define $S = \{x \in \mathbb{N} \mid 0 \leq x \leq 4\}$. Define functions f and g_n (with $n \in \mathbb{N}$) on the Cartesian product $S \times S$ of the set S by

$$\begin{aligned} f(\langle x, y \rangle) &= \langle (x+1) \bmod 5,\ (y+2) \bmod 5 \rangle, \\ g_n(\langle x, y \rangle) &= \begin{cases} \langle x, y \rangle & (\text{if } n = 0) \\ f(g_{n-1}(\langle x, y \rangle)) & (\text{if } n \geq 1). \end{cases} \end{aligned}$$

Here, for a non-negative integer a and a positive integer b, $a \bmod b$ is the remainder after division of a by b. Define a binary relation R_m (with $m \in S$) on $S \times S$ by

$$\langle x, y \rangle R_m \langle x', y' \rangle \iff \exists k \in S\ (k \leq m \wedge \langle x', y' \rangle = g_k(\langle x, y \rangle)).$$

Answer the following questions.

(1) Find the image $f(T)$ of the set $T = \{\langle x, y \rangle \in S \times S \mid x = y\}$ under function f.

(2) Compute the value $g_3(\langle 1, 2 \rangle)$. Show your steps.

(3) Find the inverse function of g_2.

(4) If a relation is reflexive, symmetric, and transitive, it is called an equivalence relation. Show that the relation R_4 is an equivalence relation on $S \times S$.

(5) R_3 is not an equivalence relation on $S \times S$. Prove this by showing a counterexample.

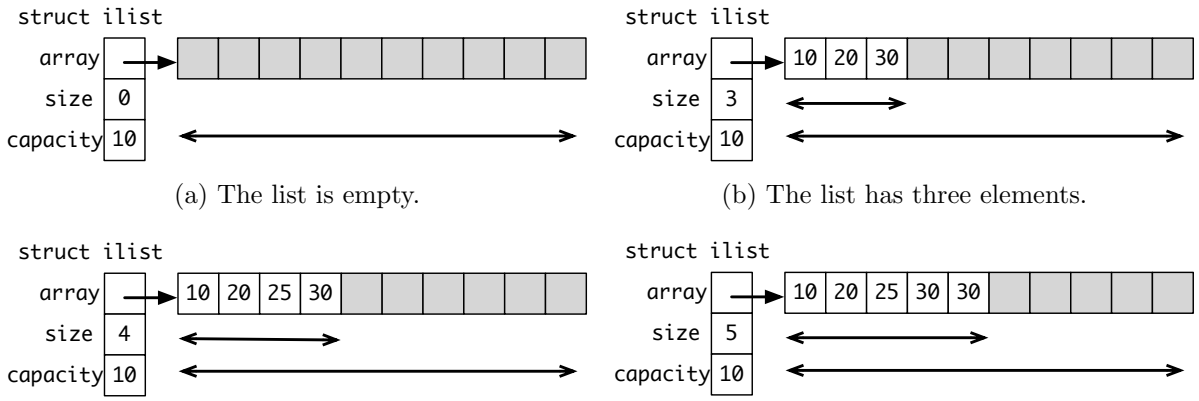
Write the answers to Problem IV on one answer sheet, and clearly label it at the top of the page as "Problem IV." Do not write answers to other problems on the answer sheet.

Problem IV

We implement a list that contains integers sorted in ascending order by using an array in the C language. Figure 1 shows the structure `ilist` for this purpose. The member `array` points to the head of an array, the member `size` stores the number of elements and the member `capacity` contains the capacity of the array. Figure 2 shows examples of the structure `ilist`. The structure `ilist` is manipulated through the functions in Table 1.

```
struct ilist {
  int *array;
  int  size;
  int  capacity;
};
```

Figure 1: The definition of the structure `ilist` that has a list of integers.



(a) The list is empty.

(b) The list has three elements.

(c) The list has four elements.

(d) The list has five elements.

Figure 2: Examples of the structure `ilist` for storing a list of integers.

Table 1: The functions that manipulate the structure `ilist`

Function	Description
`struct ilist *new_ilist()`	Allocates memory for a structure and an array, initializes the members of the structure and returns the pointer to the structure.
`void free_ilist(struct ilist *l)`	Releases the memory of the structure and its array.
`void expand(struct ilist *l)`	Expands the array in the structure and increases the capacity.
`void insert(struct ilist *l, int x)`	Inserts an integer into the sorted array in the structure in ascending order. After the insertion, the array is also sorted in ascending order.
`void append(struct ilist *l, int x)`	Appends an integer at the end of the array in the structure. When the capacity of the array is not enough, this function calls `expand()` and increases the capacity.

Answer the following questions.

(1) The following function `make_list()` allocates an empty list, adds three integers into the list and returns the list as shown in Figure 2(b). Fill in the blanks and complete the function using the functions in Table 1.

```
struct ilist *make_list() {
  struct ilist *l = new_ilist();
  (A) ( l, (B) );
  (C) ( l, 10 );
  append( l, (D) );
  return( l );
}
```

(2) The following function `uniq()` reads a list given as an argument, compares adjacent elements and returns the deduplicated list that contains each unique element. For example, if this function takes the list as in Figure 2(d), the function will return the list as in Figure 2(c). Fill in the blanks and complete the function.

```
struct ilist *uniq( struct ilist *l ) {
  struct ilist *u;
  int x, i ;
  u = new_ilist();
  if( (E) )
    return( u );
  x = (F) ;
  append( u, x );
  for( (G) ) {
    if( (H) ) {
      x = (I) ;
      append( u, x );
    }
  }
  return( u );
}
```

(3) The following function `merge()` takes two lists as arguments and merges these into a single list. This function supposes that each list in the arguments is sorted in ascending order. Fill in the blanks and complete the function.

```
struct ilist *merge( struct ilist *a,
                     struct ilist *b ) {
  struct ilist *l;
  int i, j;
  l = new_ilist();
  i=0; j=0;
  while( (J) ) {
    if( (K) ) {
      append( l, (L) );
      i++;
    }
    else {
      append( l, (M) );
      j++;
    }
  }
```

```
  if( (N) ) {
    for( ; i<a->size; i++ ) {
      append( l, (O) );
    }
  }
  else {
    for( ; j<b->size; j++ ) {
      append( l, (P) );
    }
  }
  return( l );
}
```

Continue to the upper right.

(4) The following function `insert()` inserts the integer `x` into the list `l` using the functions `expand()` and `find_pos()`. The function `insert()` supposes that the list is sorted in ascending order. Assume the function `expand()` always succeeds. The function `find_pos()` uses the binary search algorithm. Fill in the blanks and complete the functions.

```
void insert( struct ilist *l, int x ) {
  int i, j;
  if( l->size == l->capacity )
    expand( l );
  i = find_pos( l, x );
  for( [     (Q)     ] ) {
    l->array[j+1] = l->array[j];
  }
  l->array[i] = x;
  l->size ++;
}

void expand( struct ilist *l ) {
   Code omitted.
}
```

```
int find_pos( struct ilist *l, int x ) {
  int lower, upper, i;
  lower  = 0;
  upper  = l->size;
  while( [     (R)     ] ) {
    i = [     (S)     ];
    if( x == l->array[i] ) {
      return( [     (T)     ] );
    }
    else if( [     (U)     ] ) {
      upper = i;
    }
    else {
      [     (V)     ] ;
    }
  }
  return( lower );
}
```

Continue to the upper right.

(5) Show the computational complexity of the function `find_pos()` in Question (4) using the big O notation (order notation) $O()$, where the number of elements in the list is n.

(6) The function `insert()` in Question (4) has the problem that its execution time grows as the number of elements in the list increases. Let the number of elements in the list be n. When the computational complexity of the function `expand()` is $O(1)$, show the computational complexity of the function `insert()` using the big O notation $O()$.

(7) When the computational complexity of the function `expand()` is $O(1)$, unlike the function `insert()` in Question (4), the execution time of the function `append()` does not grow as the number of elements in the list increases. Explain the reason.

(8) If the problem in Question (6) is serious, we should not use an array in the structure. Show a better alternative data structure to an array. Explain why your alternative data structure is better than the original.

Write the answers to Problem V on one answer sheet, and clearly label it at the top of the page as "Problem V." Do not write answers to other problems on the answer sheet.

Problem V Answer the following questions. For all the questions, use the logic symbols in Table 1 in your answer.

Table 1: Logic Symbols

Name	Symbol
logical AND (AND)	$\cdot$
logical OR (OR)	$+$
exclusive OR (XOR)	$\oplus$
logical negation (NOT)	$-$ ($\overline{a}$ means the negation of a)

(1) Consider a binary string of length three: $\boldsymbol{x}=(x_3, x_2, x_1)$, and let a logical function $f(\boldsymbol{x})$ be the statement "$f(\boldsymbol{x})$ returns 1 if $\boldsymbol{x}$ has an odd number of ones, otherwise it returns 0." For example, $f(\boldsymbol{x})$ returns 1 when $\boldsymbol{x}=(1,1,1)$.

(a) Draw the truth table representing $f(\boldsymbol{x})$.

(b) Write down $f(\boldsymbol{x})$ in the canonical conjunctive form (CCF). CCF is also known as the maxterm expansion or the full conjunctive normal form.

(2) Consider a binary string of length four: $\boldsymbol{t}=(t_4, t_3, t_2, t_1)$ and a binary string of length three: $\boldsymbol{p}=(p_3, p_2, p_1)$, where $\boldsymbol{p}$ is defined as

$$\boldsymbol{p} = G(\boldsymbol{t}) = (t_3 \oplus t_2 \oplus t_1,\ t_4 \oplus t_3 \oplus t_2,\ t_4 \oplus t_2 \oplus t_1)$$

Also a binary string of length seven: $\boldsymbol{w}$ is defined by the concatenation of $\boldsymbol{t}$ and $\boldsymbol{p}$.

$$\boldsymbol{w} = (w_7, w_6, w_5, w_4, w_3, w_2, w_1) = (t_4, t_3, t_2, t_1, p_3, p_2, p_1) \tag{1}$$

Here, we say that $\boldsymbol{w}$ is suitable for use with data storing or data transfer. This is because when a single-bit flip error occurs in $\boldsymbol{w}$, it is possible to recover the original binary string by detecting and correcting the single-bit flip error.

(a) Show $\boldsymbol{p}$, where $\boldsymbol{t}=(1,0,1,1)$.

(b) Suppose a single-bit flip error makes $\boldsymbol{w}$ be $(0,0,1,0,1,0,0)$. Show the original data $\boldsymbol{w}$. For this question, show all the work that leads to your answer.

(3) Consider two binary strings of length four: $\boldsymbol{t'}=(t_4, t_3, t_2, \overline{t_1})$ and $\boldsymbol{t''}=(\overline{t_4}, \overline{t_3}, t_2, t_1)$ in addition to $\boldsymbol{w}$, $\boldsymbol{t}$, and $\boldsymbol{p}$ described in Question (2).

(a) Show $G(\boldsymbol{t'})$ using only $t_4,\ t_3,\ t_2,\ t_1,\ 1,\ 0$, and $\oplus$.

(b) Show $G(\boldsymbol{t''})$ using only $t_4,\ t_3,\ t_2,\ t_1,\ 1,\ 0$, and $\oplus$.

(c) If an error may cause up to two single-bit flips in $\boldsymbol{w}$, there are cases where it is not possible to recover the original data $\boldsymbol{w}$. Explain this using logical formulas or truth tables.