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　８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. 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, name of your main field, and the examination number after the examination starts.
3. この問題は全部で 18 ページ（表紙を除く）です．1〜9 ページは日本語版，10〜18 ページは英語版です．
   This booklet consists of 18 pages, excluding the cover sheet. The Japanese version is shown on pages 1–9 and the English version on pages 10–18.
4. 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 (white paper).
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２９年８月２４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　正則行列 $A = \begin{pmatrix} 1 & 3 & 1 \\ 1 & -1 & -1 \\ -1 & 3 & 3 \end{pmatrix}$ とその逆行列 B について，以下の問いに答えなさい．

(1) $D_A = P_A^{-1} A P_A$ を満たすような正則行列 P_A と対角行列 D_A を一組求めなさい．

(2) $D_B = P_B^{-1} B P_B$ を満たすような正則行列 P_B と対角行列 D_B を一組求めなさい．

(3) $A^n + B^n$ の固有値を全て求めなさい．ただし，n は 1 以上の整数である．

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) $y = e^x \sin x$ とする．以下の問いに答えよ．

(a) $y' = \sqrt{2}\, e^x \sin\left(x + \dfrac{\pi}{4}\right)$ であることを示せ．

(b) 数学的帰納法を用いて y の第 n 次導関数を求めよ．

(2) 次の定積分の値を求めよ．

$$\int_1^{\infty} \frac{dx}{x\sqrt{2x^2 - 1}}$$

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，$S = \{1, 2, 4\}$，$T = \{x \in \mathbb{N} \mid 0 \leq x \leq 7\}$ とする．また，$\mathcal{P}(X)$ は集合 X のべき集合を表すものとする．関数 $\sigma : \mathcal{P}(S) \to T$ を

$$\sigma(Y) = \sum_{x \in Y} x$$

と定義する．ただし，Y が空集合のときは $\sigma(Y) = 0$ とする．すなわち，$\sigma(Y)$ は集合 Y の要素を全て足し合わせた値を返す関数であるとする．例えば，$\sigma(\{1, 2\}) = 3$ であるとする．

ここで，二項関係 $R_1, R_2 \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$ を，それぞれ

- $R_1 = \{(A, B) \mid \sigma(A) \leq \sigma(B)\}$
- $R_2 = \{(A, B) \mid \sigma(A) - \sigma(B) = \sigma(A \cup B) - \sigma(A \cap B)\}$

と定義する．このとき，以下の問いに答えなさい．なお，以下で多重集合は考えないものとする．

(1) $(A, \{4\}) \in R_1$ を満たす $A \in \mathcal{P}(S)$ を全て求めなさい．

(2) 反射律・推移律・反対称律を全て満たす二項関係を，一般に順序関係という．関係 R_1 は $\mathcal{P}(S)$ 上の順序関係となることを示しなさい．

(3) 有向グラフ $G = (V, E)$ を以下のように定義する．

- 頂点の集合 V を $\mathcal{P}(S)$ とする．
- 辺の集合 E は，$(A, B) \in R_2$ が成り立つとき，かつそのときに限り，頂点 A から頂点 B への辺をちょうど 1 本含む．

このとき，G の頂点の数と辺の数をそれぞれ求めなさい．

(4) 関数 σ が全単射となることを示しなさい．

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

整数と表 1 に示した演算子で構成された数式を木構造で表現する．図 1 に，そのような数式を表す木構造の例を示す．木構造を深さ優先で走査する方法として，先順 (pre-order, 前順)，中順 (in-order, 間順)，後順 (post-order) がある．図 1(A) に示した木構造を中順で走査すると，次のような中置記法による数式が得られる．ただし，この数式は計算の順序を示すために括弧を含んでいる．

```
((10 + 20) - 30)
```

数式を表す木構造を後順で走査すると，後置記法 (逆ポーランド記法) による数式が得られる．図 1(A) に示した木構造を後順で走査すると，次のような数式が得られる．ただし，この数式で要素の区切りは空白である．

```
10 20 + 30 -
```

表 1: 整数を扱う演算子

演算子	説明
+	2 つの引数を取り，それらの値を加え，その結果を返す．
-	2 つの引数を取り，最初の値から 2 番目の値を引き，その結果を返す．
==	2 つの引数を取り，それらの値を比較する．それらが等しい場合，真 (0 ではない値) を返す．そうではない場合，偽 (0) を返す．
!	1 つの引数を取り，それが真の場合，偽を返す．そうではない場合，真を返す．
?	3 つの引数を取り，最初の値をチェックする．最初の値が真の場合，2 番目の値を返す．そうではない場合，3 番目の値を返す．

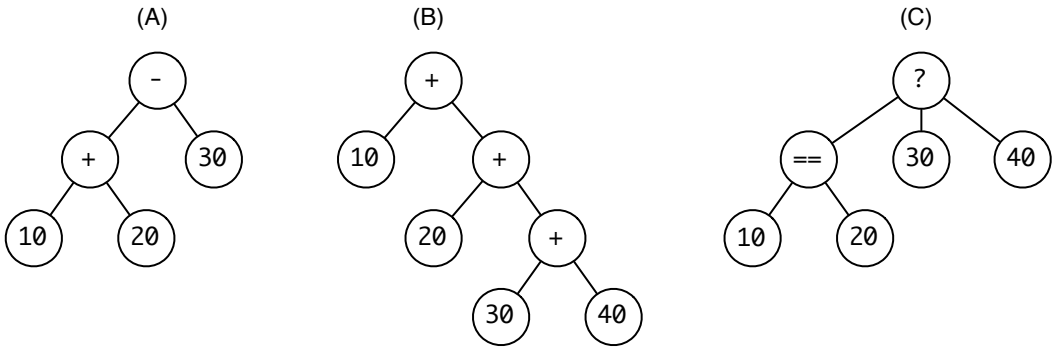

図 1: 数式を表す木構造の例

次の小問に答えなさい．ただし，以下の図に示されたプログラムは，C 言語で記述されている．

(1) 図 1(B)，および，図 1(C) に示した木構造を，後置記法で表記しなさい．

(2) 次の 2 つの数式は，後置記法で表記された木構造を表している．これらの木構造を，図 1 と同じ形式で図示しなさい．

```
(D) 1 2 + 3 4 - +
(E) 1 2 == ! 0 1 ?
```

(3) 図 2 に示した関数 `calc()` は，後置記法で表記された数式をスタックを用いて計算する．この関数は，文字列の配列で表現された数式を引数として取る．配列の各要素は，整数を表す文字列，または，演算子を表す文字列である．配列は，`NULL` ポインタで終端されている．この関数は，計算の途中結果をスタックに保存する．関数 `calc()` が利用しているスタックは，表 2 に示した関数で操作される．また，関数 `calc()` は，表 3 に示した関数を利用している．ただし，関数 `calc()` はエラー処理に問題がある．

図 3 に示した関数 `main()` が実行された時，画面にどのような出力がなされるかを示しなさい．

表 2: スタックを操作する関数

関数	説明
`void init_stack()`	スタックを空にし，大域変数 `error` を `0` に初期化する．
`void push(int x)`	スタックに整数 `x` を積む．
`int pop()`	スタックが空の場合には，大域変数 `error` に `1` をセットし，`0` を返す．このエラーは，スタック・アンダーフローとして知られている．そうではない場合，スタックから整数を取り出し，それを返す．
`int free_stack()`	スタックが空の場合，真を返す．そうではない場合，スタックの全ての要素を解放し，偽を返す．

表 3: 関数 `calc()` が文字，文字列，および，整数を操作するために利用する関数

関数	説明
`strcmp()`	2 つの文字列を引数に取り，それらが等しい場合，`0` を返す．そうではない場合，`0` 以外の値を返す．
`isdigit()`	文字を引数に取り，それが数字 (`'0'` から`'9'` の間) である場合に真，そうではない場合に偽を返す．
`atoi()`	数字から構成される文字列を引数に取り，その文字列を 10 進数とみなし整数に変換し，その整数を返す．

```
#define NULL    ((void *)0)
#define OK      1
#define ERROR   0

extern int error;

int calc( char *exp[], int *xp ) {
  init_stack();
  for( int i=0; exp[i]!=NULL; i++ ) {
    if( isdigit(exp[i][0]) )
      do_num( exp[i] );
    else if( strcmp(exp[i],"+") == 0 )
      do_add();
    else if( strcmp(exp[i],"-") == 0 )
      do_sub();
    else if( strcmp(exp[i],"==") == 0 )
      do_eq();
    else if( strcmp(exp[i],"!") == 0 )
      do_not();
    else if( strcmp(exp[i],"?") == 0 )
      do_cond();
    else {
      free_stack();
      return( ERROR );
    }
  }
  *xp = pop();
  free_stack();
  if( error ) /*(X)*/
    return( ERROR );
  else
    return( OK );
}
```

```
void do_num( char *s ) {
  int a;
  a = atoi( s );
  push( a );
}

void do_add() {
  int a, b;
  b = pop();
  a = pop();
  push( a+b );
}

void do_eq() {
  int a, b;
  b = pop();
  a = pop();
  push( a==b );
}

void do_not() {
  int a;
  a = pop();
  push( !a );
}
```

右上に続く。

図 2: 後置記法の数式を計算する関数 calc()

```
int main() {
  int x = 0;
  char *exp[] = { "30", "50", "+", NULL };
  if( calc( exp, &x )==OK )
    printf("OK. The result is %d.\n", x );
  else
    printf("Error.\n");
  return( 0 );
}
```

図 3: 関数 calc() の利用例

(4) 図 4 の空欄を埋めて，関数 do_sub()，および，do_cond() を完成させなさい．

```
void do_sub() {
  int a, b;
    (F)    ;
    (G)    ;
  push(    (H)    );
}
```

```
void do_cond() {
  int a, b, c;
    (I)    ;
  b = pop()    ;
    (J)    ;
  if( a )
      (K)    ;
  else
      (L)    ;
}
```

右上に続く。

図 4: 関数 do_sub() と do_cond()

(5) 図 2 の関数 calc() は，実行中にスタック・アンダーフローのエラーが生じ，ERROR を返す時がある．そのような入力の例を 1 つ示しなさい．

(6) 図 2 の関数 calc() は,「/*(X)*/」と印を付けた所で大域変数 error が真の場合，ERROR を返す．そうではない場合，この関数は，OK を返す．しかし，入力の数式がある種のエラーを含んでいる時，そのエラーを検出できず，OK を返すことがある．そのような入力の例を 1 つ示しなさい．

(7) 小問 (6) の問題を解決するために，関数 calc() を修正する．関数 calc() への修正方法の概略を示しなさい．

(8) 表 2 に示した関数を，リストを用いて実装する．図 5 の空欄を埋めて，関数 push()，および，pop() を完成させなさい．ただし，関数 malloc() は常に成功するものとする．

```
#define NULL    ((void *)0)
struct element {
  struct element *next;
  int data;
};

struct element *sp = NULL;
int error = 0;

void push( int x ) {
  struct element *n;
  n = malloc(
    sizeof(struct element) );
    (M)    ;
    (N)    = x;
    (O)    ;
}
int free_stack() {
    コード省略
}
```

```
int pop() {
  int x;
  struct element *f;
  if(    (P)    ) {
    error = 1;
    return( 0 );
  }
    (Q)    ;
  f = sp;
    (R)    ;
  free( f );
  return( x );
}
void init_stack() {
  free_stack();
  sp = NULL;
  error = 0;
}
```

右上に続く。

図 5: リストによるスタックの実装

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　正の整数 N_1，N_2 を整数変数 x，y に読み込み，昇順に並べ替えて，print 関数により標準出力に出力するプログラムを考える．

Program 1　N_1 と N_2 を，昇順に並び替えて，標準出力に出力するプログラム（変更前）

```
 1: function ORDERING
 2:     x ← N_1
 3:     y ← N_2
 4:     if x is greater than y then
 5:         y ← y + x
 6:         x ← y − x
 7:         y ← y − x
 8:     end if
 9:     print x
10:     print y
11: end function
```

ただし，Program 1 は，N_1 と N_2 の値によっては算術オーバーフローが発生し，エラーが発生した．そこで，Program 2 を考えた．

Program 2　N_1 と N_2 を，昇順に並び替えて，標準出力に出力するプログラム（変更後）

```
 1: function ORDERING
 2:     x ← N_1
 3:     y ← N_2
 4:     if x is greater than y then
 5:         y ← y [①] x
 6:         x ← y [②] x
 7:         y ← y [③] x
 8:     end if
 9:     print x
10:     print y
11: end function
```

(1) N_1，N_2 は 4 ビットで表現され，x，y を 4 ビットの符号無し整数変数とする．ここで，N_1 を横軸，N_2 を縦軸とした 2 次元平面を考えたとき，Program 1 において算術オーバーフローが発生する N_1 と N_2 の組み合わせをこの平面上に示しなさい．

(2) N_1，N_2 は 4 ビットで表現され，x，y を 4 ビットの符号無し整数変数とする．また，x および y の各ビットを $x_3\,x_2\,x_1\,x_0$ および $y_3\,y_2\,y_1\,y_0$ とする．このとき，Program 2 の① ② ③に適する論理記号をそれぞれ答えなさい．この解答における論理記号は表 1 の論理記号を使用すること．また，x_3 と y_3 に関して，Program 2 の 5, 6, 7 行目の各処理により値が交換される過程を真理値表を用いて示しなさい．

(3) N_1，N_2 は 2 ビットで表現され，x，y を 2 ビットの符号無し整数変数とする．また，x および y の各ビットを $x_1\,x_0$ および $y_1\,y_0$ とする．このとき，Program 1 および Program 2 の 4 行目にある条件式 “x is greater than y” について考える．この解答における論理記号は表 1 の論理記号を使用すること．

(a) 条件式が成立する場合に真 (1) を，それ以外は偽 (0) を返す論理関数 $F(x_1, x_0, y_1, y_0)$ を，和積標準形（論理積標準形または主乗法標準形）で答えなさい．

(b) 条件式が成立する場合に真 (1) を，それ以外は偽 (0) を返す論理関数 $F(x_1, x_0, y_1, y_0)$ を，カルノー図を用いて簡略化し，以下のように導いた．

$$F = \Big((x_0 \boxed{\text{ⓐ}} \neg\, y_0) \boxed{\text{ⓑ}} (x_1 \boxed{\text{ⓒ}} \neg\, y_1)\Big) \boxed{\text{ⓓ}} (x_1 \boxed{\text{ⓔ}} \neg\, y_1)$$

このとき，ⓐ ⓑ ⓒ ⓓ ⓔ に適する論理記号をそれぞれ答えなさい．

表 1: 論理記号一覧

意味	論理記号
論理積 (AND)	$\wedge$
論理和 (OR)	$\vee$
排他的論理和 (XOR)	$\nleftrightarrow$

意味	論理記号
否定論理積 (NAND)	$\uparrow$
否定論理和 (NOR)	$\downarrow$
否定排他的論理和 (EXNOR)	$\leftrightarrow$
否定 (NOT)	$\neg$

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 Answer the following questions about the regular matrix $A = \begin{pmatrix} 1 & 3 & 1 \\ 1 & -1 & -1 \\ -1 & 3 & 3 \end{pmatrix}$ and the inverse matrix B of A.

(1) Find a pair of a regular matrix P_A and a diagonal matrix D_A so that $D_A = P_A^{-1} A P_A$ holds.

(2) Find a pair of a regular matrix P_B and a diagonal matrix D_B so that $D_B = P_B^{-1} B P_B$ holds.

(3) Find all the eigenvalues of $A^n + B^n$ when n is an integer that is larger than or equal to 1.

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 $y = e^x \sin x$. Answer the following questions.

(a) Show that $y' = \sqrt{2}\, e^x \sin\left(x + \dfrac{\pi}{4}\right)$.

(b) Find the n-th derivative of y using mathematical induction.

(2) Find the following definite integral.

$$\int_1^\infty \frac{dx}{x\sqrt{2x^2 - 1}}$$

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 $S = \{1, 2, 4\}$ and $T = \{x \in \mathbb{N} \mid 0 \leq x \leq 7\}$. Let $\mathcal{P}(X)$ denote the power set of a set X. Define a function $\sigma : \mathcal{P}(S) \to T$ by

$$\sigma(Y) = \sum_{x \in Y} x.$$

Here, for the case that Y is the empty set, define $\sigma(Y) = 0$. That is, function σ returns the sum of all elements of Y. For example, $\sigma(\{1, 2\}) = 3$.

Define binary relations $R_1, R_2 \subseteq \mathcal{P}(S) \times \mathcal{P}(S)$ by

- $R_1 = \{(A, B) \mid \sigma(A) \leq \sigma(B)\}$,
- $R_2 = \{(A, B) \mid \sigma(A) - \sigma(B) = \sigma(A \cup B) - \sigma(A \cap B)\}$,

respectively.

Answer the following questions, where we assume that multisets are not considered.

(1) Find all $A \in \mathcal{P}(S)$ satisfying $(A, \{4\}) \in R_1$.

(2) If a relation is reflexive, transitive, and antisymmetric, it is called an ordered relation. Show that the relation R_1 is an ordered relation on $\mathcal{P}(S)$.

(3) Let $G = (V, E)$ be the directed graph such that

- the set V of vertices is $\mathcal{P}(S)$,
- the set E of edges includes exactly one edge from vertex A to vertex B if and only if $(A, B) \in R_2$.

Find the numbers of vertices and edges of G, respectively.

(4) Show that the function σ is bijective.

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 represent numerical expressions that consist of integers and the operators in Table 1 using tree structures. Figure 1 shows examples of such tree structures. There are three methods of depth-first tree traversals: pre-order, in-order, and post-order. If we traverse the tree structure in Figure 1(A) using in-order, we get the following numerical expression in infix notation. Note that this expression includes parentheses to clarify the order of calculation.

```
((10 + 20) - 30)
```

If we traverse a tree structure using post-order, we get a numerical expression in postfix notation, also known as reverse Polish notation. If we traverse the tree structure in Figure 1(A) using post-order, we get the following numerical expression. Note that the delimiter of elements is a space in this expression.

```
10 20 + 30 -
```

Table 1: The operators for integers.

Operator	Description
`+`	Takes two arguments, adds these values, and returns the result.
`-`	Takes two arguments, subtracts the second value from the first value, and returns the result.
`==`	Takes two arguments and compares these two values. If they are equal, it returns true (a non-zero value). Otherwise, it returns false (`0`).
`!`	Takes an argument and returns true if the value is false. Otherwise, it returns false.
`?`	Takes three arguments and checks the first value. It returns the second value if the first value is true. Otherwise, it returns the third value.

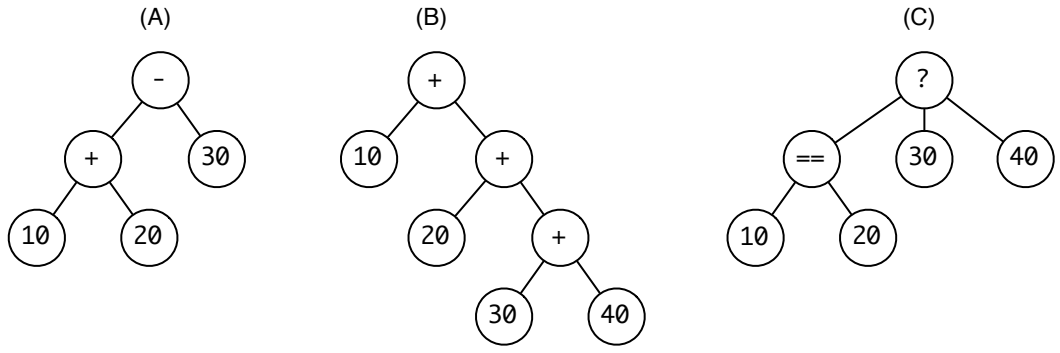

Figure 1: Three tree structures that describe numerical expressions.

Answer the following questions. Note that the programs shown in the following figures are written in the C language.

(1) Write the tree structures in Figures 1(B) and 1(C) using postfix notation.

(2) The following two numerical expressions in postfix notation represent two tree structures. Illustrate these tree structures using the style used in Figure 1.

```
(D) 1 2 + 3 4 - +
(E) 1 2 == ! 0 1 ?
```

(3) The function `calc()` in Figure 2 calculates a numerical expression in postfix notation using a stack. This function takes a numerical expression that is represented by an array of strings as an argument. Each element of the array is either a string that represents an integer or a string that represents an operator. The array is terminated with a `NULL` pointer. This function stores intermediate results in a stack. The stack used by the function `calc()` is manipulated through the functions in Table 2. Further, the function `calc()` uses the functions in Table 3. Note that the function `calc()` has a problem in error handling.

When the function `main()` in Figure 3 is executed, what is the output of the computer screen?

Table 2: The functions that manipulate the stack.

Function	Description
`void init_stack()`	Clears the stack and initializes the global variable `error` to `0`.
`void push(int x)`	Adds an integer `x` onto the stack.
`int pop()`	If the stack is empty, this function sets the global variable `error` to `1` and returns `0`. This error is known as stack underflow. Otherwise, this function removes an integer from the stack and returns the integer.
`int free_stack()`	Returns true if the stack is empty. Otherwise, this function releases all the element(s) in the stack and returns false.

Table 3: The functions that are used by `calc()` to manipulate characters, strings, and integers.

Function	Description
`strcmp()`	Takes two strings as arguments, compares these strings, and returns `0` if these two strings are equal. Otherwise, this function returns a non-zero value.
`isdigit()`	Takes a character as an argument, and returns true if the character is a decimal digit character (from `'0'` to `'9'`). Otherwise, this function returns false.
`atoi()`	Takes a string that consists of decimal digit characters as an argument, converts the string to an integer as a decimal number, and returns the integer.

```
#define NULL    ((void *)0)
#define OK      1
#define ERROR   0

extern int error;

int calc( char *exp[], int *xp ) {
  init_stack();
  for( int i=0; exp[i]!=NULL; i++ ) {
    if( isdigit(exp[i][0]) )
      do_num( exp[i] );
    else if( strcmp(exp[i],"+") == 0 )
      do_add();
    else if( strcmp(exp[i],"-") == 0 )
      do_sub();
    else if( strcmp(exp[i],"==") == 0 )
      do_eq();
    else if( strcmp(exp[i],"!") == 0 )
      do_not();
    else if( strcmp(exp[i],"?") == 0 )
      do_cond();
    else {
      free_stack();
      return( ERROR );
    }
  }
  *xp = pop();
  free_stack();
  if( error ) /*(X)*/
    return( ERROR );
  else
    return( OK );
}
```

Continue to the upper right.

```
void do_num( char *s ) {
  int a;
  a = atoi( s );
  push( a );
}

void do_add() {
  int a, b;
  b = pop();
  a = pop();
  push( a+b );
}

void do_eq() {
  int a, b;
  b = pop();
  a = pop();
  push( a==b );
}

void do_not() {
  int a;
  a = pop();
  push( !a );
}
```

Figure 2: The function `calc()` that calculates a numerical expression in postfix notation.

```
int main() {
  int x = 0;
  char *exp[] = { "30", "50", "+", NULL };
  if( calc( exp, &x )==OK )
    printf("OK. The result is %d.\n", x );
  else
    printf("Error.\n");
  return( 0 );
}
```

Figure 3: An example of using the function `calc()`.

(4) Fill in the blanks in Figure 4 and complete the functions `do_sub()` and `do_cond()`.

```
void do_sub() {
  int a, b;
    (F)    ;
    (G)    ;
  push(    (H)    );
}
```

```
void do_cond() {
  int a, b, c;
    (I)    ;
  b = pop()    ;
    (J)    ;
  if( a )
      (K)    ;
  else
      (L)    ;
}
```

Continue to the upper right.

Figure 4: The functions `do_sub()` and `do_cond()`.

(5) The function `calc()` in Figure 2 can return `ERROR` when a stack underflow error has occurred during the execution of `calc()`. Show an example of such an input.

(6) The function `calc()` in Figure 2 returns `ERROR` if the global variable `error` is true at the point of the mark `/*(X)*/`. Otherwise, this function returns `OK`. However, when an input expression includes a certain kind of error, the function `calc()` cannot detect the error and will return `OK`. Show an example of such an input.

(7) We can modify the function `calc()` to fix the problem in Question (6). Sketch the modifications to the function `calc()`.

(8) We implement the functions in Table 2 using a list. Fill in the blanks in Figure 5 and complete the functions `push()` and `pop()`. Note that the function `malloc()` always succeeds.

```
#define NULL    ((void *)0)
struct element {
  struct element *next;
  int data;
};

struct element *sp = NULL;
int error = 0;

void push( int x ) {
  struct element *n;
  n = malloc(
    sizeof(struct element) );
    (M)    ;
    (N)    = x;
    (O)    ;
}
int free_stack() {
   Code omitted.
}
```

```
int pop() {
  int x;
  struct element *f;
  if(    (P)    ) {
    error = 1;
    return( 0 );
  }
    (Q)    ;
  f = sp;
    (R)    ;
  free( f );
  return( x );
}
void init_stack() {
  free_stack();
  sp = NULL;
  error = 0;
}
```

Continue to the upper right.

Figure 5: The implementation of the stack using a list.

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 Suppose we are given a function defined with the following program: it reads two positive integers, N_1 and N_2, and sets their values to integer variables, x and y, respectively. The program arranges these values in ascending order and places the results on the standard output by using a print function.

Program 1 N_1 and N_2 are output on the standard output in ascending order (original)

```
 1: function ORDERING
 2:     x ← N_1
 3:     y ← N_2
 4:     if x is greater than y then
 5:         y ← y + x
 6:         x ← y − x
 7:         y ← y − x
 8:     end if
 9:     print x
10:     print y
11: end function
```

In Program 1, certain values of N_1 and N_2 cause arithmetic overflow, and it invites errors. Thus, Program 2 is proposed.

Program 2 N_1 and N_2 are output on the standard output in ascending order (modified)

```
 1: function ORDERING
 2:     x ← N_1
 3:     y ← N_2
 4:     if x is greater than y then
 5:         y ← y [①] x
 6:         x ← y [②] x
 7:         y ← y [③] x
 8:     end if
 9:     print x
10:     print y
11: end function
```

(1) Let N_1 and N_2 be 4-bit integers, and let x and y be 4-bit unsigned integer variables, respectively. Consider a two-dimensional plane where N_1 is plotted on the horizontal axis and N_2 is plotted on the vertical axis. By using this plane, represent the combinations of N_1 and N_2 that cause arithmetic overflow in Program 1.

(2) Let N_1 and N_2 be 4-bit integers, and let x and y be 4-bit unsigned integer variables, respectively. And, let $x_3\,x_2\,x_1\,x_0$ and $y_3\,y_2\,y_1\,y_0$ be the individual bits of x and y. Fill the appropriate logic symbols in ①, ②, and ③ in Program 2. Use the logic symbols in Table 1 in your answer. Besides, use truth tables to show how the processing of lines 5, 6, and 7 in Program 2 results in the swapping of x_3 and y_3, line by line.

(3) Let N_1 and N_2 be 2-bit integers, and let x and y be 2-bit unsigned integer variables, respectively. And let $x_1\,x_0$ and $y_1\,y_0$ be the individual bits of x and y. Consider the conditional expression "x is greater than y" on line 4 in Programs 1 and 2. Use the logic symbols in Table 1 in your answer.

(a) Write down a logical function $F(x_1, x_0, y_1, y_0)$ in canonical conjunctive form (CCF), which returns TRUE (1) if the conditional expression is true, otherwise returns FALSE (0). CCF is also known as the maxterm expansion or full conjunctive normal form.

(b) We simplified a logical function $F(x_1, x_0, y_1, y_0)$, which returns TRUE (1) if the conditional expression is true, otherwise returns FALSE (0), as follows.

$$F = \Big((x_0 \boxed{\text{ⓐ}} \neg\, y_0) \boxed{\text{ⓑ}} (x_1 \boxed{\text{ⓒ}} \neg\, y_1)\Big) \boxed{\text{ⓓ}} (x_1 \boxed{\text{ⓔ}} \neg\, y_1).$$

Choose the appropriate logic symbols for ⓐ, ⓑ, ⓒ, ⓓ and ⓔ.

Table 1: Logic Symbols

State	Symbol
logical AND (AND)	$\wedge$
logical OR (OR)	$\vee$
exclusive OR (XOR)	$\nleftrightarrow$

State	Symbol
negative AND (NAND)	$\uparrow$
negative OR (NOR)	$\downarrow$
exclusive NOR (EXNOR)	$\leftrightarrow$
logical NOT (NOT)	$\neg$