平 成 ２７ 年 度

名古屋大学大学院情報科学研究科
情報システム学専攻
入 学 試 験 問 題

専　　門

平成２６年８月７日（木）
１２：３０～１５：３０

注 意 事 項

1．試験開始の合図があるまでは、この問題冊子を開いてはならない。

2．試験終了まで退出できない。

3．外国人留学生は英語で解答してよい。また、和英辞書などの辞書を１冊に限り使用してよい。電子辞書の持ち込みは認めない。

4．問題冊子、解答用紙3枚、草稿用紙3枚が配布されていることを確認せよ。

5．問題は(１)解析・線形代数、(２)確率・統計、(３)プログラミング、(４)計算機理論、(５)ハードウェア、(６)ソフトウェアの６科目がある。(４)~(６)の３科目から少なくとも１科目を選択して解答し、(１)~(３)を含めた６科目から合計３科目を選択して解答せよ。
なお、選択した科目名を解答用紙の指定欄に記入せよ。

6．解答用紙は指定欄に受験番号を必ず記入せよ。解答用紙に受験者の氏名を記入してはならない。

7．解答用紙に書ききれない場合は、裏面を使用してもよい。
ただし、裏面を使用した場合は、その旨、解答用紙表面右下に明記せよ。

8．解答用紙はホッチキスを外さず、試験終了後に3枚とも提出せよ。

9．問題冊子、草稿用紙は試験終了後に持ち帰ってよい。

解析・線形代数

（解の導出過程も書くこと）

[1] xy平面上の曲線$y = a^2 - x^2$とx軸とで囲まれる部分D、及び、Dをx軸のまわりに回転してできる立体Tについて、以下の問いに答えよ。ただし、$a > 0$とする。

(a) Tの体積をaを使って記せ。

(b) Dをy軸のまわりに回転してできる立体の体積がTの体積に等しくなるとき、aの値を求めよ。

[2] 次の行列Aについて、以下の問いに答えよ。ただし、nは正の整数を表す。

$$A = \begin{pmatrix} 3 & -2 \\ 1 & 6 \end{pmatrix}$$

(a) Aのすべての固有値を求めよ。また、それに対応する固有ベクトルを求めよ。

(b) A^nを求めよ。

(c) 数列$\{x_n\}, \{y_n\}$が漸化式

$$x_{n+1} = 3x_n - 2y_n,\ y_{n+1} = x_n + 6y_n,\ x_1 = 1,\ y_1 = 1$$

で与えられているとき、$\{x_n\}, \{y_n\}$の一般項を求めよ。

[3] 次の複素関数について、以下の問いに答えよ。ただし、iは虚数単位を表す。

$$w = z^2 \tag{1}$$

(a) $z = x + iy$, $w = u + iv$とおくとき、u, vをx, yを用いて表せ。

(b) (1)により、z平面上の直線$z = 1 + it$ $(-\infty < t < \infty)$をw平面上に写してできる図形を求め、その概形を図示せよ。

(c) (1)により、z平面上の点$1, i, 1+i$を頂点とする三角形の周をw平面上に写してできる図形を求め、その概形を図示せよ。

Translation of technical terms

平面	plane	曲線	curve
軸	axis	回転	rotation
立体	solid	体積	volume
行列	matrix	正の整数	positive integer
固有値	eigenvalue	固有ベクトル	eigenvector
数列	progression	漸化式	recurrence formula
一般項	general term	複素関数	complex function
虚数単位	imaginary unit	直線	line
図形	shape	概形	rough sketch
点	point	頂点	vertex
三角形	triangle	周	perimeter

確率・統計

解の導出過程も書くこと.

[1] 袋に赤玉2つと白玉5つの計7つの玉が入っている．袋から無作為に玉を1つずつ取り出す試行を，袋が空になるまで繰り返す．一度取り出した玉は袋に戻さない．2つ目の赤玉が出るときの試行回数を X $(2 \leq X \leq 7)$ とする．以下の問いに答えよ．但し，答えは既約分数で示せ．

(1) 確率 $P(X = 3)$ を求めよ.

(2) 確率 $P(X \leq 4)$ を求めよ.

(3) 期待値 $E[X]$ を求めよ.

[2] 確率変数 X, Y の同時確率密度関数 $f_{X,Y}(x, y)$ が次式で与えられている．但し，c は定数とする．以下の問いに答えよ.

$$f_{X,Y}(x, y) = \begin{cases} c(y - x) & (0 \leq x \leq 1,\ 1 \leq y \leq 3) \\ 0 & (\text{その他}) \end{cases}$$

(1) 定数 c の値を求めよ.

(2) 周辺確率密度関数 $f_Y(y)$ を求めよ.

(3) 条件付き確率密度関数 $f_{X|Y}(x \mid y)$ を求めよ.

[3] X, Y は互いに独立で同一の分布に従う確率変数であり，X の確率密度関数 $f_X(x)$ が次式で与えられている．以下の問いに答えよ.

$$f_X(x) = \begin{cases} 2e^{-2x} & (0 \leq x) \\ 0 & (\text{その他}) \end{cases}$$

(1) 累積分布関数 $F_X(x)$ を求めよ.

(2) 確率変数 $Z = \min\{X, Y\}$ の累積分布関数 $F_Z(z)$ を求めよ.

[4] N 人の生徒 $(N \geq 2)$ が数学と物理の試験を受けた．i 番目の生徒の数学と物理の点数をそれぞれ x_i, y_i $(1 \leq i \leq N)$ とするとき，数学と物理の点数の相関係数を求める式を示せ．但し，数学，物理ともに N 人の点数が全て同じということはなかった.

Translation of technical terms

無作為に at random, 試行 trial, 試行回数 number of trials, 既約分数 irreducible fraction, 確率 probability, 期待値 expectation, 確率変数 random variable, 同時確率密度関数 joint probability density function, 定数 constant, 周辺確率密度関数 marginal probability density function, 条件付き確率密度関数 conditional probability density function, 互いに独立で同一の分布に従う independent and identically distributed, 確率密度関数 probability density function, 累積分布関数 cumulative distribution function, 相関係数 correlation coefficient

プログラミング

ハッシュを用いて文字列を検索するプログラムを実装したい．プログラムPはこれを実現するためのC言語プログラムである．構造体listは1つの文字列に対応する情報を格納し，hash_tableは検索対象の文字列を記憶する配列である．このプログラムについて以下の問いに答えよ．なお，プログラム中の演算子，関数，アスキーコードの説明を問いの後に示す．

(1) 空欄ア～エを埋めよ．

(2) 標準出力に出力されるこのプログラムの実行結果を書け．

(3) 66行目のinit_hash_table()を実行した直後のhash_table[HASHSIZE]（14行目）が保持する文字列wordの内容を下の形式で書け．nextにNULL以外の値が入る場合，該当するwordを「→」を使って連結せよ．

例:

添え字	word
0	red
1	NULL
2	black → blue → green
・・・	・・・

(4) 7行目のN_WORDSの値を4から5に変更し，15行目を
char colors[N_WORDS][MAX_LEN] = {"red", "blue", "green", "yellow", "pink"};
とする．66行目のinit_hash_table()を実行した直後のhash_table[HASHSIZE]（14行目）の内容を問い(3)の形式で書け．

(5) 以下の空欄を埋める形式で45行目のmy_strcpyの機能を満たすよう回答せよ(ただし他の関数を呼び出さないこと)．

```
void my_strcpy(char* a, const char* b)
{
  int i = 0;
  while( [   1   ] ) {
     [   2   ];
     i++;
  }
  [   3   ];
}
```

演算子，関数，アスキーコードの説明

24 行目の演算子「%」は剰余演算子であり「x % y」は x を y で割った余りである．

28 行目の関数 find_word(char * key)は文字列 key が hash_table に存在するとき 1，存在しないとき 0 を返す．

32 行目の関数 strcmp(const char* a, const char* b)は文字列 a，b が一致するとき 0 を，一致しないときは 0 以外を返す関数である．

45 行目の関数 my_strcpy(char* a, const char* b)は文字列 a に文字列 b をコピーする関数である．

char 配列の各文字はアスキーコードとして保持される．アルファベットに対応する 10 進数の値は以下の表のとおりである．

アルファベット	a	b	c	d	e	f	g	h	i	j	k
アスキーコード	97	98	99	100	101	102	103	104	105	106	107
アルファベット	l	m	n	o	p	q	r	s	t	u	v
アスキーコード	108	109	110	111	112	113	114	115	116	117	118
アルファベット	w	x	y	z							
アスキーコード	119	120	121	122							

Translation of technical terms

ハッシュ	hash	演算子	operator
文字列	string	関数	function
プログラム	program	アスキーコード	ASCII code
C 言語	C language	剰余演算子	modulus operator
構造体	structure	メモリ	memory
配列	array	ポインタ	pointer

プログラムP

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define HASHSIZE    10
6  #define MAX_LEN     64
7  #define N_WORDS      4
8
9  struct list {                                /* 1文字列用の構造体          */
10   char word[MAX_LEN];                        /* 文字列を記憶しておく場所    */
11   struct list *next;                         /* 次の list へのポインタ      */
12 };
13
14 struct list *hash_table[HASHSIZE];          /* ハッシュテーブル */
15 char colors[N_WORDS][MAX_LEN] = {"red", "blue", "green", "yellow"};
16
17 int hash(char *key)
18 {
19   int hashval = 0;
20   while (*key != '¥0') {                     /* 与えられた文字列のハッシュ値を求める */
21     hashval += *key;
22     key++;
23   }
24   return (hashval % HASHSIZE);               /* ハッシュ値を返す */
25 }
26
27 /* find_word は key が hash_table に存在するとき 1、存在しないとき 0 を返す */
28 int find_word(char *key)
29 {
30   struct list *p;
31   for (p = hash_table[hash(key)]; p != NULL; [ ア ])
32     if (strcmp(key, p->word) == 0) return 1;
33   return 0;
34 }
```

```
35 void init_hash_table()
36 {
37   int i, hashvalue;
38   struct list *p, *q;
39   for (i = 0; i < [  イ  ]; i++) { hash_table[i] = NULL; }
40   for (i = 0; i < N_WORDS; i++) {
41     if ((find_word(colors[i])) == [  ウ  ]) {
42       /* ポインタ p に新しい文字列用のメモリを割り当てる */
43       p = (struct list *)malloc(sizeof(struct list));
44
45       my_strcpy(p->word, colors[i]); /*割り当てたメモリに文字列をコピーする*/
46       hashvalue = hash(colors[i]);
47
48       if (hash_table[hashvalue] == NULL) {
49         /* 文字列がなければ新しい単語をそのまま追加 */
50         hash_table[hashvalue] = p;
51         [  エ  ];
52       } else {        /* 文字列がすでにあれば現在のリストの末尾に新しい単語を追加 */
53         q = hash_table[hashvalue];
54         while (q->next != NULL) q = q->next;
55         q->next = p;
56         [  エ  ];
57       }
58     }
59   }
60 }
61
62
63 void main(void)
64 {
65   /* 配列 colors の文字列を登録する */
66   init_hash_table();
67   /* "red"が登録した文字列に含まれるか確認する */
68   printf ("result = %d¥n", find_word("red"));
69 }
```

[2] アルファベット $\{0,1\}$ 上の言語について以下の問いに答えよ.

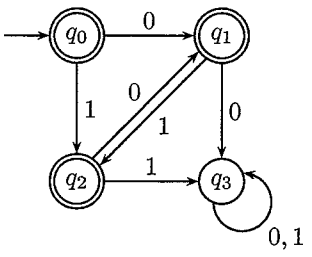
(1) 長さが1以上である入力記号列 $a_1a_2\ldots a_n$ (各 a_i は0もしくは1のどちらか) を0以上の2進数 $[a_1a_2\ldots a_n]_2$ とみなした場合に3の倍数になる記号列で構成される言語を L_1 とする. L_1 を受理する決定性有限オートマトン A_1 の状態遷移図と状態遷移表を下記の例のように書け. なお, 解答するオートマトンの状態の数は高々5つまでとし, 初期状態および最終状態を必ず明記すること.

例

状態遷移表

現状態		次状態 入力記号 0	1
(初期状態, 最終状態)	q_0	q_1	q_2
(最終状態)	q_1	q_3	q_2
(最終状態)	q_2	q_1	q_3
	q_3	q_3	q_3

状態遷移図



(2) 記号列 w を逆順にして得られる記号列を w^{-1} と表し, 言語 L に対し言語 $\{w^{-1} \mid w \in L\}$ を L^{-1} と表すとする. また, 正規表現 E が表現する言語を $L(E)$ と書くこととする. 下記の正規表現 E_2 について, 言語 $(L(E_2))^{-1}$ を表す正規表現の1つを30文字以内で記せ.

$$E_2: \quad (1+\epsilon)(10(10)^*+0)^*$$

なお, 正規表現中の ϵ は空列, $+$ は選択, * は Kleene 閉包を表すこととし, 括弧は結合の優先度を明示的に表し, 連接を表現する演算子は省略されている.

(3) 正規表現 E に対して, $L(E(E)^*) = L((E)^*E)$ であることを説明せよ.

(4) 長さ1以上である入力記号列 $a_1a_2\ldots a_n$ を0以上の2進数 $[a_n\ldots a_2a_1]_2$ とみなした場合に3の倍数となる記号列から構成される言語を L_4 とする. このとき, (1) の L_1 について, $L_1 = L_4$ であることを説明せよ. (ヒント: $(0+1(01^*0)^*1)(0+1(01^*0)^*1)^*$ は L_1 を表現する正規表現の一つである.)

Translation of technical terms

アルファベット	alphabet	初期状態	initial state
言語	language	最終状態	final state
入力記号列	input sequence of symbols	現状態	current state
2進数	binary digit	次状態	next state
倍数	multiple	逆順	reversed sequence
記号列	sequence of symbols	正規表現	regular expression
受理する	accept	空列	empty sequence
決定性有限オートマトン	deterministic finite automaton	選択	alternation
状態遷移図	transition diagram	閉包	closure
状態遷移表	state transition table	連接	concatenation
状態	state	演算子	operator

計算機理論

[1] 次の一階述語論理式 $\phi_1 \sim \phi_4$ について問いに答えよ．ここで，論理結合子の結合の強さは以下の通りとする．

(弱い) 限量子$(\forall, \exists)$ $< \rightarrow < \vee < \wedge$ (強い)

$$\phi_1 : \forall x.A(x) \wedge \neg B(x) \wedge C(x) \rightarrow \exists y.D(x,y) \wedge E(y)$$
$$\phi_2 : \exists x.A(x) \wedge F(x) \wedge \forall y.(D(x,y) \rightarrow F(y))$$
$$\phi_3 : \forall x.F(x) \rightarrow \neg B(x)$$
$$\phi_4 : \forall x.F(x) \rightarrow C(x)$$

(1) ϕ_1, ϕ_2 を同値な冠頭連言標準形に変換せよ．

(2) ψ_1, ψ_2 をそれぞれ ϕ_1, ϕ_2 を冠頭連言標準形に変換した論理式とする．ψ_1, ψ_2 を1引数のスコーレム関数 f とスコーレム定数 a を用いて，充足不能性において同値な全称限量子のみをもつスコーレム標準形に変換せよ．

(3) ψ_1, ψ_2 を同値な節集合に変換せよ．(節はリテラルの集合として記述すること．)

(4) ϕ_3, ϕ_4 を同値な節集合に変換せよ．(節はリテラルの集合として記述すること．)

(5) 論理式 H が，論理式の集合 S の論理的帰結であることを導く手法として反駁がある．反駁について簡単に説明せよ．

(6) $H = \exists x.E(x) \wedge F(x)$ とするとき，導出原理を用いて反駁によって H が $\phi_1 \sim \phi_4$ の論理的帰結であることを示す導出木を示せ．

Translation of technical terms

一階述語論理	first order logic	スコーレム定数	Skolem constant
論理結合子	logical connective	充足不能性	unsatisfiability
限量子	quantifier	節	clause
同値な	equivalent	節集合	set of clauses
冠頭連言標準形	prenex conjunctive normal form	論理的帰結	logical consequence
全称限量子	universal quantifier	反駁	refutation
スコーレム標準形	Skolem normal form	導出原理	resolution principle
スコーレム関数	Skolem function	導出木	resolution tree

ハードウェア

[1] $B = \{0,1\}$を定義域とする論理変数a, b, cに対し，$abc \in B^3$は3ビットの値を表わすものとする．これに対し関数$L: B^3 \to B^3, R: B^3 \to B^3, U: B^3 \to B$をそれぞれ，$L(abc) = bcc$, $R(abc) = aab, U(abc) = (a \oplus b \oplus c)$と定義する．ここで二項演算子$\oplus$は排他的論理和を表わすものとする．このとき，8状態$\{S_{000}, S_{001}, S_{010}, S_{011}, S_{100}, S_{101}, S_{110}, S_{111}\}$および1ビット入力，1ビット出力を持つ順序機械$M$を考え，その状態遷移と出力を以下の通り定義する．

- 状態S_{abc}に対して入力0が与えられたとき，状態$S_{L(abc)}$に遷移し，
  論理値$x = (U(abc) \oplus U(L(abc)))$を出力する
- 状態S_{abc}に対して入力1が与えられたとき，状態$S_{R(abc)}$に遷移し，
  論理値$x = (U(abc) \oplus U(R(abc)))$を出力する

このとき以下の問いに答えよ．

(1) 順序機械Mが状態S_{010}にあるとする．このとき入力系列1,0,1,1,0が，この順で与えられた場合の各入力後の状態と出力をそれぞれ求めよ．

(2) 順序機械Mの状態遷移および出力を，右表のような書式で記述せよ．

(3) 順序機械Mの状態数を最小化し，最小化後の順序機械M^*の状態遷移図を書け．なお，最小化の過程ではM^*の最小性について言及すること．また，状態遷移図については，M^*の各状態とMの各状態との対応関係がわかるように記述すること．

	入力 0		入力 1	
	次状態	出力	次状態	出力
S_{000}				
S_{001}				
S_{010}				
S_{011}				
S_{100}				
S_{101}				
S_{110}				
S_{111}				

Translation of technical terms

定義域	domain	順序機械	sequential machine
論理変数	logical variable	状態遷移	state transition
ビット	bit	論理値	logical value
二項演算子	binary operation	入力系列	input sequence
排他的論理和	exclusive OR	状態遷移図	state transition diagram

[2]マイクロプロセッサのキャッシュと仮想記憶に関して以下の問いに答えよ．

(1)次の用語についてそれぞれ 100 文字以内（英語の場合は 50word 以内）で説明せよ．

(a)キャッシュの連想度

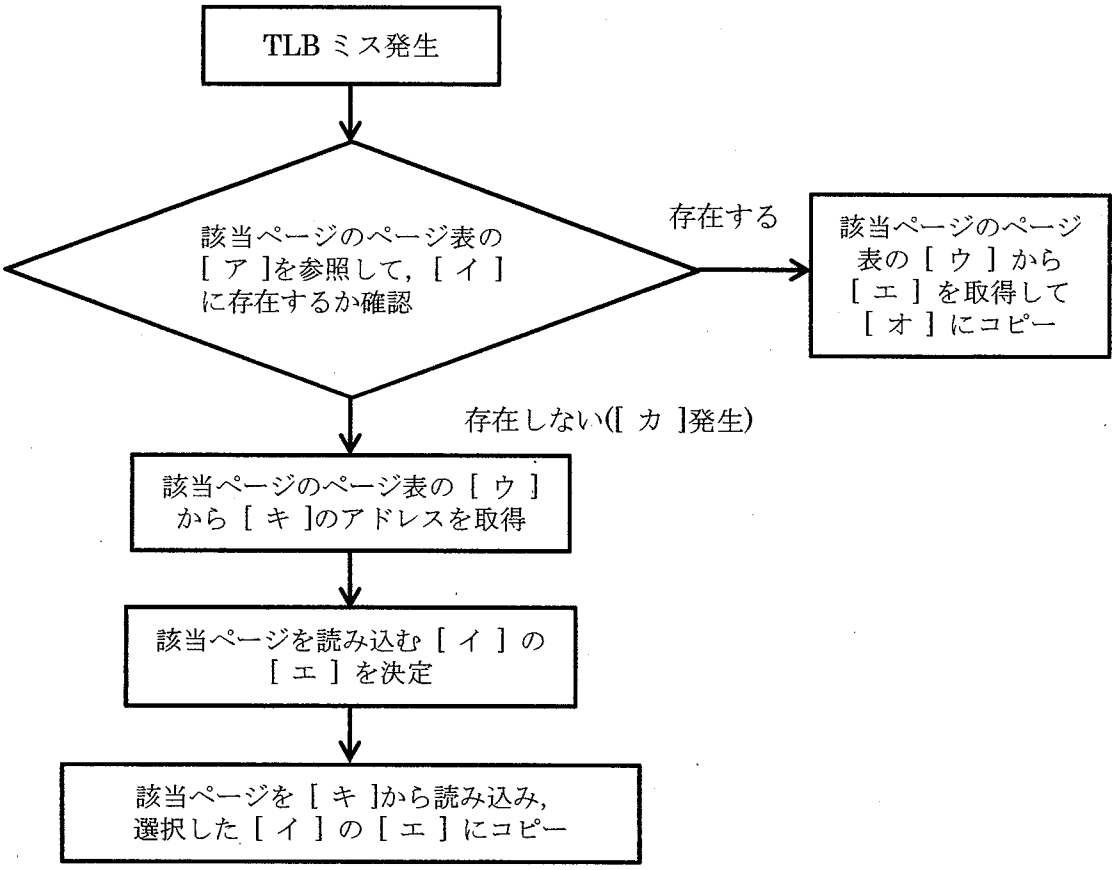
(b)仮想記憶におけるアドレス変換

(c)TLB(translation lookaside buffer)

(2) 1次記憶と 2次記憶を持ち，TLB ミスの処理をオペレーティングシステムで管理するプロセッサにおいて，以下の TLB ミス発生時の処理のフローチャートについて，[ア]から[キ]の空欄を下記の選択肢から選び埋めよ．なお，ページ表の各エントリは，有効ビットとエントリ情報から構成されている．対応するページが 1 次記憶にある場合は有効ビットがオンに，1 次記憶になく 2 次記憶にある場合はオフとなる．エントリ情報には，有効ビットがオンの場合は 1 次記憶の物理ページ番号が，有効ビットがオフの場合は 2 次記憶のアドレスが格納される．また，TLB ミス発生時には，1 次記憶の物理ページには空きがあるとする．

選択肢

1 次記憶・2 次記憶・TLB・有効ビット・エントリ情報・物理ページ番号・ページフォルト・アドレス・アドレス変換・物理アドレス・仮想アドレス



(3)以下に示すようにキャッシュアーキテクチャが異なる 3 種類のプロセッサが存在する．全てのプロセッサにおいて，キャッシュのサイズは 2048 バイト，ブロックサイズは 16 バイトである．なお，キャッシュはデータにのみ有効である．アソシアティブ方式における置き換え対象ブロックの選択方式は LRU法である．

No	ブロック配置方式	ライト方式
1	ダイレクトマップ	ライトスルー
2	2 ウェイセットアソシアティブ	ライトスルー
3	2 ウェイセットアソシアティブ	ライトバック

それぞれのプロセッサにおいて，以下の C 言語プログラムの関数 func() を実行した場合の，int 型の配列 array のアクセスに関して，プロセッサから 1 次記憶へのリード回数とライト回数をそれぞれ答えよ．なお，int 型の変数のサイズは 4 バイトである．プロセッサと 1 次記憶は 32 ビットバスで接続されている．プログラム開始時にはキャッシュの内容は空とする．配列 array は，16 バイト境界(下位 4 ビットが 0 のアドレス)から配置されている．register 宣言されたローカル変数は全てレジスタに割り付けられており，キャッシュに影響を及ぼさないものとする．

```
 1 : int array[2048];
 2 : void func(void){
 3 :         register int i, sum1, sum2, sum3, p_sum1, p_sum2, p_sum3;
 4 :         sum1 = sum2 = sum3 = p_sum1 = p_sum2 = p_sum3 = 0;
 5 :         for(i=0; i<16; i++){sum1 += array[i];}
 6 :         for(i=0; i<16; i++){array[i] = i;}
 7 :         for(i=0; i<16; i++){sum2 += array[i+512];}
 8 :         for(i=0; i<16; i++){array[i+512] = i+16;}
 9 :         for(i=0; i<16; i++){p_sum1 += array[i]*2;}
10 :         for(i=0; i<16; i++){sum3 += array[i+1024];}
11 :         for(i=0; i<16; i++){p_sum2 += array[i+512]*2;}
12 :         for(i=0; i<16; i++){array[i+1024] = i+32;}
13 :         for(i=0; i<16; i++){p_sum3 += array[i+1024]*2;}
14 : }
```

Translation of technical terms

キャッシュ	cache	物理ページ番号	physical page number
仮想記憶	virtual memory	ページフォルト	page fault
連想度	associativity	仮想アドレス	virtual address
アドレス変換	address translation	物理アドレス	physical address
1 次記憶	primary memory	LRU 法	LRU policy
2 次記憶	secondary memory	ダイレクトマップ	direct mapped
TLB ミス	TLB miss	2 ウェイセットアソシアティブ	2-way set associative
オペレーティングシステム	operating system	ライトスルー	write through
ページ	page	ライトバック	write back
ページ表	page table	ローカル変数	local variable

ソフトウェア

［1］メモリ領域の動的な管理方法に関する以下の問いに答えよ．

（1）C言語においては，動的なメモリ領域を確保する時にはmalloc関数を，確保したメモリ領域を解放する時にはfree関数を呼び出すことで，動的なメモリ管理を実現している．この管理方法には，メモリリークと呼ばれる問題と，ダングリングポインタと呼ばれる問題がある．メモリリークとダングリングポインタに関して，それぞれ100文字以内（英語の場合は50 word以内）で説明せよ．

これらの問題を解決するために，ガベージコレクションの機能を持ったプログラミング言語が存在する．ガベージコレクションの典型的な実現方法であるマーク・アンド・スイープでは，グローバル変数およびその時点で有効なローカル変数から，直接または間接に参照できるすべてのメモリ領域に印をつけた後，印のつかなかったメモリ領域を破棄し，そのメモリ領域を回収する．

（2）C言語にガベージコレクションを導入しようとしても，言語の使用方法に制限を加えない限りは，使用していないにもかかわらず回収できないメモリ領域が生じる．この理由を説明せよ．

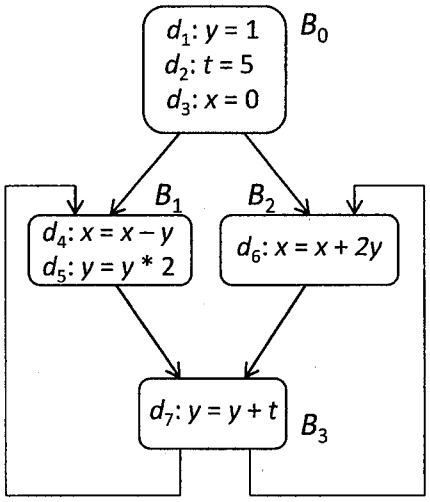
（3）マーク・アンド・スイープを適用しただけでは，メモリの断片化の問題を防ぐことができない．メモリの断片化の問題について説明せよ．また，それを解決するための手法を1つ挙げて説明せよ．

Translation of technical terms

メモリ領域	memory area	プログラミング言語	programming language
動的な	dynamic	マーク・アンド・スイープ	mark-and-sweep
C言語	C language	グローバル変数	global variable
確保する	allocate	ローカル変数	local variable
関数	function	印	mark
解放する	release	破棄	sweep
メモリリーク	memory leak	回収する	collect
ダングリングポインタ	dangling pointer	断片化	fragmentation
ガベージコレクション	garbage collection		

［2］フローグラフは，プログラムの制御の流れを表す有向グラフ $G = (V, E, B_0)$ である．ここで V は頂点の有限集合，E は有向辺の有限集合，$B_0 \in V$ は初期頂点である．各頂点 $B \in V$ はブロックともいい，1つ以上の代入文の列である．代入文は，$d : i = \alpha$ の形式であり，d は固有のラベル，左辺 i はプログラム変数（以降，単に変数とよぶ），右辺 α は変数，定数，演算子からなる算術式とする．変数，定数を含め，算術式のデータ型はすべて整数型とする．代入文 $d : i = \alpha$ をその固有のラベルを使って，単に代入文 d とよぶ．

G の実行は初期頂点 B_0 から開始される．有向辺は実行順を表す．例えば，図のフローグラフには有向辺 $B_0 \to B_1, B_0 \to B_2$ があるが，これは，B_0 の次に B_1 か B_2 のどちらかが実行されることを表す．



変数 i に対し，i の定義文とは，左辺が i である代入文をいう．また i の参照文とは，右辺に i が現れる代入文をいう．例えば，代入文 $i = i + j$ は，i の参照文，j の参照文，i の定義文である．ある変数 i の定義文 d から i の別の定義文を通過することなく，フローグラフのある場所 l に至るパスがあるとき，d は l に到達可能であるという．ここで場所とは，代入文の直前か直後をいう．

ブロック B に対し，

$$
\begin{aligned}
GEN[B] &= \{d \mid d \text{ は } B \text{ に含まれる定義文（のラベル）であり，} d \text{ は } B \text{ の末尾に到達可能である}\} \\
KILL[B] &= \{d \mid d \text{ は } B \text{ 以外のブロックに含まれるある変数 } i \text{ の定義文であり，} B \text{ に } i \text{ の別の定義文が含まれる}\}
\end{aligned}
$$

とする．図のフローグラフにおいて，例えば，

$$GEN[B_0] = \{d_1, d_2, d_3\}, \quad KILL[B_0] = \{d_4, d_5, d_6, d_7\}$$

である．また，

$$IN[B] = \bigcup_{(B' \to B) \in E} OUT[B']$$

と定義する．ここで，$OUT[B']$ は，ブロック B' の末尾に到達可能な定義文全部の集合を表す．例えば，

$$OUT[B_0] = \{d_1, d_2, d_3\}$$

である．

(1) 図のフローグラフにおいて，$GEN[B_1], KILL[B_1], GEN[B_2], KILL[B_2], GEN[B_3], KILL[B_3]$ を求めよ．

(2) $GEN[B], KILL[B], IN[B]$ と集合演算（和集合，積集合，補集合，差集合）を用いて $OUT[B]$ を表せ．

(3) 以下のアルゴリズム A によって，フローグラフ $G = (V, E, B_0)$ のすべてのブロック $B \in V$ について，$OUT[B]$ を求めたい．

アルゴリズム A

```
for each B in V { IN[B] = 空集合 ; OUT[B] = GEN[B] ; }
changed = true ;
while (changed) {
    changed = false ;
    for each B in V {
        NEWIN = ⋃_{(B'→B)∈E} OUT[B'] ;
        if ( [ ア ] ) { changed = true ;
            IN[B] = NEWIN ; OUT[B] = (問 (2) の答) ; }
    }
}
```

(ア) アルゴリズム A 中の空欄 [ア] にあてはまる式を書け．

(イ) 図のフローグラフに対してアルゴリズム A を実行し，while 文を 1 回実行するごとに，各 $IN[B], OUT[B]$ がどのように変化するかを以下のような表で示せ．

	B_0	B_1	B_2	B_3
IN				
OUT				
IN				
OUT				
$\cdots$				

(ウ) アルゴリズム A の各実行文は 1 回あたり定数時間で実行できると仮定する．A の while 文の繰り返し回数を m 回とすると，A の時間計算量は $O(m|V|)$ であることを 50 字程度（英語の場合，25 word 程度）で説明せよ．

(エ) 入力として閉路(へいろ)をもたないフローグラフのみが与えられると仮定する．アルゴルズム A を改良して時間計算量を $O(|V|)$ とするためにはどのように $OUT[B]$ の計算順を変更すればよいかを，150 字程度（英語の場合，75 word 程度）で説明せよ．（ヒント：前処理として深(ふか)さ優先探索(ゆうせんたんさく)を行う．）

(4) あるブロックの中に変数 i の参照文があり，そこに到達可能な変数 i の定義文はただ一つ（d とする）で，d の代入文右辺は定数であるとする．このとき，可能な最適化(さいてきか)について 50 字程度（英語の場合，25 word 程度）で答えよ．

Translation of technical terms

制御の流れ	control flow	データ型	data type
有向グラフ	directed graph	整数型	integer type
頂点	vertex	定義文	definition
有限集合	finite set	参照文	use
有向辺	directed edge	到達可能	reachable
初期頂点	initial vertex	和集合	union
代入文	assignment statement	積集合	intersection
列	sequence	補集合	complement
左辺	left-hand side	差集合	set difference
プログラム変数	program variable	時間計算量	time complexity
右辺	right-hand side	閉路	cycle
定数	constant	深さ優先探索	depth-first search
演算子	operation	最適化	optimization
算術式	arithmetic expression		