専門科目（午前）　25　大修

情報工学

時間　9：30 ～ 12：30

集積システム専攻・計算工学専攻

注　意　事　項

1. 各解答用紙に受験番号を記入せよ．
2. 次の 7 題の中から 4 題を選択して解答せよ．5 題以上解答した場合はすべて無効とする．
3. 解答は 1 題ごとに別々の解答用紙に，問題番号を明記した上で記入せよ．必要であれば，解答用紙の裏面に記入してよいが，解答用紙の表面にその旨を明記すること．
4. 1 枚の解答用紙に 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 枚以上の解答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. 有限次元実ベクトル空間の零ベクトルを $\boldsymbol{o}$ とする．ベクトルの非空な集合 $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ は，$a_1\boldsymbol{v}_1 + a_2\boldsymbol{v}_2 + \cdots + a_n\boldsymbol{v}_n = \boldsymbol{o}$ となるような少なくとも1つは非零である実数 $a_1, a_2, \ldots, a_n$ が存在するとき，線形従属であるという．線形従属ではないベクトルの非空な集合は線形独立であるという．空集合は線形独立であるとする．

V を任意の有限次元実ベクトル空間とすると，V の任意の線形独立なベクトルの集合は有限集合である．線形独立なベクトルの集合 $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\} \subseteq V$ は，任意のベクトル $\boldsymbol{u} \in V - \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ に対して $\{\boldsymbol{u}, \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ が線形従属であるとき，V の極大線形独立なベクトルの集合であるという．ここで，$S - T$ は集合 S と T の差集合を表す．また，有限集合 S の要素数を $|S|$ で表す．

1) 命題『V の線形独立なベクトルの集合の任意の部分集合はまた線形独立なベクトルの集合である.』を証明せよ．

2) 命題『$I, J \subseteq V$ が線形独立なベクトルの集合であり，$|I| = |J| + 1$ であるならば $J \cup \{\boldsymbol{u}\}$ が線形独立であるようなベクトル $\boldsymbol{u} \in I - J$ が存在する.』の証明を下に示す．この証明に関する次の問に答えよ．

   a) 補題1を証明せよ．

   b) 補題2を証明せよ．

命題の証明

J が空集合であるときに命題が成立することは明らかである．J が非空の場合にもこの命題が成立することを背理法を用いて証明する．そこで，線形独立なベクトルの非空な集合 $I = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{n+1}\}$ と $J = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ が存在して，任意の $\boldsymbol{u}_j \in I - J$ に対して $J \cup \{\boldsymbol{u}_j\}$ が線形従属であると仮定する．このとき次の補題1を証明することができる．

補題1 『任意の $j, 1 \leq j \leq n+1,$ に対して $\boldsymbol{u}_j = c_{1j}\boldsymbol{v}_1 + c_{2j}\boldsymbol{v}_2 + \cdots + c_{nj}\boldsymbol{v}_n$ となる実数 $c_{ij}, 1 \leq i \leq n,$ が存在する.』

さて，任意の $j, 1 \leq j \leq n+1,$ に対して $\boldsymbol{c}_j = (c_{1j}, c_{2j}, \ldots, c_{nj})$ と定義するとベクトルの集合 $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{n+1}\}$ は線形従属であることを証明できるので，$d_1\boldsymbol{c}_1 + d_2\boldsymbol{c}_2 + \cdots + d_{n+1}\boldsymbol{c}_{n+1} = \boldsymbol{o}$ となる少なくとも1つは非零である実数 $d_1, d_2, \ldots, d_{n+1}$ が存在する．

以上のことを用いて次の補題2を証明することができる．

補題2 『$I = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{n+1}\}$ は線形従属である.』

しかしながら，補題2は，I は線形独立であるという最初の仮定に矛盾するので，命題が証明された．

3) X と Y が V の極大線形独立なベクトルの集合であるとき，$|X| = |Y|$ であることを問1)と2)の命題を用いて証明せよ．

2. x_1 と x_2 を実数値をとる確率変数とし，互いに独立であるとする．また，整数 n $(n = 1, 2)$ に対して，確率密度関数 $p_n(x_n)$ を

$$p_n(x_n) = \frac{1}{\sqrt{2\pi v_n}} \exp\left[-\frac{(x_n - m_n)^2}{2v_n}\right] \tag{2.1}$$

とする．ただし，m_n と v_n は実数の定数であり，$v_n > 0$ とする．以下の問に答えよ．

1) 式 (2.2) で定める $\mu_{1,2}$ を計算せよ．

$$\mu_{1,2} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x_1 - m_1)(x_2 - m_2)p_1(x_1)p_2(x_2)dx_1dx_2 \tag{2.2}$$

2) 式 (2.3) で定める $\mu_{1,1}$ を計算せよ．

$$\mu_{1,1} = \int_{-\infty}^{\infty} (x_1 - m_1)^2 p_1(x_1)dx_1 \tag{2.3}$$

3) 式 (2.4) で定める確率変数 y の期待値 m と分散 v を計算せよ．ただし，a_1 と a_2 は同時に 0 とならない実数の定数である．

$$y = a_1x_1 + a_2x_2 \tag{2.4}$$

4) 式 (2.1) の $p_n(x_n)$ のフーリエ変換 $\int_{-\infty}^{\infty} p_n(x_n)\exp(-j\omega x_n)dx_n$ を計算せよ．ただし，以下の式 (2.5) を用いてもよい．なお，j は $j^2 = -1$ を満足する虚数単位，ω と b は任意の実数，v' は任意の正の実数である．

$$\frac{1}{\sqrt{2\pi v'}}\int_{-\infty}^{\infty} \exp\left[-\frac{(x - jb)^2}{2v'}\right] dx = 1 \tag{2.5}$$

5) 式 (2.4) の y の確率密度関数を $q(y)$ とする．$q(y)$ のフーリエ変換を計算せよ．ただし，式 (2.6) を用いてもよい．

$$\int_{-\infty}^{\infty} q(y)\exp(-j\omega y)dy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_1(x_1)p_2(x_2)\exp[-j\omega(a_1x_1 + a_2x_2)]dx_1dx_2 \tag{2.6}$$

6) $q(y)$ を求めよ．ただし，4) と 5) の結果を用いてもよい．

3. 記号 a, b, c がある．それらの記号各々 0 個以上からなる任意の列を w とする．そして，そのような w すべての集合を W とする．このとき，以下の問に答えよ．

1) $\{ww^R | w \in W\}$ は正規言語でないことを証明せよ．ここで，w^R は列 w の記号を逆順に並べた列であり，ww^R は w と w^R の連接である．

2) W の任意の部分集合 U について，$\{aw | w \in U\}$ が正規言語ならば U は正規言語であることを証明せよ．

3) $\{ww | w \in W\}$ は文脈自由言語でないことを証明せよ．

4) ある列 x について，それを構成する記号を任意の順序で並べかえた列の集合を x に対する置換集合と呼ぶ．そして，列の集合 L において，その要素すべてに対する置換集合の和集合を L^p とする．例えば，列 ba に対する置換集合は $\{ab, ba\}$ であり，$L = \{ba, abc\}$ のとき $L^p = \{ab, ba, abc, acb, bac, bca, cab, cba\}$ である．
   W のある部分集合 V が正規言語であっても V^p が文脈自由言語になるとは限らないことを，反例とともに示せ．

4. 線形受動回路に関する以下の問に答えよ.

1) 図 4.1 に示すインダクタンス L のコイルと抵抗値 R の抵抗との直列接続からなる 1 ポートの線形受動回路を考える.

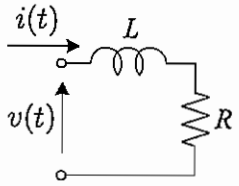

図 4.1

a) 図 4.1 の電流 $i(t)$ と電圧 $v(t)$ との関係を表す微分方程式を求めよ.

b) $t < 0$ において, $v(t) = 0$, $i(t) = 0$ であるとき, $v(t)$, $i(t)$ のラプラス変換 $V(s)$, $I(s)$ で定まる関数

$$H(s) = I(s)/V(s)$$

とその極 ($H(s)$ が無限大となる s) を求めよ.

c) $H(s)$ の逆ラプラス変換 $h(t)$ を求めよ.

d) 図 4.1 の回路では任意の t に対して

$$\int_{-\infty}^{t} v(t')\, i(t')\, dt' \geqq 0 \tag{4.1}$$

が成立していることを, a) の微分方程式を用いて示せ. ただし, $\lim_{t \to -\infty} i(t) = 0$ とする.

2) 図 4.2 に示す一般の 1 ポートの線形受動回路の任意の電圧 $v(t)$ とそれに対応して流れる電流 $i(t)$ に対しても,

$$\int_{-\infty}^{t} v(t')\, i(t')\, dt' \geqq 0 \tag{4.2}$$

が任意の t において成立することが知られている.

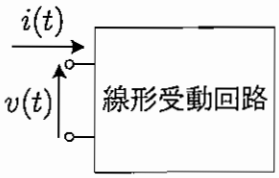

図 4.2

(次ページにつづく)

(前ページのつづき)

この性質から，$v(t) = 0\ (t < 0)$ を満たす電圧 $v(t)$ に対する電流 $i(t)$ は，

$$i(t) = 0\ (t < 0) \tag{4.3}$$

となることを示したい．以下の問に答えよ．

a) 1ポートに印加する2通りの電圧を $v_1(t)$, $v_2(t)$ とし，それぞれに対応する電流を $i_1(t)$, $i_2(t)$ とする．電圧 $v_1(t)$, $v_2(t)$ と実数 α とからなる1ポートの印加電圧 $v_1(t) + \alpha v_2(t)$ に対応する電流は，$i_1(t) + \alpha i_2(t)$ となる理由を簡潔に述べよ．

b) 任意の実数 α に対して，電圧 $v_1(t) + \alpha v_2(t)$ とそれに対応する電流 $i_1(t) + \alpha i_2(t)$ に関して式 (4.2) が成り立つためには，

$$\left[\int_{-\infty}^{t} \left(v_1(t')\, i_2(t') + v_2(t')\, i_1(t')\right) dt'\right]^2 \leqq 4 \int_{-\infty}^{t} v_1(t')\, i_1(t')\, dt' \int_{-\infty}^{t} v_2(t')\, i_2(t')\, dt' \tag{4.4}$$

の不等式が成り立つことが必要かつ十分であることを示せ．

c) $v_1(t) = 0\ (t < 0)$ とすると，$t < 0$ において式 (4.4) より

$$\int_{-\infty}^{t} v_2(t')\, i_1(t')\, dt' = 0 \tag{4.5}$$

が導かれることを示せ．

d) 任意の電圧 $v_2(t)$ に対して，式 (4.5) が成り立つためには，電流 $i_1(t)$ は $t < 0$ において

$$i_1(t) = 0 \tag{4.6}$$

とならなければならないことを示せ．

5. 定常無記憶情報源 S，2元対称通信路 C_0, C_1，および写像 f_0, f_1, g からなる，図 5.1 に示す通信システムについて考える．

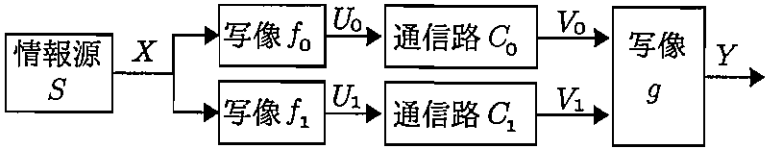

図 5.1 通信システム

情報源 S はアルファベット $A = \{\alpha, \beta, \gamma\}$ の記号を出力するとし，S の出力を確率変数 X で表す．X の確率分布を以下のとおりとする．

$$P(X=\alpha) = P(X=\gamma) = q, \quad P(X=\beta) = 1 - 2q$$

ただし，$0 < q < 1/2$ とする．写像 $f_0 : A \to \{0,1\}$ および写像 $f_1 : A \to \{0,1\}$ を以下のとおり定義する．

$$f_0(x) = \begin{cases} 0 & (x \in \{\alpha, \beta\}) \\ 1 & (x = \gamma) \end{cases}, \quad f_1(x) = \begin{cases} 0 & (x \in \{\beta, \gamma\}) \\ 1 & (x = \alpha) \end{cases}$$

写像 f_0 および f_1 の出力の値をそれぞれ確率変数 U_0 および U_1 で表す．2元対称通信路 C_0 および C_1 の誤り率は 1/3 であり，入力アルファベットおよび出力アルファベットはともに $\{0,1\}$ である．通信路 C_0 および C_1 の出力をそれぞれ確率変数 V_0 および V_1 で表す．

写像 $g : \{0,1\}^2 \to \{a, b, c, d\}$ を以下のとおり定義する．

$$g(0,0) = a, \quad g(0,1) = b, \quad g(1,0) = c, \quad g(1,1) = d$$

ただし，$g(v_0, v_1) = y$ は，確率変数 V_0 および V_1 の値がそれぞれ v_0 および v_1 のとき，写像 g の値が y であることを示す．写像 g の出力の値を確率変数 Y で表す．

以下の問に答えよ．ただし，通信路容量およびエントロピーの単位はビットとし，$\log_2 3$ の値は 1.585 と近似せよ．

1) 通信路 C_0 の通信路容量を小数点以下第2位まで求めよ．

2) 入力および出力がそれぞれ確率変数 X および Y で与えられる通信路の通信路行列を求めよ．ただし，通信路行列は以下のとおり定義し，各要素の値は既約分数で表すこと．

$$\begin{bmatrix} P(Y=a|X=\alpha) & P(Y=b|X=\alpha) & P(Y=c|X=\alpha) & P(Y=d|X=\alpha) \\ P(Y=a|X=\beta) & P(Y=b|X=\beta) & P(Y=c|X=\beta) & P(Y=d|X=\beta) \\ P(Y=a|X=\gamma) & P(Y=b|X=\gamma) & P(Y=c|X=\gamma) & P(Y=d|X=\gamma) \end{bmatrix}$$

3) 条件付きエントロピー $H(Y|X)$ を小数点以下第2位まで求めよ．

4) 確率 $P(Y=a), P(Y=b), P(Y=c), P(Y=d)$ をそれぞれ q を用いて表せ．

5) 相互情報量 $I(X;Y)$ を最大とする q の値を小数点以下第2位まで求めよ．ただし，$I(X;Y)$ を q の関数として表すと，この関数は $0 < q < 1/2$ において上に凸であることを用いてよい．また，$\sqrt{65}$ の値は 8.062 と近似せよ．

6. 以下の問に答えよ．

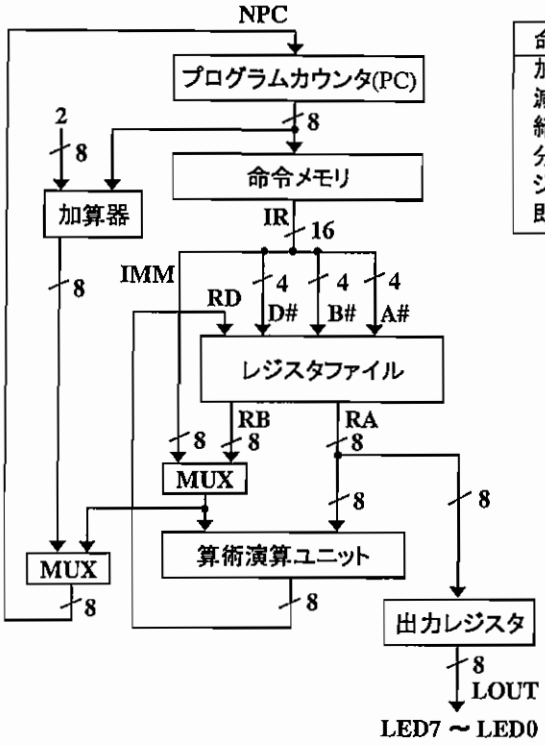

図6.1

命令	ニーモニック形式	レジスタ転送式
加算	ADD RD, RB, RA	R[D#] ← R[B#] + R[A#]
減算	SUB RD, RB, RA	R[D#] ← R[B#] - R[A#]
結果出力	OUT RA	出力レジスタ ← R[A#]
分岐	B RB, RA	if(R[A#]!=0) PC ← R[B#]
ジャンプ	J IMM	PC ← IMM
即値設定	SET RD, IMM	R[D#] ← IMM

図6.2

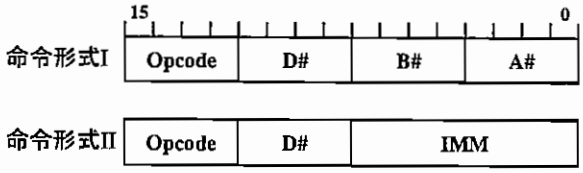

図6.3

1) 図 6.1 に示す 8 ビットプロセッサを用いて，8 個の LED(発光ダイオード) を制御する．このプロセッサの主な構成要素は，16 ビット幅の命令 128 個を格納できる命令メモリ，8 ビット幅のレジスタ 16 個 (R0～R15) を含むレジスタファイル，算術演算ユニットである．プログラムカウンタ，出力レジスタ，レジスタファイルに含まれる全てのレジスタはクロックの立ち上がりのタイミングで更新され，それらの初期値を 0 とする．

図 6.2 に命令仕様を示す．ニーモニック形式では，RD, RB, RA はそれぞれデスティネーションレジスタ，ソースレジスタ B, ソースレジスタ A を表し，R0～R15 の任意のレジスタを指定できる．レジスタ転送式では，D#はデスティネーションレジスタの番号，B#はソースレジスタ B の番号，A#はソースレジスタ A の番号を表す．レジスタ転送式における R[n] はレジスタ番号 n で指定されるレジスタが保持するデータを表す．左矢印記号←はデータ転送である．IMM は命令に含まれる 8 ビット即値である．即値とレジスタファイルに格納される値は符号無し整数とする．

結果出力命令は RA の値を出力レジスタに書き込み，書き込まれた値の各ビットが LED7～LED0 として出力される．ジャンプ命令は即値 IMM をプログラムカウンタに転送する．分岐命令は，RA の値が 0 でない場合に RB の値をプログラムカウンタに転送し，そうでない場合にプログラムカウンタの値に 2 を加算する．ジャンプと分岐を除く命令では，プログラムカウンタの値に 2 を加算する．

図 6.3 に命令形式を示す．加算，減算，結果出力，分岐の各命令は命令形式 I，ジャンプ，即値設定は命令形式 II を採用する．

（次ページにつづく）

（前ページのつづき）

	LED7	LED6	LED5	LED4	LED3	LED2	LED1	LED0
Clock 3	○	○	○	●	○	○	○	●
Clock 6	○	○	●	○	○	○	●	○
Clock 9	○	●	○	○	○	●	○	○
Clock 12	●	○	○	○	●	○	○	○
Clock 15	○	○	○	●	○	○	○	●
Clock 18	○	○	●	○	○	○	●	○
Clock 21	○	●	○	○	○	●	○	○
Clock 24	●	○	○	○	●	○	○	○

図6.4

```
00: SET R10, 11
02: OUT R10
04: ADD R10, R10, R10
06: J 08
08: OUT R10
0a: [      (ア)      ]
0c: J 0e
0e: OUT R10
10: [      (イ)      ]
12: J 14
14: OUT R10
16: J 00
```

図6.5

	A#	B#	D#	RA	RB	RD	NPC	LOUT
Clock 1	–	–	a	–	–	11	02	00
Clock 2	a	–	–	11	–	–	04	00
Clock 3	a	a	a	11	11	22	06	11

図6.6

(a) 図 6.4 のパターンで LED の点滅を繰り返すプログラムを考える．出力 LED0～LED7 が 8 個の LED を制御する．出力レジスタの LSB が LED0 に対応する．

各出力のビットが 1 の時に LED が点灯し，これを記号●を用いて表す．0 の時に LED が消灯し，これを記号○を用いて表す．図 6.5 は，図 6.4 の点滅を実現するプログラムである．行頭の数字はアドレスである．即値は 16 進数で記述している．空欄 (ア),(イ) を埋めよ．

(b) 図 6.5 のプログラムを実行する時，4 クロック目 (アドレス 06 のジャンプ命令が実行される) から 12 クロック目までの 9 命令の，図 6.1 における A#, B#, D#, RA, RB, RD, NPC, LOUT の値を 16 進数で示せ．NPC はプログラムカウンタの入力, LOUT は出力レジスタの出力である．図 6.6 に 1 から 3 クロック目の値を示す．同様の形式にて回答すること．NPC と LOUT を除き，命令実行において意味を持たない箇所には記号 – を記入すること．

（次ページにつづく）

(前ページのつづき)

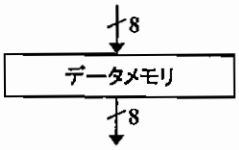

図6.7

命令	ニーモニック形式	レジスタ転送式
加算	ADD RD, RB, RA	R[D#] ← R[B#] + R[A#]
減算	SUB RD, RB, RA	R[D#] ← R[B#] - R[A#]
結果出力	OUT RA	出力レジスタ ← R[A#]
分岐	B RB, RA	if(R[A#]!=0) PC ← R[B#]
ジャンプ	J IMM	PC ← IMM
即値設定	SET RD, IMM	R[D#] ← IMM
ロード	LD RD, RB, RA	R[D#] ← M[R[B#] + R[A#]]

図6.8

命令メモリ	データメモリ
00: SET R1, 01	20: 01
02: SET R2, 02	21: 10
04: SET R13, 20	22: 02
06: SET R14, 0c	23: 28
08: SET R10, 08	24: 04
0a: SET R11, 01	25: 44
0c: LD R8, R13, R11	26: 08
0e: OUT R8	27: 44
10: ADD R11, R11, R2	28: 10
12: SUB R10, R10, R1	29: 7c
14: B R14, R10	2a: 20
16: J 08	2b: 44
	2c: 40
	2d: 44
	2e: 80
	2f: 00

図6.9

2) 図 6.7 に示す 8 ビットのデータ 256 個を格納できるデータメモリからレジスタにデータをロードするロード命令を追加する．図 6.8 は，図 6.2 にロード命令を加えた命令仕様である．レジスタ転送式における M[a] はデータメモリのアドレス a に格納されているデータを表す．

(a) 図 6.1 を修正して，この仕様のプロセッサの全体構成を図示せよ．

(b) このプロセッサで図 6.9 のプログラムを実行する．9, 14, 19, 24, 72, 77, 82, 87 クロック目における LED の点灯の様子を図 6.4 と同様の形式で示せ．

7. 以下のような条件 (A), (B) を満たす 2 分木を考える.

(A) 子節点が存在する全ての節点について，その節点の値が子節点の値より大きいか等しい.

(B) 一番深いレベル以外は節点が全て詰まっており，一番深いレベルでは節点が左に詰まっている.

このような木を以下ではヒープ木と呼ぶ．ヒープ木を幅優先で左から走査しながら節点の値を順番に格納した配列を以下ではヒープ配列と呼ぶ．配列の添字は 0 から始まるものとし，n 個の要素からなる配列 b を b[0..n-1] と表記する．図 7.1 は，ヒープ木の例とそれに対応するヒープ配列 c[0..9] である.

1) b[0..n-1] をヒープ配列として，それに対応するヒープ木を T とする.

   a) T の高さを h とするとき，n の最大値を h で表せ．ここで木の高さとは，根節点と一番深いレベルの節点間にある枝の数とする．例えば，図 7.1 のヒープ木は高さ 3 である.

   b) T において節点 p の左の子節点を q, 右の子節点を r とし，p, q, r がそれぞれ b[s], b[t], b[u] に対応するとき，t と u を s で表せ.

   c) b[0..n-1] における n 個の要素が相異なる値を持ち，その最大値と最小値を持つ要素をそれぞれ b[v], b[w] とするとき，v と w それぞれの値もしくは値の範囲を答えよ.

2) ヒープ配列からは最大の要素を効率良く取り出すことができるので，最大の要素を取り出し，それを除いた配列をヒープ配列として構成する処理を繰り返すことによって，要素を大きい順に次々と取り出すことができる．この考えを応用したデータ整列のアルゴリズムはヒープソートと呼ばれる．図 7.2 はヒープソートを実装した C 言語の関数である.

   a) func1 は上記の下線部を実行する関数である．以下の説明を読んで，図 7.2 の空欄 ア, イ, ウ を埋めよ.

      最大要素の取り出しとヒープ配列の構成について，図 7.1 を用いて説明する．まず，c[0] と c[9] の値を交換して最大の要素を c[9] に格納する．これは，図 7.1 のヒープ木において根節点の「12」と葉節点の「3」を交換することに対応する．次に，c[9] を除く c[0..8] をヒープ配列として構成する．この処理は func1(c,0,8) を呼び出すことで実行される．図 7.3 はこの処理を木構造で示しており，(i), (ii), (iii) の各段階において値が「3」に更新された節点を x と呼ぶ．図 7.3 (i) では，x とその子節点以外の親子は条件 (A) を満たしているので，x とその子節点が条件 (A) を満たすかどうか調べて，満たさない場合は x と子節点の値を交換する．ここで，x と値を交換するのは，2 つ存在する子節点のうち値が大きい方だけである．図 7.3 (ii) でも (i) と同様の処理を行う．x とその子節点が条件 (A) を満たすか，図 7.3 (iii) のように x に子節点が存在しなければ処理を終了する.

   b) func1 は，図 7.3 のように x とその子節点以外の親子が条件 (A) を満たすことを前提としている．func2 は，そのような前提が成り立たない場合でも func1 を使用して，与えられた配列をヒープ配列として構成する関数である．func2 の第 1 引数と第 2 引数には，それぞれ配列とその要素数を与える．図 7.2 の空欄 エ, オ, カ を埋めよ.

   c) func3 は，与えられた配列をヒープソートによって昇順に整列する関数である．func3 の第 1 引数と第 2 引数には，それぞれ配列とその要素数を与える．図 7.2 の関数を可能な限り使用して，図 7.2 の空欄 キ, ク, ケ を埋めよ.

（次ページにつづく）

（前ページのつづき）

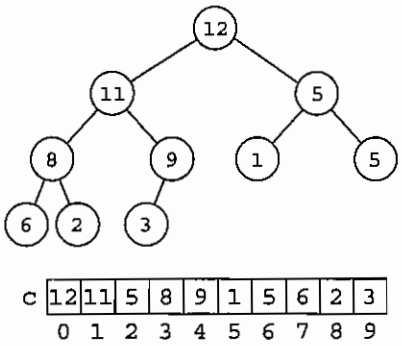

図 7.1: ヒープ木とヒープ配列の例

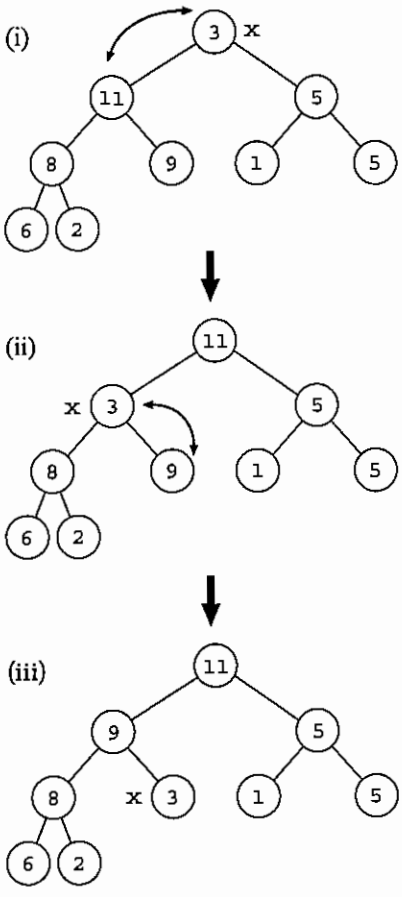

図 7.3: ヒープ木を構成する例

```
void swap(int a[], int i, int j) {
  int k;
  k = a[i]; a[i] = a[j]; a[j] = k;
}

void func1(int a[], int i, int j) {
  int k;
  while ((k = ア ) <= j) {
    if (k < j && イ ) {
      k++;
    }
    if (a[i] >= a[k]) {
      break;
    }
    swap(a, i, k);
    ウ ;
  }
}

void func2(int a[], int m) {
  int i;
  for ( エ ; オ ; i--) {
    func1(a, i, カ );
  }
}

void func3(int a[], int m) {
  キ ;
  while (m > 1) {
    ク ;
    func1(a, ケ , m - 2);
    m--;
  }
}
```

図 7.2: ヒープソートの関数