令 和 5 年 度

名古屋大学大学院情報学研究科
情報システム学専攻
入学試験問題（専門）

令和４年８月８日

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．外国人留学生の志願者は、日本語と日本語以外の１言語間の辞書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．日本語または英語で解答すること。

５．問題冊子、解答用紙６枚、草稿用紙３枚が配布されていることを確認すること。

６．問題は問１～８の８問がある。このうち６問を選択して解答すること。なお、選択した問題番号を解答用紙の指定欄に記入すること。

７．全ての解答用紙の所定の欄に受験番号を必ず記入すること。解答用紙に受験者の氏名を記入してはならない。

８．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すること。

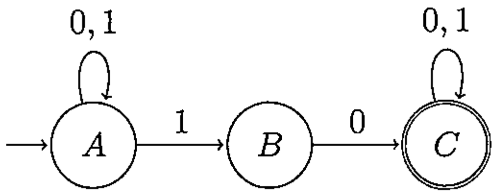
９．解答用紙は試験終了後に６枚とも提出すること。

１０．問題冊子、草稿用紙は試験終了後に持ち帰ること。

問 1

本問では，アルファベットを $\Sigma = \{0, 1\}$ とする．

(1) 次に図示される非決定性有限オートマトン M_1 が認識する言語 $L(M_1)$ について，以下の問いに答えよ．



(a) $w \in L(M_1)$ を満たし，かつ，長さ 3 以下の w をすべて列挙せよ．

(b) $L(M_1) = L(M_2)$ を満たす決定性有限オートマトン M_2 を図示せよ．

(c) $L(M_1) = L(E_1)$ を満たす正規表現 E_1 を示せ．

(2) 言語 $L(\subseteq \Sigma^*)$，および，記号 $a(\in \Sigma)$ に対して，言語 L/a を次のように定義する．

$$L/a = \{w \in \Sigma^* \mid wa \in L\}$$

このとき，以下の問いに答えよ．

(a) 問 (1) の M_1 に対して，$w \in L(M_1)/0$ を満たし，かつ，長さ 2 以下の w をすべて列挙せよ．

(b) 非決定性有限オートマトン M から $L(M)/a$ を認識する非決定性有限オートマトンを構成する方法を示せ．

(c) 正規表現 E が入力されると $L(E)/a$ を表す正規表現を返す関数 ϕ を再帰的に定義せよ．

Translation of technical terms

アルファベット	alphabet
非決定性有限オートマトン	non-deterministic finite automaton
認識する	recognize
言語	language
列挙する	enumerate
決定性有限オートマトン	deterministic finite automaton
正規表現	regular expression

問 2

論理設計に関する以下の全ての問いに答えよ．複数の解答が存在する場合でも一つの解答のみを示せ。

(1) 論理関数 $f_1(x_4, x_3, x_2, x_1) = \Sigma(2, 3, 4, 6, 11, 12, 14)$ について次の (a)，(b) に答えよ．なお，$f_1(x_4, x_3, x_2, x_1) = \Sigma(2, 3, 4, 6, 11, 12, 14)$ で表わされる f_1 とは，入力変数 x_4, x_3, x_2, x_1 に与えられる 4 桁の 2 進数の値が 10 進数の 2,3,4,6,11,12,14 のいずれかに対応するときには出力が 1 となり，10 進数の 0,1,5,7,8,9,10,13,15 のいずれかに対応するときには出力が 0 となる論理関数である．例えば，$(x_4, x_3, x_2, x_1) = (1, 1, 0, 0)$ のときには，f_1 の入力値は $(1100)_2$ である．この 2 進数 $(1100)_2$ の値は 10 進数の 12 に対応するため f_1 の出力は 1 となる．同様に，$(x_4, x_3, x_2, x_1) = (1, 1, 0, 1)$ のときには入力値が 10 進数の 13 に対応するため f_1 の出力は 0 となる．

(a) 論理関数 f_1 に対するカルノー図を示せ．

(b) 論理関数 f_1 の最小の積和形論理式を示せ．最小の積和形論理式とは，積項数が最小の積和形論理式のうち，リテラル数の総和が最小のものである．

(2) 上記 (1) の論理関数 f_1 と論理関数 $f_2(x_4, x_3, x_2, x_1) = \Sigma(1, 2, 6, 9, 12, 13) + \Sigma_{dc}(4, 8, 14)$ を実現する 4 入力 (x_4, x_3, x_2, x_1) 2 出力 (f_1, f_2) の組合せ回路を作りたい．次の (a)，(b) に答えよ．なお，$f_2(x_4, x_3, x_2, x_1) = \Sigma(1, 2, 6, 9, 12, 13) + \Sigma_{dc}(4, 8, 14)$ とは，入力変数 x_4, x_3, x_2, x_1 に与えられる 4 桁の 2 進数の値が 10 進数の 1,2,6,9,12,13 のいずれかに対応するときには出力が 1 となり，4,8,14 のいずれかに対応するときには出力がドントケア (*) となり，0,3,5,7,10,11,15 のいずれかに対応するときには出力が 0 となる論理関数である．

(a) 論理関数 f_2 の最小の積和形論理式を示せ．

(b) 二つの論理関数 (f_1, f_2) の積和形論理式を最小化し，それぞれ積和形論理式で示せ．二つの積和形論理式の最小化とは，二つの積和形論理式の共通項を一つの項と数え，全体としての積項数を最小にし，さらにその中でリテラル数の総和が最小になるように二つの論理関数をそれぞれ積和形論理式にすることである．

Translation of technical terms

論理設計	logic design
論理関数	logic function
2 進数	binary number
10 進数	decimal number
カルノー図	Karnaugh map
積和形	sum-of-products form
論理式	logical formula
積項	product term
リテラル	literal
組合せ回路	combinational circuit
ドントケア	don't care

問3

(1) A, B を有限集合とし，それらの要素数を，$m = |A|, n = |B|$ とおく．以下のような写像の総数をそれぞれ求めよ．(a),(b) については m, n を用いて表し，(c) については整数で表せ．(c) については導出過程も簡単に記すこと．

- (a) A から B への写像
- (b) A から B への単射
- (c) $m = 5, n = 3$ のときの A から B への全射

(2) $N = \{1, 2, \ldots\}$ を自然数全部の集合，R を実数全部の集合とする．

- (a) $N \times N$ から N への全単射 f を一つ与えよ．具体的に，任意の $i, j \in N$ に対して，$f(i, j)$ を，i, j を用いた式で表せ．また，f が全単射であることを説明せよ．
- (b) $[0,1] = \{r \in R \mid 0 \leq r \leq 1\}$, $[0,1) = \{r \in R \mid 0 \leq r < 1\}$ とおく．$[0,1]$ から $[0,1)$ への全単射が存在するかどうかを答えよ．存在する場合，そのうちの一つを示せ．存在しない場合，そのことを証明せよ．

Translation of technical terms

有限集合	finite set	単射	injection
要素数	cardinality	全射	surjection
写像	mapping	自然数	natural number
総数	total number	実数	real number
整数	integer	全単射	bijection

問 4

古典論理のシーケント計算 LK の推論規則を図 1 に示す．図 1 において，A, B は任意の論理式を表す．$\Gamma, \Pi, \Delta, \Sigma$ は有限個 (0 個でもよい) の論理式をコンマで区切って並べた列を表す．LK のシーケントは $\Gamma \vdash \Delta$ の形をした表現である．直観主義論理のシーケント計算 LJ は，LK のシーケントの右辺に現れる論理式の数を高々一つに制限したものである．つまり，LJ のシーケントは $\Gamma \vdash B$ または $\Gamma \vdash$ の形に限る．括弧を省略したときの論理結合子の結合の強さは，強いものから順に否定 ($\neg$)，論理積 ($\wedge$)，論理和 ($\vee$)，含意 ($\supset$) とする．以下の問いに答えよ．

証明図の書き方に関する注意

シーケント $\vdash (\neg q \supset \neg p) \supset (p \supset q)$ を根とする証明図の例を示す．

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{$(init)$}
\UnaryInfC{$q \vdash q$}
\RightLabel{(R$\neg$)}
\UnaryInfC{$\vdash q, \neg q$}
\AxiomC{}
\RightLabel{$(init)$}
\UnaryInfC{$p \vdash p$}
\RightLabel{(L$\neg$)}
\UnaryInfC{$\neg p, p \vdash$}
\RightLabel{(L $\supset$)}
\BinaryInfC{$\neg q \supset \neg p, p \vdash q$}
\RightLabel{(L$exch$)}
\UnaryInfC{$p, \neg q \supset \neg p \vdash q$}
\RightLabel{(R $\supset$)}
\UnaryInfC{$\neg q \supset \neg p \vdash p \supset q$}
\RightLabel{(R $\supset$)}
\UnaryInfC{$\vdash (\neg q \supset \neg p) \supset (p \supset q)$}
\end{prooftree}
$$

証明図を書く場合は，上の例のように推論規則を適用する過程をすべて記述すること．R $\supset$ などの推論規則の名前も必ず書くこと．

(1) LJ の cut 規則を示せ．

(2) LK の推論規則のうち，LJ のシーケントには適用できない推論規則の名前をすべて挙げよ．

(3) p, q, r を命題変数とする．以下の 2 つのシーケントについて，それを根とする LJ の証明図を示せ．LJ の証明図が存在しない場合は，LK の証明図を示せ．

(a) $\vdash (p \wedge (p \supset q)) \wedge (p \wedge q \supset r) \supset r$

(b) $\neg(p \wedge q) \vdash \neg p \vee \neg q$

(4) 図 1 の導入規則のうち，名前が R で始まるものを右導入規則とよぶ．LJ の証明図が次の条件を満たすとき，LJ のユニフォーム証明図であるという．

- cut 規則を含まず，右辺の論理式が論理結合子をもつシーケントは必ず右導入規則の結論 (下側のシーケント) になっている．

シーケント S で，S を根とする LJ の証明図は存在するが，S を根とする LJ のユニフォーム証明図は存在しないような S の例を一つ示せ．また，その結果となる理由を示せ．

init 規則と *cut* 規則

$$\frac{}{A \vdash A}\,(init) \qquad \frac{\Gamma \vdash \Delta, A \quad A, \Pi \vdash \Sigma}{\Gamma, \Pi \vdash \Delta, \Sigma}\,(cut)$$

構造規則(こうぞうきそく)

$$\frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\,(\mathrm{L}weak) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\,(\mathrm{R}weak)$$

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\,(\mathrm{L}contra) \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\,(\mathrm{R}contra)$$

$$\frac{\Gamma, A, B, \Pi \vdash \Delta}{\Gamma, B, A, \Pi \vdash \Delta}\,(\mathrm{L}exch) \qquad \frac{\Gamma \vdash \Delta, A, B, \Sigma}{\Gamma \vdash \Delta, B, A, \Sigma}\,(\mathrm{R}exch)$$

導入規則

$$\frac{A, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\,(\mathrm{L} \wedge 1) \qquad \frac{B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\,(\mathrm{L} \wedge 2)$$

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}\,(\mathrm{R}\wedge) \qquad \frac{A, \Gamma \vdash \Delta \quad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta}\,(\mathrm{L}\vee)$$

$$\frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \vee B}\,(\mathrm{R} \vee 1) \qquad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \vee B}\,(\mathrm{R} \vee 2)$$

$$\frac{\Gamma \vdash \Delta, A \quad B, \Pi \vdash \Sigma}{A \supset B, \Gamma, \Pi \vdash \Delta, \Sigma}\,(\mathrm{L} \supset) \qquad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \supset B}\,(\mathrm{R} \supset)$$

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta}\,(\mathrm{L}\neg) \qquad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\,(\mathrm{R}\neg)$$

図 1: 古典論理のシーケント計算 *LK* の推論規則

Translation of technical terms

古典論理	classical logic
シーケント計算	sequent calculus
推論規則	inference rule
論理式	formula
シーケント	sequent
表現	expression
直観主義論理	intuitionistic logic
論理結合子	logical connective
否定	negation
論理積	conjunction
論理和	disjunction
含意	implication
証明図	proof diagram
根	root
適用する	apply
命題変数	propositional variable
導入規則	introduction rule
右導入規則	right introduction rule
ユニフォーム証明図	uniform proof diagram
結論	conclusion
構造規則	structural rule

問5

配列を利用してヒープを実現し，優先度付きキューとして使用することを考える．取り扱うデータは整数であると仮定し，データの値そのものをヒープのキーとして使用する．ヒープは，最下層のみに節点の欠落を許す完全２分木として構成し，最下層の節点は左側から詰めて配置する．ヒープの根節点を節点 1，節点 i の左右の子をそれぞれ節点 $2i$，節点 $2i+1$ と呼ぶ．内部節点 i のキーの値は，その子 $2i$ および $2i+1$ のキーの値以上となっている必要があり，これをヒープ条件という．

十分大きなサイズを持つ配列 A の存在を仮定し，ヒープにおける節点 i のデータを配列要素 $A[i]$ に格納する．また，配列 A の中に格納されているデータ数（ヒープの節点数）を変数 $A.size$ に代入して記録する（したがって，ヒープのデータは $A[1], \ldots, A[A.size]$ に格納されている）．疑似コード 1 の Heapify は節点 i で局所的に崩れたヒープ条件を修復する操作，疑似コード 2 の Build-Heap は全ての節点でヒープ条件が成り立つようデータを移動する操作である．なお，$\lfloor x \rfloor$ は x 以下の最大整数を表す．

(1) 配列 A の中に，図 1 に示すようなデータが格納されている．この図に対応する配列要素 $A[1], \ldots, A[10]$ の値を示せ．

(2) 問 (1) の配列 A に対し Build-Heap(A) を実行した後のヒープを，図 1 のような２分木として示せ．

上記のように実現したヒープを利用し，優先度付きキューを構成する．優先度付きキューから最大のデータを取り出す Pull 操作，優先度付きキューにデータ（キー）を挿入する Push 操作は，疑似コード 3, 4 のように実現される．

(3) 疑似コード 3 で (X) となっている箇所に記載すべき操作を，1 行の疑似コードとして示せ．複数の疑似コードが考えられる場合は，できるだけ効率の良いものを解答すること．

(4) 図 2 のヒープが配列 A に格納されているとする．このとき，Push(A, 15) を実行した後のヒープを２分木として示せ．

疑似コード 1 : Heapify(A, i)

```
l = 2i
r = 2i + 1
largest = i
if l ≤ A.size and A[l] > A[largest] then
    largest = l
if r ≤ A.size and A[r] > A[largest] then
    largest = r
if largest ≠ i then
    A[i] の値と A[largest] の値を交換
    Heapify(A, largest)
```

疑似コード 2 : Build-Heap(A)

```
for i = ⌊A.size/2⌋ down to 1 do
    Heapify(A, i)
```

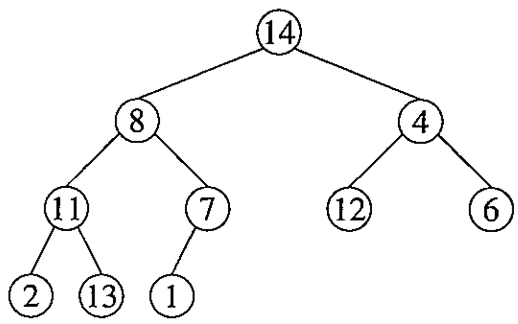


図 1: ヒープ構成前の初期データ

疑似コード 3 : Pull(A)

```
if A.size < 1 then
    error "underflow"
max = A[1]
A[1] = A[A.size]
A.size = A.size − 1
[        (X)        ]
return max
```

疑似コード 4 : Push(A, key)

```
A.size = A.size + 1
A[A.size] = key
i = A.size
p = ⌊i/2⌋
while i > 1 and A[p] < A[i] do
    A[p] の値と A[i] の値を交換
    i = p
    p = ⌊i/2⌋
```

優先度付きキューに格納されているデータ数（ヒープの節点数）を n とする.

(5) Push の最悪時間計算量を n に関するオーダー記法で示せ．また，その計算量となる理由を説明せよ.

(6) Build-Heap の最悪時間計算量を n に関するオーダー記法で示せ．また，その計算量となる理由を説明せよ.

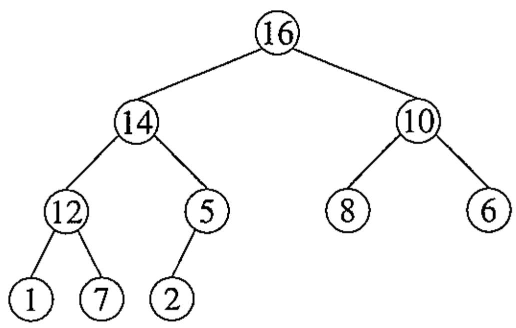


図 2: Push 操作前のヒープ

Translation of technical terms

配列	array	根	root
ヒープ	heap	内部	internal
優先度付きキュー	priority queue	ヒープ条件	heap condition
整数	integer	疑似コード	pseudo-code
キー	key	最悪	worst-case
最下層	lowest level	時間計算量	time complexity
節点	node	オーダー記法	order notation
完全 2 分木	complete binary tree		

問 6

Web ブラウザを使ってファイル送信者からファイル受信者にファイルを渡すインターネットサービス S に関する問い(1), (2)に答えよ.

(1) 図 1 に示す S の UML(Unified Modeling Language) ユースケース図について(a), (b), (c)に答えよ.

(a) 図 1 が表現する内容として正しいものをすべて選べ.

① ファイル送信者はログインした後にファイルをアップロードしなければならない.

② ファイル送信者とファイル受信者はすべてのデータや情報を S を介してやりとりしなければならない.

③ ファイル送信者とファイル受信者は同一人物でもよい.

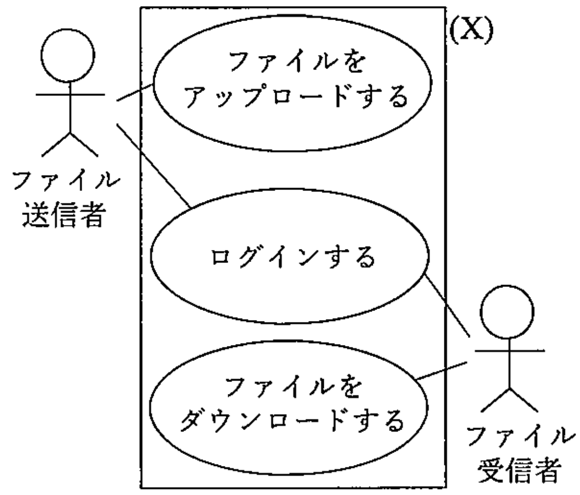


図 1 ユースケース図

(b) 図 1 の(X)の枠線は何を表すか答えよ.

(c) 図 1 の(X)の枠線による設計への効果を 50 字(英文の場合, 30 words) 以内で説明せよ.

(2) 図 2 に示す S の UML シーケンス図に関する下の問いに答えよ. 図 2 中の URL は Uniform Resource Locator の略である. URL には送信するファイルごとに一意の文字列が割り当てられる. なお, ファイル送信者とファイル受信者のアカウント, パスワードは図 2 のやりとりの前に登録してあるものとする. また, Web ブラウザと Web サーバは実装対象外であるため記載していない.

(a) 図 2 中の①, ②のラベルの名称をそれぞれ答えよ.

(b) ユーザ認証クラス, ファイル管理クラス, ファイルクラスをクラス図で書け. クラスは図 2 のやりとりを実現するために必要な属性と操作を持つものとする. 属性名と操作名は内容を適切に表すよう命名してよいが, 図 2 に同一のものが定義されている場合にはその名称を使うこと. クラス図には図 2 から考えられる関連と多重度も書くこと. 複数の解答が考えられる場合にはそのうちの一つを解答すること.

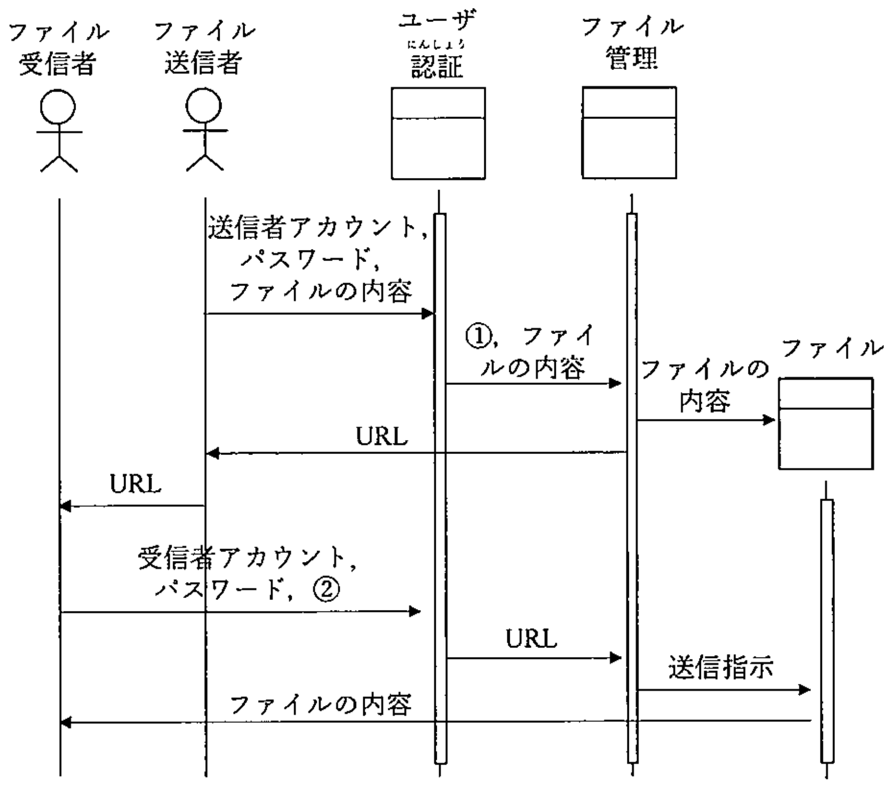


図 2 シーケンス図

Translation of technical terms

ブラウザ	browser	アカウント	account
ファイル	file	パスワード	password
インターネットサービス	Internet service	サーバ	server
ユースケース図	use case diagram	実装	implementation
アップロード	upload	ラベル	label
ダウンロード	download	データ	data
ログイン	login	クラス	class
シーケンス図	sequence diagram	属性	attribute
認証	authentication	操作	operation
一意	unique	関連	association
文字列	string	多重度	multiplicity

問 7

以下の全ての問いに答えよ．ただし，コンパイル時の最適化は行われないものとし，オーバーフローエラーが起こらない範囲での実行のみを考えることとする．

(1) ソースコード 1 に掲げた C 言語のプログラム `binsearch.c` について，以下の全ての問いに答えよ．

(a) `binsearch(x,i,j)` が「昇順にソートされた配列 `a[i],...,a[j]` に対し，値 `x` の存在を二分探索によって判定する」関数になるように，ソースコード 1 の $\boxed{\mathrm{A}}$ ～ $\boxed{\mathrm{C}}$ を埋めよ．

(b) `binsearch_loop` が，`binsearch` と同じ二分探索を再帰呼出しを用いずに記述したものになるように，ソースコード 1 の $\boxed{\mathrm{D}}$ ～ $\boxed{\mathrm{F}}$ を埋めよ．

(2) ソースコード 2 に掲げた C 言語のプログラム `function.c` について，以下の全ての問いに答えよ．

(a) `main()` を実行した時，標準出力に出力される実行結果を書け．

(b) 任意の非負整数 `n` に対して，`f2(n,0,1)` の返り値が，`f1(n)` の返り値と常に同じになるように，ソースコード 2 の $\boxed{\mathrm{G}}$ を埋めよ．

(c) 以下の回数をそれぞれ答えよ．

i. `main()` を実行したとき，`f1` が呼び出される回数．

ii. 137 行目の `f1(7)` を，`f2(7,0,1)` に置き換えて `main()` を実行したとき，`f2` が呼び出される回数．

(d) `f_loop` が，`f2` と同じ結果を返す関数を再帰呼出しを用いずに記述したものになるように，ソースコード 2 の $\boxed{\mathrm{H}}$ ～ $\boxed{\mathrm{K}}$ を埋めよ．

(3) `binsearch` や `f1`，`f2` のような再帰呼出しを含むプログラムよりも，`binsearch_loop` や `f_loop` のように再帰呼出しを用いない方が，多くの場合，実行時間が短くなる．この理由を，50 字以内（英語の場合，30 words 以内）で説明せよ．

Translation of technical terms

コンパイル	compile	二分探索	binary search
最適化	optimization	関数	function
オーバーフローエラー	overflow error	再帰呼出し	recursive call
昇順	ascending order	標準出力	standard output
ソート	sort	非負整数	non-negative integer
配列	array	返り値	return value

ソースコード 1: binsearch.c

```
#include <stdio.h>
#define TRUE 1
#define FALSE 0

int a[10] = { 1, 4, 6, 10, 11, 13, 15, 20, 30, 32};

int binsearch (int x, int i, int j) {
  int k;
  if(i>j)
    return FALSE;
  else {
    k=(i+j)/2;
    if ( [    A    ] )
      return binsearch(x,k+1,j);
    else if ( [    B    ] )
      return binsearch( [    C    ] );
    else
      return TRUE;
  }
}

int binsearch_loop (int x, int i, int j) {
  int k;
  while(1) {
    if(i>j) {
      return FALSE;
    } else {
      k=(i+j)/2;
      if ( [    D    ] )
        i=k+1;
      else if ( [    E    ] )
        j= [    F    ] ;
      else {
        return TRUE;
      }
    }
  }
}

int main (void) {
  if(binsearch(14,0,9)) printf("found\n");
  else printf("not found\n");

  if(binsearch_loop(14,0,9)) printf("found\n");
  else printf("not found\n");

  return 0;
}
```

ソースコード 2: function.c

```c
#include <stdio.h>

int f1 (int x) {
  if (x <= 0)
    return 0;
  else if (x == 1)
    return 1;
  else
    return f1(x-2) + f1(x-1);
}

int f2 (int x, int y, int z) {
  if (x <= 0)
    return y;
  else if (x == 1)
    return z;
  else
    return f2(x-1, z, [   G   ] );
}

int f_loop (int x, int y, int z) {
  int tmp;
  while (1) {
    if (x<=0) {
      return y;
    } else if (x == 1) {
      return [   H   ] ;
    } else {
      tmp = y;
      x = [   I   ] ;
      y = [   J   ] ;
      z = [   K   ] ;
    }
  }
}

int main (void) {
  printf("%d\n", f1(7));

  return 0;
}
```

問 8

下の表に示す 3 つのプロセスを，残り処理時間が短いプロセスから順にスケジューリングしたところ，図 1 に示すようにスケジューリングされた。これに関して，以下の問いに答えよ。ただし，スケジューリングはプリエンプティブに行うこととし，明示されていない限り，プロセス切換えにかかる時間は無視できるものとする。また，平均応答時間（プロセスが到着してから完了するまでの時間の平均値）は，小数点以下を四捨五入して，ミリ秒単位で答えよ。

	到着時刻	処理時間
プロセス 1	0 ミリ秒	800 ミリ秒
プロセス 2	300 ミリ秒	200 ミリ秒
プロセス 3	200 ミリ秒	400 ミリ秒

(1) 図 1 において，プロセスの平均応答時間を求めよ。

(2) 残り処理時間が短い順でスケジューリングする方法は，利点はあるものの，実際にオペレーティングシステムで採用するのは難しい。この方法の利点と，採用するのが難しい理由を述べよ。

(3) 同じプロセスセットを，到着順（FCFS）でスケジューリングした時のスケジュールを図示し，プロセスの平均応答時間を求めよ。

(4) 同じプロセスセットを，タイムスライスを無視できる位まで短くしてラウンドロビンスケジューリングした時の，プロセスの平均応答時間を求めよ。

(5) 同じプロセスセットを，タイムスライスを 100 ミリ秒としてラウンドロビンスケジューリングした時のスケジュール（複数の可能性がある）の 1 つを図示し，プロセスの平均応答時間を求めよ。ただし，プロセス切換えに 1 ミリ秒かかるものとする。

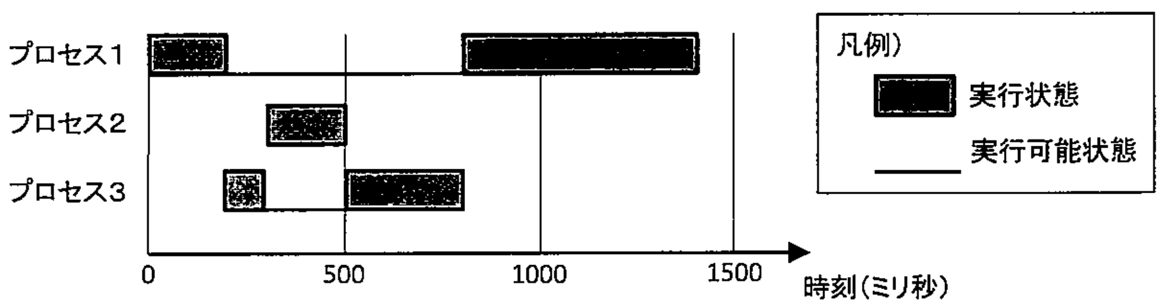


図 1: 残り処理時間が短い順でのスケジュール

Translation of technical terms

プロセス	process
処理時間	processing time
スケジューリング	scheduling
プリエンプティブ	preemptive
プロセス切換え	process switch
平均応答時間	average response time
到着時刻	arrival time
到着順	first come first served
タイムスライス	time slice
ラウンドロビン	round robin