文部科学省後援

情報システム試験

令和６年度前期 情報検定

＜実施　令和6年9月8日（日）＞

プログラミングスキル

（説明時間　10：00～10：10）

（試験時間　10：10～11：40）

・試験問題は試験開始の合図があるまで開かないでください。

・解答用紙（マークシート）への必要事項の記入は，試験開始の合図と同時に行いますので，それまで伏せておいてください。

・試験開始の合図の後，次のページを開いてください。**＜受験上の注意＞**が記載されています。必ず目を通してから解答を始めてください。

・試験問題は，すべてマークシート方式です。正解と思われるものを１つ選び，解答欄の()をＨＢの黒鉛筆でぬりつぶしてください。２つ以上ぬりつぶすと，不正解になります。

・辞書，参考書類の使用および筆記用具の貸し借りは一切禁止です。

・電卓の使用が認められます。ただし，下記の機種については使用が認められません。

＜使用を認めない電卓＞

１．電池式（太陽電池を含む）以外の電卓

２．文字表示領域が複数行ある電卓（計算状態表示の一行は含まない）

３．プログラムを組み込む機能がある電卓

４．電卓が主たる機能ではないもの

＊パソコン（電子メール専用機等を含む），携帯電話，スマートフォン，タブレット，電子手帳，電子メモ，電子辞書，翻訳機能付き電卓，音声応答のある電卓，電卓付き腕時計，時計型ウェアラブル端末等

５．その他試験監督者が不適切と認めるもの

一般財団法人　職業教育・キャリア教育財団

＜受験上の注意＞

１．この試験問題は22ページあります。ページ数を確認してください。
乱丁等がある場合は，手をあげて試験監督者に合図してください。
※問題を読みやすくするために空白ページを設けている場合があります。

２．解答用紙（マークシート）に，受験者氏名・受験番号を記入し，受験番号下欄の数字をぬりつぶしてください。正しく記入されていない場合は，採点されませんので十分注意してください。

３．試験問題についての質問には，一切答えられません。自分で判断して解答してください。

４．試験中の筆記用具の貸し借りは一切禁止します。筆記用具が破損等により使用不能となった場合は，手をあげて試験監督者に合図してください。

５．試験を開始してから30分以内は途中退出できません。30分経過後退出する場合は，もう一度，受験番号・マーク・氏名が記載されているか確認して退出してください。なお，試験終了５分前の合図以降は退出できません。試験問題は各自お持ち帰りください。

６．試験後の合否結果（合否通知），および合格者への「合格証・認定証」はすべて，Ｗｅｂ認証で行います。
①情報検定（J検）Webサイト合否結果検索ページ及びモバイル合否検索サイト上で，ディジタル「合否通知」，ディジタル「合格証・認定証」が交付されます。
②団体宛には合否結果一覧ほか，試験結果資料一式を送付します。
③合否等の結果についての電話・手紙等でのお問い合わせには，一切応じられませんので，ご了承ください。

問題１　次のデータ構造に関する記述を読み，各設問に答えよ。

データ構造の１つにスタックがある。スタックはプログラムからのサブルーチンや関数の呼び出しなど，コンピュータの内部処理に利用されている。

＜設問１＞　次のスタックに関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

［スタック構造］

スタックとは後入れ先出し法(LIFO)によりデータなどを管理するメモリ領域のことである。スタックの操作は次の２つである。

・push(A)　A をスタックに格納する

・pop()　　スタックからデータを取り出す

1000 番地	
1001 番地	
1002 番地	
1003 番地	
1004 番地	10

図１　メモリ領域

ここでスタックを管理する上で重要な役割を果たしているのがスタックポインタ(以下，SP)である。SP はスタックの最上段のデータのアドレスを保持する。

図１のスタックを例にすると,1001 番地から 1004 番地をスタック領域とした場合，現在の SP の値は 1004 である。次に push(10)，push(20)，push(30)，POP()の順に実行すると，取り出される値は［　(1)　］である。この時，SP の値は 1003→1002→1001→［　(2)　］の順に変化する。

(1) の解答群

ア．10　　イ．20　　ウ．30

(2) の解答群

ア．1001　　イ．1002　　ウ．1003　　エ．1004

＜設問２＞　次のスタックの一連の動作に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

図２の状態のスタックがあり，スタック領域は 1001 番地から 1007 番地とする。

1000 番地	
1001 番地	
1002 番地	
1003 番地	
1004 番地	10
1005 番地	60
1006 番地	40
1007 番地	70

図２　スタック

図２の状態のとき，SP は［　(3)　］となる。ここで次の順に命令を実行した。

pop() → pop() → push(80) → push(20) → pop() → push(30)
→ push(90) → pop() → pop() → push(50) → push(100)

すべての命令が完了したあと，1004 番地の内容は［　(4)　］となり，SP の値は［　(5)　］となる。

(3)，(5)の解答群

ア．1000　　イ．1003　　ウ．1004　　エ．1007

(4) の解答群

ア．30　　イ．50　　ウ．90　　エ．100

問題２　次の流れ図の説明を読み，流れ図中の［　　］に入れるべき適切な字句を解答群から選べ。

［流れ図の説明］

エラトステネスのふるいを用いて，100 以下の素数を求める流れ図である。素数とは，1 とその数自体でしか割り切れない 2 以上の整数である。

エラトステネスのふるいでは，２から順に整数の倍数を消去していき，残った数が素数となる。次の手順で求める。

手順１：作業用配列 w を使い，w[0]と w[1]には 0 を，w[2]から w[100]に，2～100 までの整数を格納する。

手順２：w[2]から順に取り出した整数が，0 以外のときは手順３を実行し，0 のときは指標を 1 だけ増やし，指標が 10 以下の間手順 2 を繰り返す。（※）

手順３：取り出した整数の倍数の格納位置に 0 を格納することで配列内の倍数を消去する。

手順４：最後に配列 w に残った 0 以外の数値が素数であり，1 次元配列 s[0]から順に格納する。

（※）100 までの素数を求める場合，100 の平方根である 10 以下の値を検証することで全ての素数を求めることができる。

(1) の解答群

ア．w[k] ← 0　　イ．w[k] ← k　　ウ．w[k+2] ← k

(2) の解答群

ア．j ← w[0]　　イ．j ← w[2]　　ウ．j ← w[k]

(3) の解答群

ア．m ← m － 1　　イ．m ← m ＋ 1　　ウ．m ← m ＋ j

(4) の解答群

ア．w[j] ： 0　　イ．w[k] ： 0　　ウ．w[k+2] ： 0

(5) の解答群

ア．j ← j － 1　　イ．j ← j ＋ 1　　ウ．j ← j ＋ m

［流れ図］

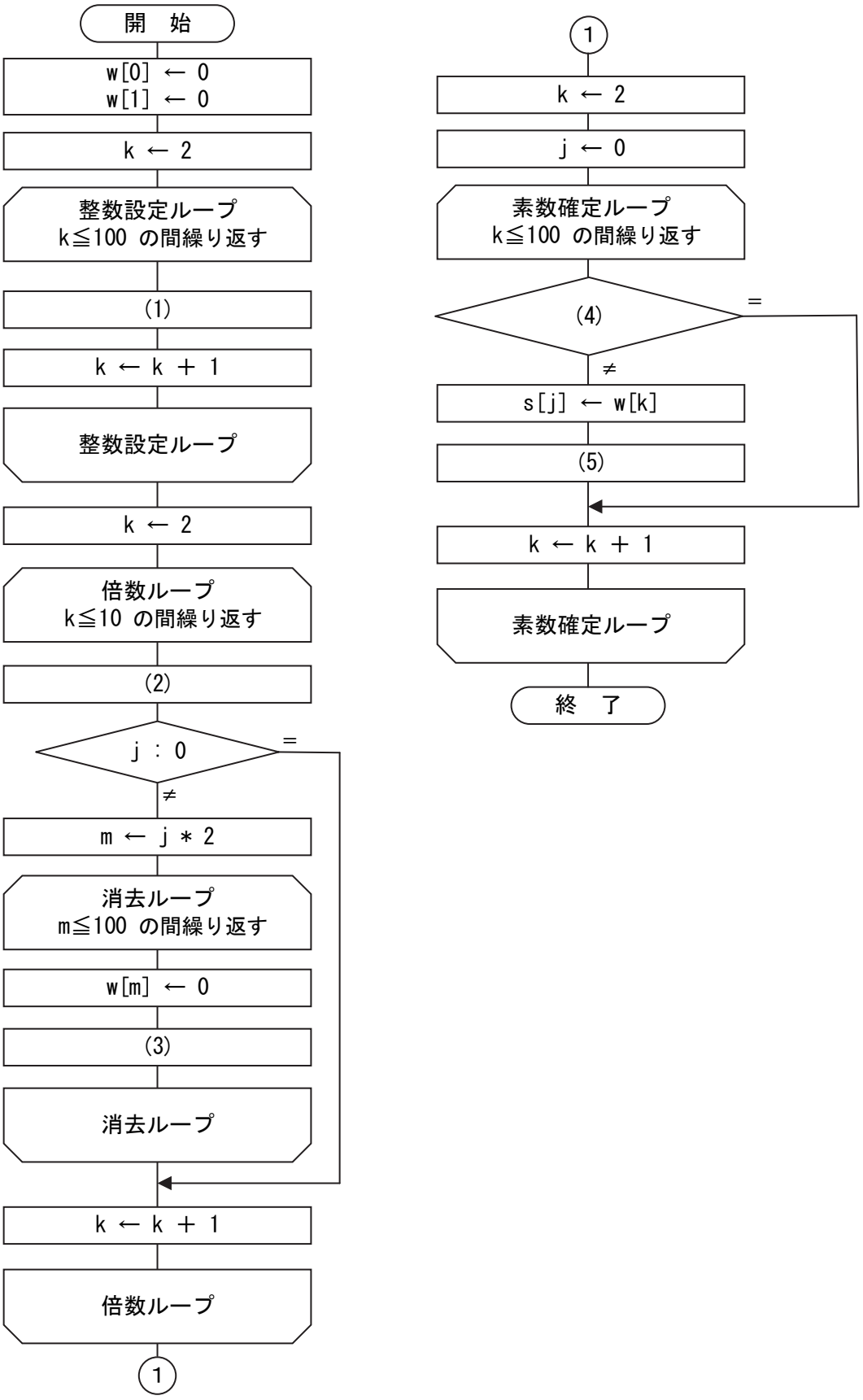

図　エラトステネスのふるいの流れ図

問題３　次の基数変換に関する記述を読み，各設問に答えよ。

［基数変換について］

基数変換とは，ある基数に基づいて記述された数字列を他の基数の数字列に置き換えることである。日常生活では 10 を基数とした 10 進数が使われているが，コンピュータの中では２を基数とした２進数が使われている。また，２進数は桁数が多くなり読みにくくなるケースがあるため，８進数や 16 進数で表示することがある。

この問題では，10 進数の整数値を２進数へ変換する処理を考える。なお，ここで扱う２進数は 16 ビットで，最左端ビットを符号ビットとした２の補数で負数を表すものとする。また，２進数の各桁の値は整数型の配列 bin[i]（i=0,1,…,15）に格納し，bin[15]は 2^0 の位，bin[14]は 2^1 の位，･･･，となる。

位置	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
bin	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

図１　10 進数の 100 を 2 進数に変換して格納した例

流れ図中で使っている mod は剰余を求める演算子であり，この問題で扱う 10 進数の値は-32768 以上 32767 以下の整数値とする。なお，除算結果は小数点以下を切り捨てた整数値になる。

＜設問１＞　次の流れ図の説明を読み，流れ図中の［　　］に入れるべき適切な字句を解答群から選べ。

［流れ図の説明］

変数 dec に設定された 10 進数を２進数に変換して配列 bin に格納する流れ図 dec2bin である。設定された 10 進数が負数の時は符号を反転して正数として２進数を作成し，２進数に変換後符号を戻すために２の補数をとる。dec2bin では，２の補数を求める complement を使用する。complement は，配列 bin に格納された値の 0 と 1 を反転し，2^0 の桁に 1 を加えてから繰上処理を行う。

なお，配列 bin と変数 dec は大域変数として宣言してある。

［流れ図］

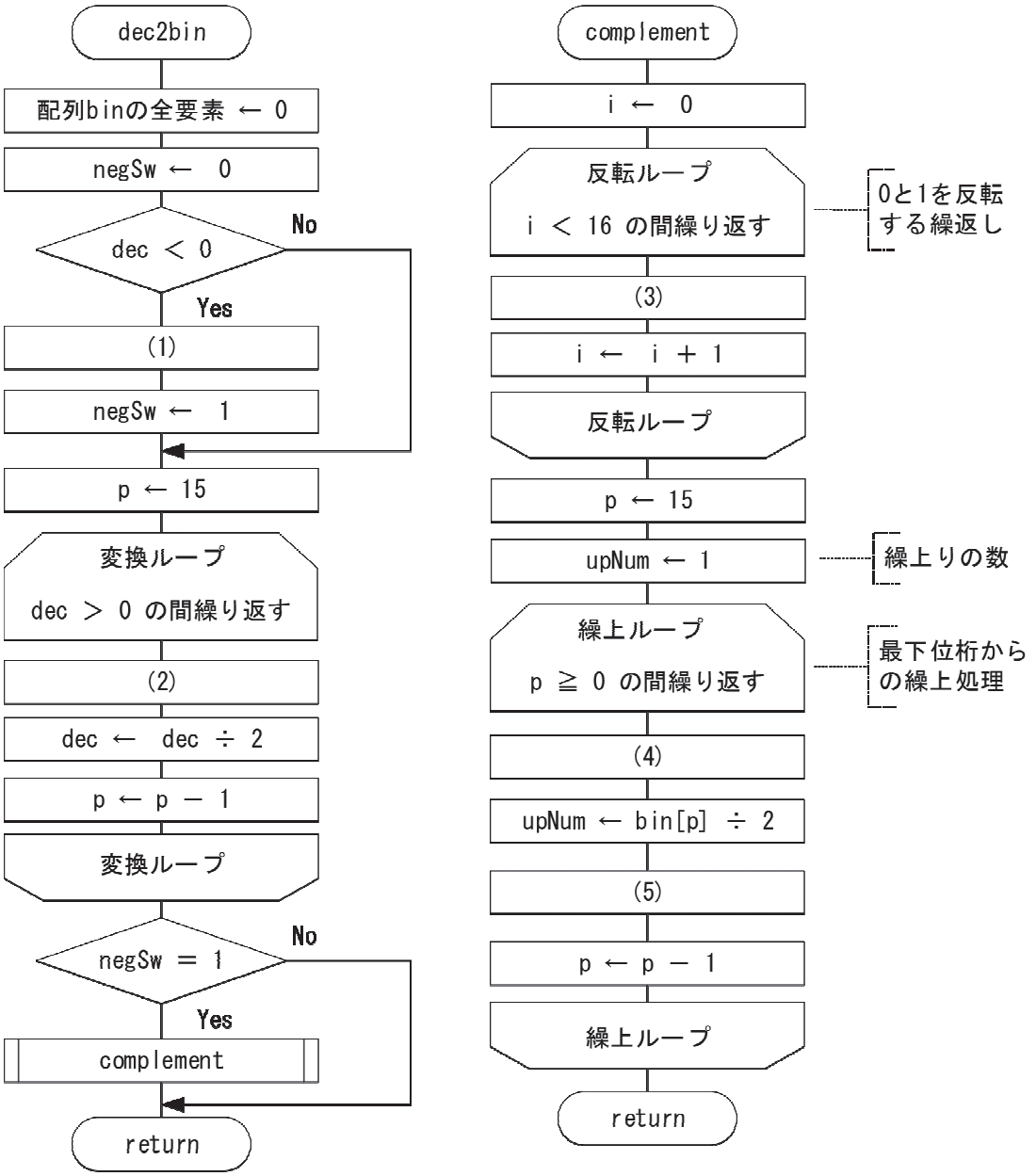

図２　dec2bin と complement の流れ図

(1) の解答群

ア. dec ← 0 − dec　　イ. dec ← 1 − dec

ウ. dec ← dec − 0　　エ. dec ← dec − 1

(2) の解答群

ア. bin[p] ← bin[p] mod 2　　イ. bin[p] ← bin[p] ÷ 2

ウ. bin[p] ← dec mod 2　　エ. bin[p] ← dec ÷ 2

(3) の解答群

ア. bin[i] ← 0 − bin[i]　　イ. bin[i] ← 1 − bin[i]

ウ. bin[i] ← bin[i] − 1　　エ. bin[i] ← bin[i] × (−1)

(4) の解答群

ア. bin[p − 1] ← bin[p] + 1　　イ. bin[p − 1] ← bin[p] + upNum

ウ. bin[p] ← bin[p] + upNum　　エ. bin[p] ← bin[p] + 1

(5) の解答群

ア. bin[p] ← 0 − bin[p]　　イ. bin[p] ← bin[p] − 1

ウ. bin[p] ← bin[p] × 2　　エ. bin[p] ← bin[p] mod 2

＜設問２＞　次の流れ図の説明を読み，流れ図中の［　　］に入れるべき適切な字句を解答群から選べ。

［流れ図の説明］

配列 bin に格納された２進数を表示する流れ図 printBin である。ここで表示する処理では改行を行わないものとする。

printBin は，bin[i](i＝0,1,…,15)を順番に表示するが，見やすくするために４桁分を表示した後に空白文字を表示する。

```
0000 0000 0110 0100
```

図３　printBin の実行例（先頭と末尾に空白文字は無い）

［流れ図］

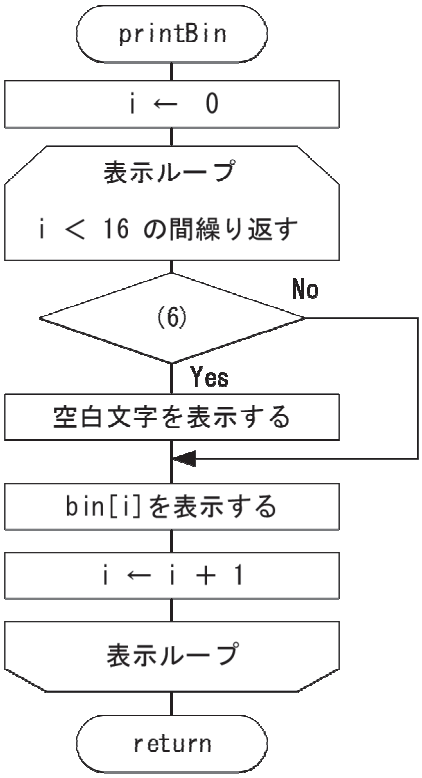

図４　printBin の流れ図

(6) の解答群

ア．i ＝ 4　　イ．i ＞ 0　かつ　i mod 4 ＝ 0

ウ．i ＞ 4　　エ．i ＝ 0　かつ　(i＋1) mod 4 ＝ 0

＜設問３＞　次の流れ図の説明を読み，流れ図中の［　　］に入れるべき適切な字句を解答群から選べ。

［流れ図の説明］

２の補数を求めるために 0 と 1 を反転して 1 を加える方法が良く知られている。

この方法以外に，下位の桁から上位の桁へ向かって最初に"1"の出現する位置を探し，その位置より上位の桁のみ 0 と 1 を反転する方法がある。この方法により，図２の流れ図 complement を変更したものである。

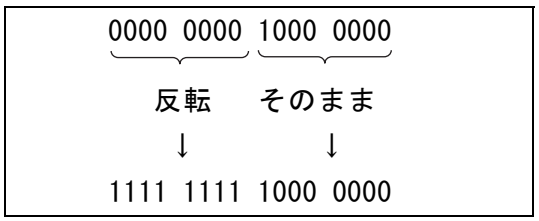

図５　２の補数を求める方法の例

［流れ図］

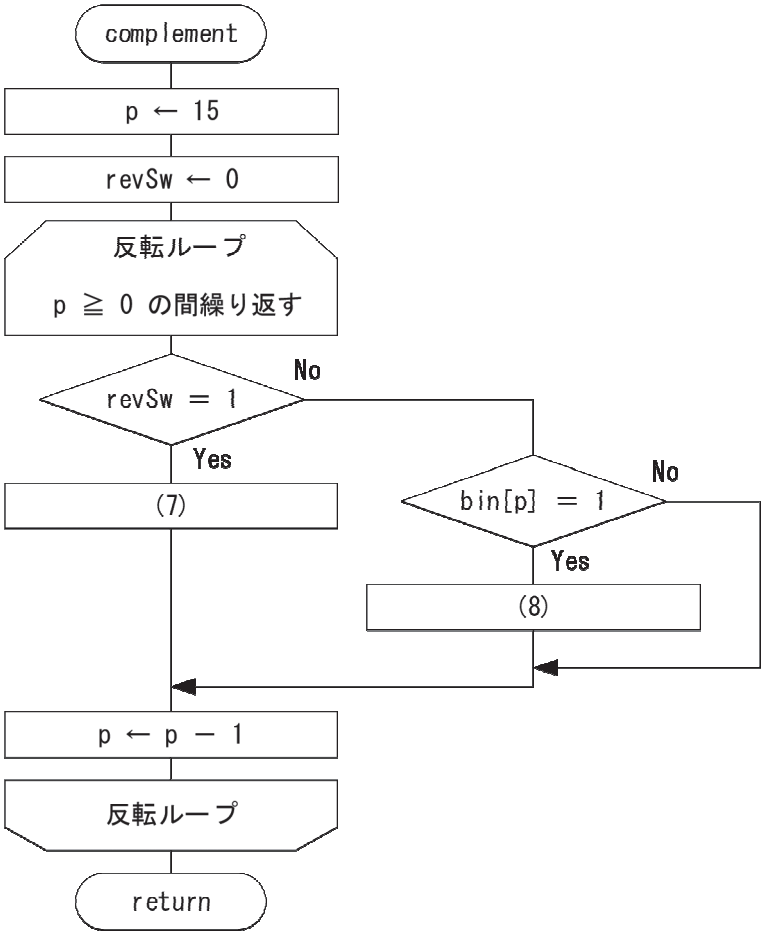

図６　変更した complement の流れ図

(7) の解答群

ア．bin[p] ← 0 − bin[p]　　イ．bin[p] ← 1 − bin[p]

ウ．bin[p] ← bin[p] − 1　　エ．bin[p] ← bin[p] × (−1)

(8) の解答群

ア．p ← p − 1　　イ．p ← p ＋ 1

ウ．revSw ← 0　　エ．revSw ← 1

＜設問４＞　次の流れ図の説明を読み，流れ図中の［　　］に入れるべき適切な字句を解答群から選べ。

［流れ図の説明］

変数 dec に格納された 10 進数を 16 進数に変換して文字型の配列 hex に格納する流れ図 dec2hex である。dec2hex では，図２の流れ図 dec2bin により得られる２進数４桁ずつを 16 進数に変換する。

なお，流れ図中で使用する配列 num は次の文字が格納されている。
また，配列 hex は大域変数として宣言されている。

位置	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
num	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

図７　配列 num に格納されている文字

［流れ図］

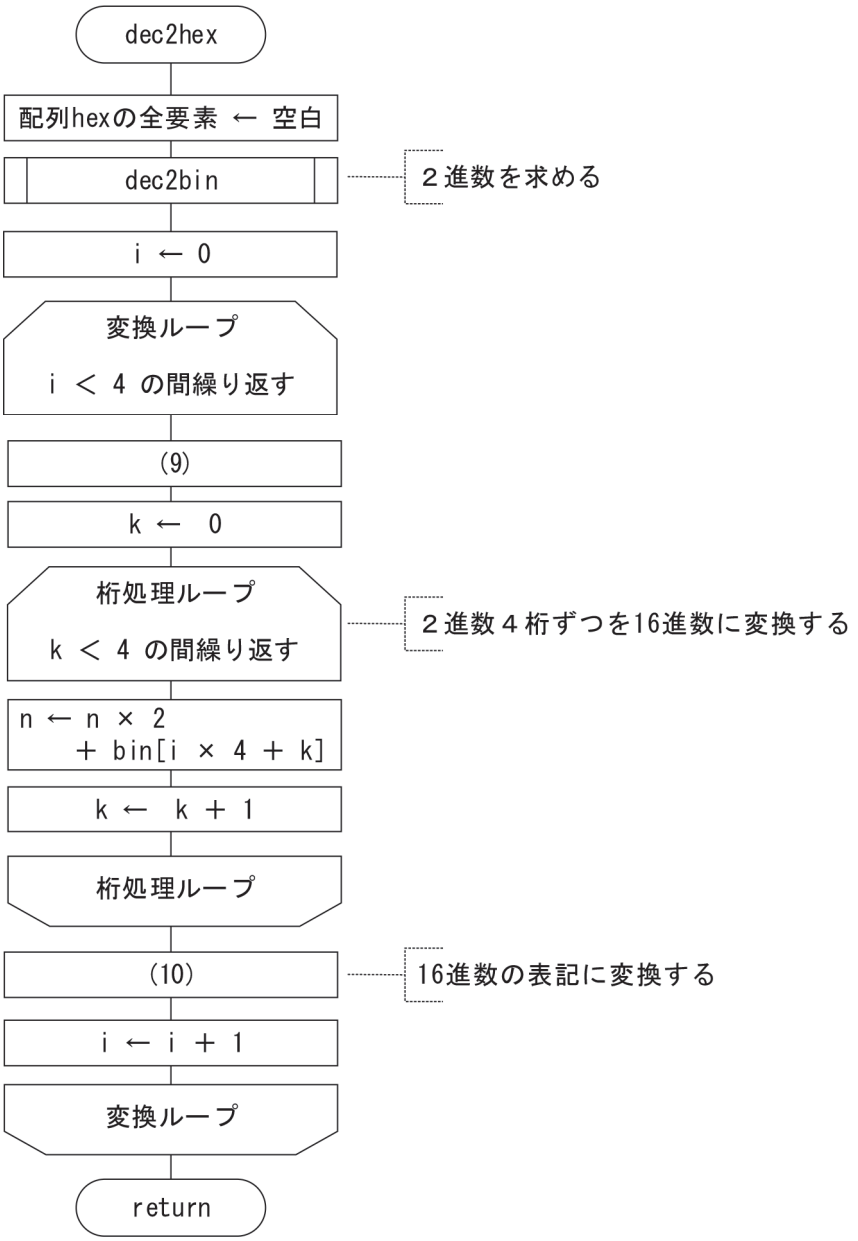

図８　dec2hexの流れ図

(9) の解答群

ア．n ← －1　　イ．n ← 0

ウ．n ← 1　　エ．n ← 16

(10) の解答群

ア. hex[i] ← num[n]　　イ. hex[i] ← num[n ＋ 1]

ウ. hex[n] ← num[i]　　エ. hex[n] ← num[i ＋ 1]

問題４，問題５で使用する擬似言語の仕様は以下のとおりである。

［擬似言語の記述形式の説明］

記述形式	説明
○*手続名又は関数名*	手続又は関数を宣言する。
型名：*変数名*	変数を宣言する。
/* *注釈* */ // *注釈*	注釈を記述する。
変数名 ← *式*	変数に*式*の値を代入する。
手続名又は関数名(*引数*, …)	手続又は関数を呼び出し，*引数*を受け渡す。

if（*条件式1*）
　処理1
elseif（*条件式2*）
　処理2
elseif（*条件式n*）
　処理n
else
　処理n＋1
endif
	選択処理を示す。
	*条件式*を上から評価し，最初に真になった*条件式*に対応する*処理*を実行する。以降の*条件式*は評価せず，対応する*処理*も実行しない。どの*条件式*も真にならないときは，*処理n＋1*を実行する。
	各*処理*は，0 以上の文の集まりである。
	elseif と*処理*の組みは，複数記述することがあり，省略することもある。
	else と*処理n＋1* の組みは一つだけ記述し，省略することもある。

while（*条件式*） 　　*処理* endwhile	前判定繰返し処理を示す。 *条件式*が真の間，*処理*を繰返し実行する。 *処理*は，0 以上の文の集まりである。
do 　*処理* while（*条件式*）	後判定繰返し処理を示す。 *処理*を実行し，*条件式*が真の間，*処理*を繰返し実行する。 *処理*は，0 以上の文の集まりである。
for（*制御記述*） 　*処理* endfor	繰返し処理を示す。 *制御記述*の内容に基づいて，*処理*を繰返し実行する。 *処理*は，0 以上の文の集まりである。

〔演算子と優先順位〕

演算子の種類		演算子	優先度
式		( ) .	高
単項演算子		not ＋ －	↑
二項演算子	乗除	mod × ÷	↑
	加減	＋ －	↑
	関係	≠ ≦ ≧ ＜ ＝ ＞	↑
	論理積	and	↓
	論理和	or	低

注記 演算子 . は，メンバ変数又はメソッドのアクセスを表す。
演算子 mod は，剰余算を表す。

〔論理型の定数〕

true，false

〔配列〕

配列の要素は，“[”と“]”の間にアクセス対象要素の要素番号を指定することでアクセスする。なお，二次元配列の要素番号は，行番号，列番号の順に“,”で区切って指定する。

“{”は配列の内容の始まりを，“}”は配列の内容の終わりを表す。ただし，二次元配列において，内側の“{”と“}”に囲まれた部分は，1 行分の内容を表す。

〔未定義，未定義の値〕

変数に値が格納されていない状態を，“未定義”という。変数に“未定義の値”を代入すると，その変数は未定義になる。

問題４　次の整列に関するプログラムの説明を読み，各設問に答えよ。

ある一定の規則に従ってデータを並び替える処理を整列(ソート)と呼ぶ。データの値を小さなものから大きなものへ順番に並べることを昇順，その反対に大きなものから小さなものへと並べることを降順という。

ソートには選択法や挿入法など，様々なアルゴリズムがある。

＜設問１＞　次の選択法を用いた Sort_Selection に関する説明を読み，プログラム中の［　　］に入れるべき適切な字句を解答群から選べ。なお，配列の要素番号は 1 から始まるものとする。

［選択法の説明］

１次元配列 DAT[1]～DAT[N]に N 個のデータが格納されている。このデータを，選択法により昇順に整列する。選択法は，先頭要素から順番に一つずつ要素を決定していくアルゴリズムである。N＝5 とした例を示す。

	1	2	3	4	5
DAT	30	40	80	10	20

図１　配列 DAT の初期状態

手順 1：全要素の中から最小値を選択し，先頭要素 DAT[1]と交換する。

	1	2	3	4	5
DAT	10	40	80	30	20

図２　配列の１番目に最小値を求めた状態

手順 2：残りの要素についても手順１と同様に最小値を選択し交換する。

	1	2	3	4	5
DAT	10	20	80	30	40

	1	2	3	4	5
DAT	10	20	30	80	40

	1	2	3	4	5
DAT	10	20	30	40	80

図３　整列が進んでいく過程

［プログラム］

```
○Sort_Selection(整数型：N, 整数型の配列：DAT)
  整数型：i, j, k, wk
  i ← 1
  while( i ≦ N − 1 )
      j ← i ＋ 1
      k ← i
      (1)
      while( j ≦ N )
          if( wk ＞ DAT[j] ) ◀━━ α
              wk ← DAT[j]
              k ← j
          endif
          j ← j ＋ 1
      endwhile
      wk ← DAT[i]
      DAT[i] ← DAT[k]
      (2)
      i ← i ＋ 1
  endwhile
```

(1)，(2) の解答群

ア．DAT[i] ← DAT[j]　　イ．DAT[j] ← DAT[k]

ウ．DAT[i] ← wk　　エ．DAT[k] ← wk

オ．wk ← DAT[i]　　カ．wk ← DAT[j]

＜設問２＞　次の要素同士の比較に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

設問１の選択法のプログラムを用いてデータを昇順に整列するとき，整列前のデータが昇順になっている場合と降順になっている場合を考える。要素同士の比較を行っているαの処理は，［　(3)　］

(3) の解答群

ア．降順の場合が多くなる。

イ．昇順の場合が多くなる。

ウ．どちらの場合も変わらない。

＜設問３＞　次の挿入法を用いた Sort_insert に関する説明を読み，プログラム中の［　　］に入れるべき適切な字句を解答群から選べ。

１次元配列 DAT[1]～DAT[N]に N 個のデータが格納されている。このデータを，挿入法により昇順に整列する。挿入法とは，整列済みのデータに対して，新たなデータを適切な位置に挿入し，整列済みの範囲を広げていく方法で，次の手順１，手順２を実行し,未整列部分が無くなったら終了する。

手順 1：最初の段階では，整列されていないため，最初の要素だけを整列済みの要素と考える。

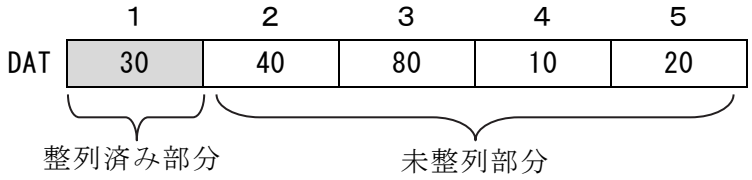

図４　配列 DAT の初期状態

手順２：未整列部分の要素を，DAT[2]から DAT[5]まで順に整列済み部分に挿入し，整列済み部分を増やしていく。

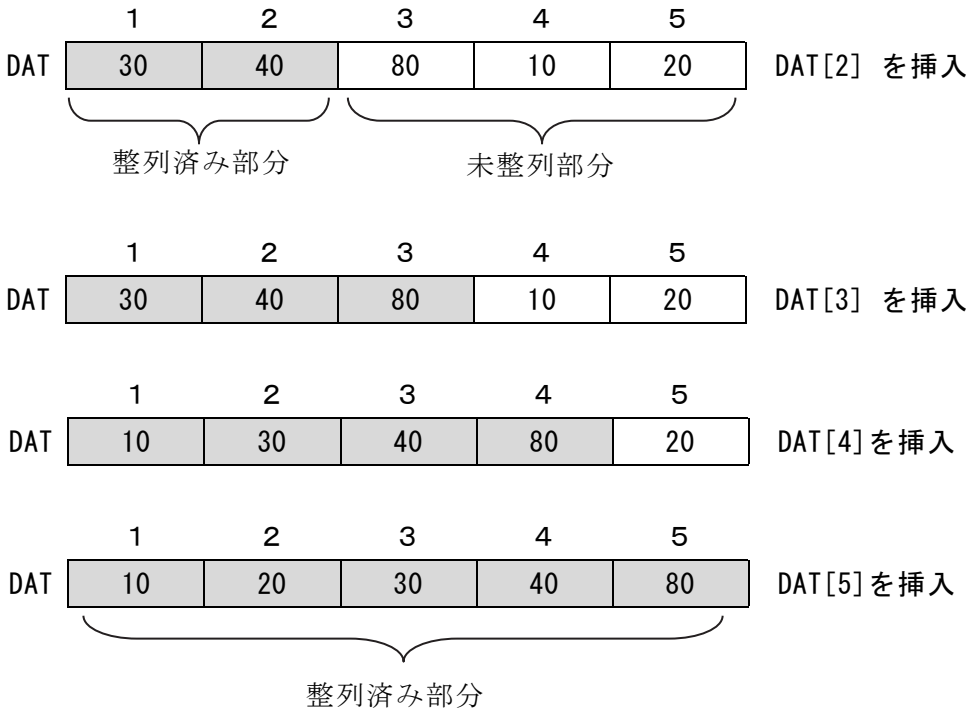

図５　整列が進んでいく過程

［プログラム］

```
○Sort_insert(整数型：N, 整数型の配列：DAT)
  整数型：i, k, wk, sw
  i ← 2
  while( i ≦ N )
      k ← i
      sw ← 0
      while( k ＞ 1 and sw ＝ 0 )
          if( DAT[k － 1] > DAT[k] ) ←――― β
              wk ← DAT[k]
              DAT[k] ← DAT[k － 1]
              DAT[k － 1] ← wk
              (4)
          else
              sw ← 1
          endif
      endwhile
      i ← i ＋ 1
  endwhile
```

(4) の解答群

ア．i ← i － 1　　イ．i ← i ＋ 1

ウ．k ← k － 1　　エ．k ← k ＋ 1

＜設問４＞　次の要素同士の比較に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

設問３の挿入法のプログラムを用いてデータを昇順に整列するとき，整列前のデータが昇順になっている場合と降順になっている場合を考える。要素同士の比較を行っているβの処理は，(5)

(5) の解答群

ア．降順の場合が多くなる。

イ．昇順の場合が多くなる。

ウ．どちらの場合も変わらない。

問題５　次の文字列の置換に関するプログラムの説明を読み，各設問に答えよ。

［文字列の置換の説明］

文字列の置換とは，ある文字列において特定の箇所を検索し，見つかった文字列を別の文字列に置き換えることである。例えば，検索対象文字列 ABCDBCE において，検索文字列を BC，置換文字列を EF とすると，置換結果文字列は AEFDEFE となる。

本問題における文字列の置換は，以下２つの手続きをもつクラスにより行われる。なお，各文字列は配列に１文字ずつ格納されているものとし，置換結果を格納する配列は作業に十分な大きさを持ち，配列の添え字は 0 から始める。

［手続き１の説明］

文字列の置換に必要となる検索処理を行う手続き fString()である。検索対象文字列 t[]から検索文字列 f[]を検索する場合，t[0]と f[0]，t[1]と f[1]，...，と比較し，f[]の全要素が t[]の特定の箇所と一致するかどうかの比較を行う(図１)。

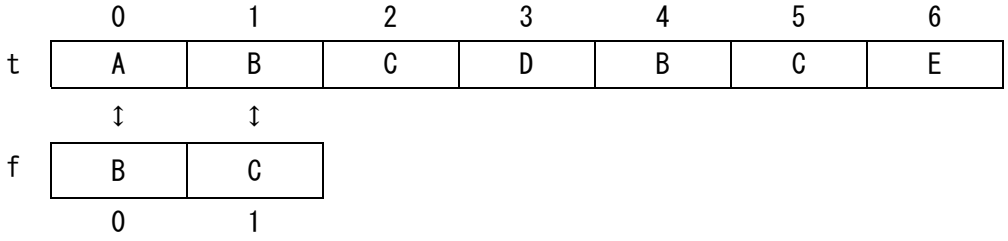

図１　t[]から f[]を検索

一致しない文字が出現した場合，t[1]と f[0]，t[2]と f[1]，...，と比較し，次の要素の比較を繰り返す。一致する箇所が見つかった場合，そのときの比較開始位置を返す。最後まで見つからない場合は－1 を返す(図２)。

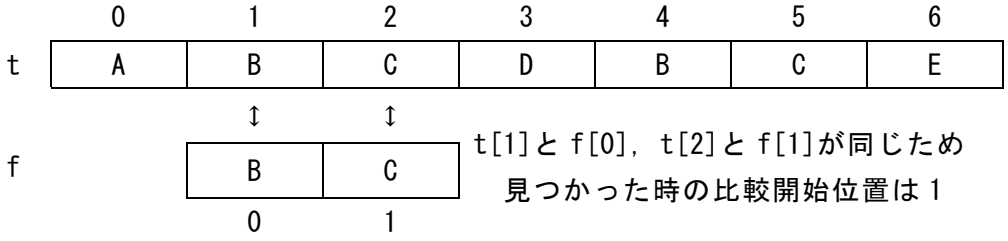

図２　次の要素の比較，および検索結果が見つかった場合

引数とその並び，および戻り値は以下である。

引　数　：　s…比較開始位置

戻り値　：　見つかった文字列 t[]の比較開始位置または－1

＜設問１＞　次の検索を行うプログラム中の［　　］に入れるべき適切な字句を解答群から選べ。

［プログラム］

```
○整数型：fString(整数型：s )
  文字型の配列：t, f
  整数型：tLen, fLen, pos, i, k, sw
  検索対象文字列を t に読み込む
  検索文字列を f に読み込む
  検索対象文字列 t の文字数を tLen に読み込む
  検索文字列 f の文字数を fLen に読み込む
  i ← s
  pos ← −1
  while( i ≦ [ (1) ] and pos = −1 )
      k ← 0
      sw ← 0
      while( [ (2) ] and sw = 0 )
          if( t[i ＋ k] = f[k] )
              k ← k ＋ 1
          else
              sw ← 1
          endif
      endwhile
      if( [ (3) ] )
          pos ← i
      else
          i ← i ＋ 1
      endif
  endwhile
  return pos
```

(1) の解答群

ア．fLen − 1　　イ．fLen − tLen

ウ．tLen − 1　　エ．tLen − fLen

(2)，(3) の解答群

ア．k ＜ tLen　　イ．k ＜ tLen − 1

ウ．k ＜ fLen　　エ．k ＜ fLen − 1

オ．k ＝ tLen　　カ．k ＝ tLen − 1

キ．k ＝ fLen　　ク．k ＝ fLen − 1

［手続き２の説明］

検索処理を踏まえ置換処理を行う手続き rString()である。検索対象文字列 t[]に存在する検索文字列 f[]を置換文字列 r[]で置き換える。置換処理後の文字列は結果用配列 res[]に格納し，返却する。置換は以下の３つの処理で行う。

処理１：

手続き fString()により検索を行う(図３)。

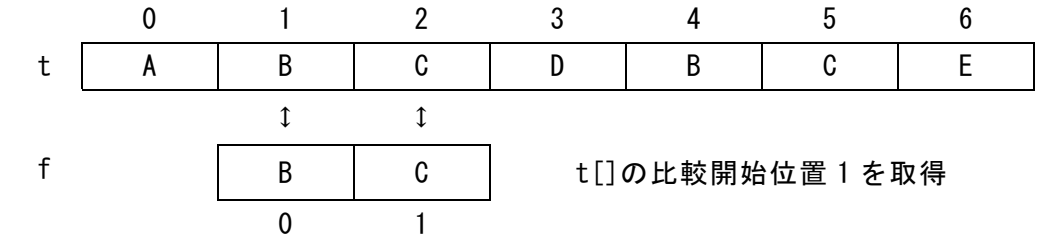

図３　t[]の中に f[]が存在するか検索

処理２：

t[]の中に f[]が存在する間，文字列置換を繰り返す。t[]の比較開始位置より前に存在する t[]の文字を res[]に格納し，t[]の比較開始位置から r[]の置換文字列を res[]に格納する。その後，新たな比較開始位置を取得する。以下は r[]が EF の場合の例である(図４)。

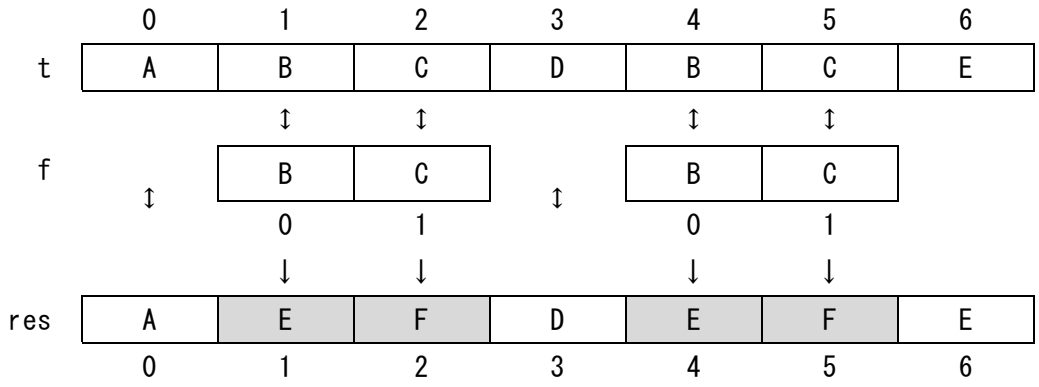

図４　文字列の置換（r[]が EF の場合の例）

処理３：

r[]の置換が完了しているため，t[]の未格納部分があれば res[]に格納する。

引数とその並び，および戻り値は以下である。

引　数　：　なし

戻り値　：　置換処理後の文字列

＜設問２＞　次の置換を行うプログラム中の［　　］に入れるべき適切な字句を解答群から選べ。

［プログラム］

```
○文字型の配列：rString()
  文字型の配列：t, r, res
  整数型：tLen, fLen, rLen, i, j, k, p
  検索対象文字列を t に読み込む
  置換文字列を r に読み込む
  検索対象文字列 t の文字数を tLen に読み込む
  検索文字列 f の文字数を fLen に読み込む
  置換文字列 r の文字数を rLen に読み込む
  i ← 0
  j ← 0
  p ← fString( 0 )
  while( p ≧ 0 )
     while( i ＜ p )
        [ (4) ]    / *  変換しない処理  * /
        i ← i ＋ 1
        j ← j ＋ 1
     endwhile
     k ← 0
     while( k ＜ rLen )
        [ (5) ]    / *  変換処理  * /
        j ← j ＋ 1
        k ← k ＋ 1
     endwhile
     i ← p ＋ [ (6) ]
     p ← fString( i )
  endwhile
  while( i ＜ tLen )
     res[j] ← t[i]
     i ← i ＋ 1
     j ← j ＋ 1
  endwhile
  return res
```

(4)，(5) の解答群

ア．**res[i] ← r[k]**　　イ．**res[i] ← t[k]**

ウ．**res[j] ← r[k]**　　エ．**res[j] ← t[i]**

オ．**res[k] ← r[i]**　　カ．**res[k] ← t[i]**

(6) の解答群

ア．**1**　　イ．**tLen**

ウ．**fLen**　　エ．**rLen**

＜メモ欄＞

＜メモ欄＞

＜メモ欄＞