文部科学省後援

情報システム試験

令和元年度前期 情報検定

＜実施　令和元年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（ＰＨＳ），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35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．試験後にお知らせする合否結果（合否通知），および合格者に交付する「合格証・認定証」はすべて，Ｗｅｂページ（ＰＣ，モバイル）での認証によるディジタル「合否通知」，ディジタル「合格証・認定証」に移行しました。
①団体宛にはこれまでと同様に合否結果一覧ほか，試験結果資料一式を送付します。
②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。

＜問題の構成＞

必須問題　全員解答

問題１～問題４	2 ページ～17 ページ

選択問題　次の問題から１問選択し解答せよ。

（選択した問題は解答用紙「選択欄」に必ずマークすること）

※選択欄にマークがなく，解答のみマークした場合は採点を行いません。

・Ｃ言語の問題	20 ページ～25 ページ
・表計算の問題	26 ページ～31 ページ
・アセンブラの問題	32 ページ～35 ページ

必須問題

問題１　次の線形リストに関する記述を読み，各設問に答えよ。

線形リストとは，データとその格納位置を示すポインタによって構成するデータ構造である。線形リストの先頭は root に格納されており，最後の要素のポインタには NULL を格納する。

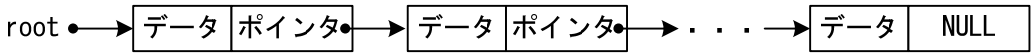


図１　リスト構造

ここでは root から順番に参照する片方向リストを扱うものとする。

また，線形リストは配列 list の２要素を使って表現し，添字が小さい方にデータを，大きい方にポインタを格納する。

＜設問１＞　次の線形リストへの追加に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

root と配列 list の状態が図２のような状態である時，list[6]と list[7]を利用して，データ 400 を格納する場合を考える。

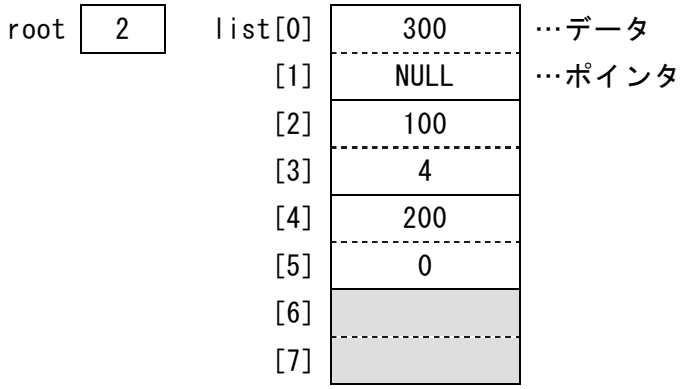


図２　配列 list

root から始まる線形リストの最後のデータとして 400 が参照されるように格納するには，次のように操作する。

・list[6]に 400 を格納する
・［　(1)　］に 6 を格納する
・［　(2)　］に NULL を格納する

(1)，(2) の解答群

ア．list[1]　　イ．list[3]　　ウ．list[5]　　エ．list[7]

＜設問２＞　次の線形リストからの削除に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

線形リスト上の要素を削除する場合は，ポインタの値を変更して削除対象となった要素を参照しないようにすればよい。

root と配列 list の内容が図３のような状態からデータを削除する場合を考える。

root	4

	list
[0]	200
[1]	6
[2]	400
[3]	NULL
[4]	100
[5]	0
[6]	300
[7]	2

図３　配列 list

・線形リストの先頭にあるデータを削除する場合は，root の値を変更する。図３の list[4]に格納されている 100 を削除する場合は，root を［　(3)　］に変更すればよい。

・線形リストの末尾にあるデータを削除する場合は，末尾のデータを指しているポインタの値を変更する。図３の list[2]に格納されている 400 を削除する場合は，list[7]を［　(4)　］に変更すればよい。

・上記以外の場合は，削除するデータを指しているポインタの値を，削除するデータが持っているポインタの値にする。図３の list[0]に格納されている 200 を削除する場合は，list[5]を［　(5)　］に変更すればよい。

(3) ～ (5) の解答群

ア．0　　イ．2　　ウ．4　　エ．6　　オ．NULL

問題を読みやすくするために,
このページは空白にしてあります。

問題２　次のバブルソートに関する記述を読み，各設問に答えよ。

［バブルソートの説明］

１次元配列 t[0]～t[n－1]に n 個のデータが格納されている。このようなデータを，隣接する要素間で大小の判定を繰り返しながら整列するアルゴリズムをバブルソートという。繰返しの継続条件の違いにより方法１と方法２の二つの方法を示す。

［方法１の説明］

手順１：配列の先頭から，隣接する要素を順次比較し，最大値を t[n－1]に求める。

図１に n＝5 とした例を示す

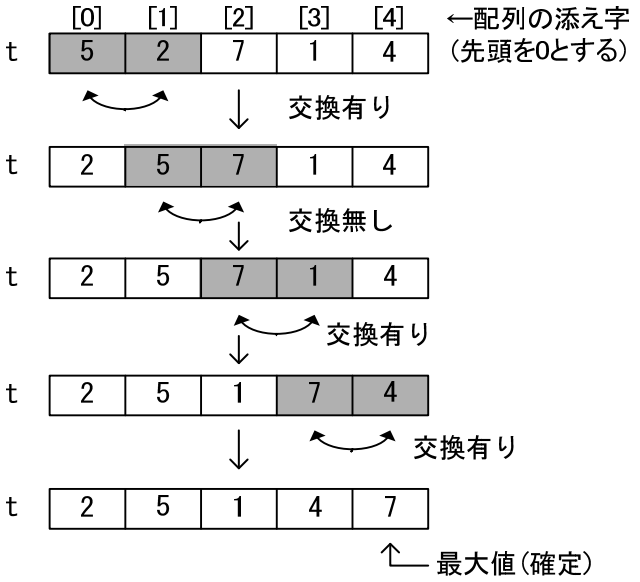


図１　n＝5 とした手順１の例

手順２：n を 1 ずつ減らしながら，手順 1 を n＝1 となるまで繰り返す。

［方法２の説明］

隣接要素間の比較であるため，図１のように比較すべき全要素の比較が終了した時点で交換が発生していなければ整列が終了することになる。そこで変数 sw を使って交換の有無を判断する。sw の初期値は 1 とする。

手順１：sw を 0 にして，方法１の手順１を実行する。ただし，交換が発生したときは sw を 1 とする。

手順２：sw が 1 の間，方法１の手順２を実行する。

＜設問１＞　次のバブルソートに関する流れ図中の［　　］に入れるべき適切な字句を解答群から選べ。

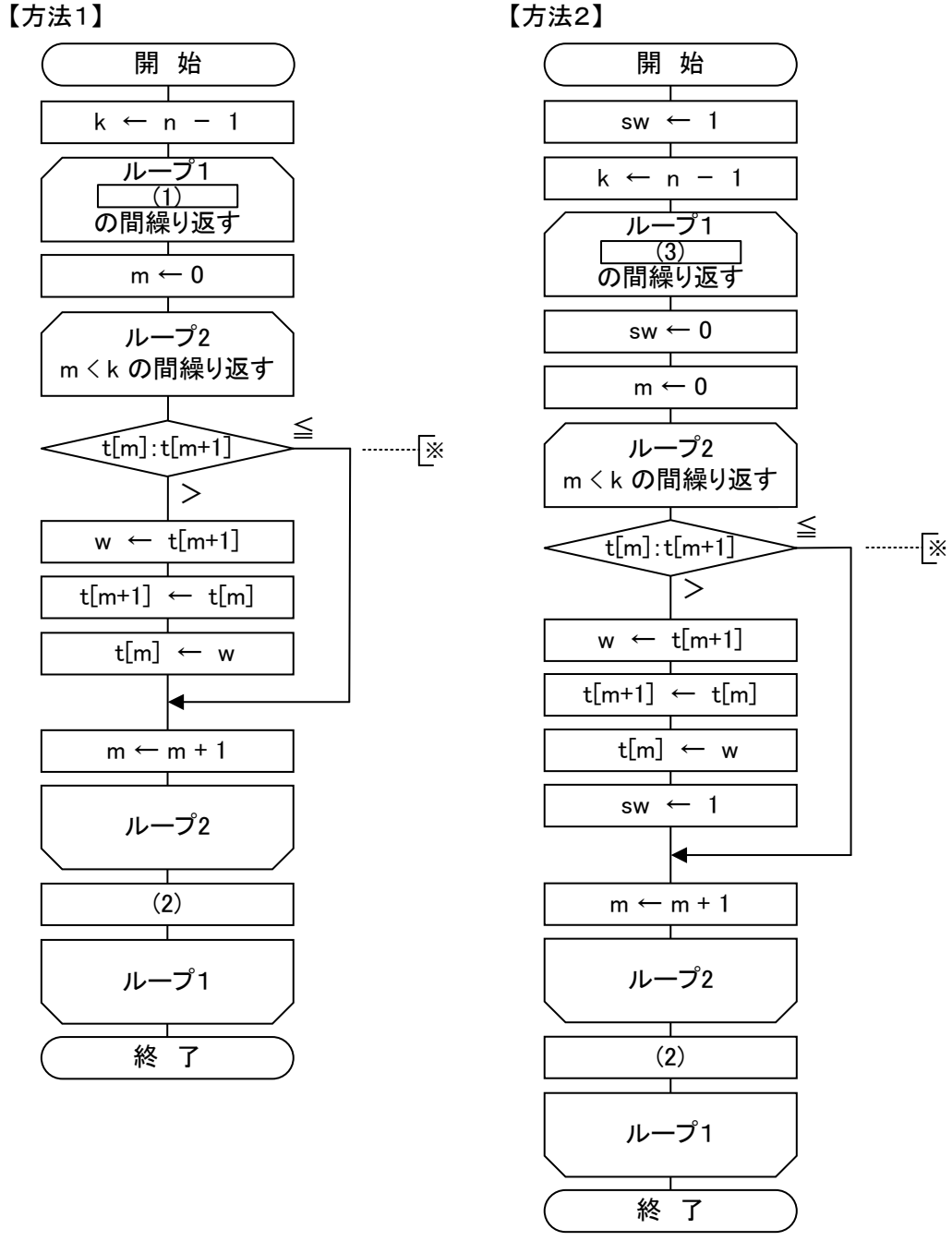


図２　二つの方法によるバブルソートの流れ図

(1) の解答群

ア．k＞0　　イ．k＞1　　ウ．k＜n－1　　エ．k＜n

(2) の解答群

ア．k ← k－1　　イ．k ← k＋1　　ウ．m ← m－1　　エ．m ← m＋1

(3) の解答群

ア．k＞0 かつ sw＝1　　イ．k＞0 または sw＝0

ウ．k＞1 かつ sw＝1　　エ．k＞1 または sw＝0

＜設問２＞　次のバブルソートの比較回数に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

1 次元配列 t[0]～t[n－1]に格納されているデータが図３の場合，図２の流れ図中※で示される t[m]と t[m＋1]の比較回数は，方法１が［(4)］回，方法２が［(5)］回となる。

	[0]	[1]	[2]	[3]	[4]
t	3	2	5	8	7

図３　1 次元配列 t の内容(n＝5)

(4)，(5) の解答群

ア．4　　イ．7　　ウ．10　　エ．15

問題３　次の文字列の置換に関する記述を読み，各設問に答えよ。

文字列の置換とは，ある文字列の中から特定の文字列を検索し，その文字列を別の文字列に置き換えることである。なお，文字列は１文字ずつ配列に格納し，配列は作業に十分な大きさを持つものとし，添字は０から始まるものとする。

＜設問１＞　次の文字列の検索に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

文字を格納している配列から対象となる文字列を検索する方法がある。例えば，配列ｓから配列ｆの文字列を検索する場合，s[0]と f[0]，s[1]と f[1]，…，と比較する。

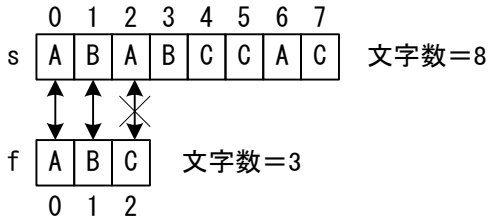


図１　配列ｓから配列ｆを検索する

しかし，図１の s[2]と f[2]のように，比較結果が一致しない場合は，s[1]と f[0]の比較から検索をやり直す。

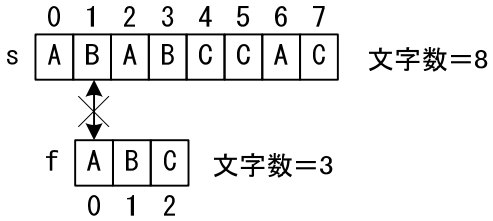


図２　s[1]と f[0]の比較からやり直す

図２のように，比較結果が再度途中で一致しなければ，s[2]と f[0]の比較から検索をやり直す。もし，配列ｓの連続する一部の要素と配列ｆの全要素が一致すれば，文字列が見つかったことになる。

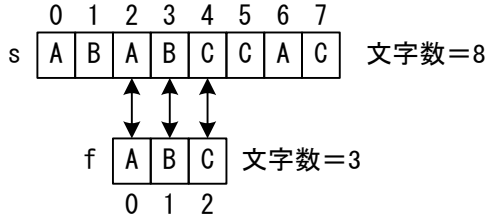


図３　配列ｆの全要素が一致した

ここで，配列 s の文字数を sLength，配列 f の文字数を fLength とし，文字列が見つかったかどうかを判断することを考える。

文字列が見つかった場合，f の添字が［ (1) ］になるまで一致したことになる。

また，配列 s の中に配列 f の文字列が存在しない場合は，配列 s の最大要素位置を超えないように比較しなければならない。図１～図３の例では，s[6]と f[0]の比較から始めようとすると，s[8]と f[2]まで比較することになり，配列 s の最大要素位置である 7 を超えてしまう。検索をやり直すときの配列 s の開始位置は，［ (2) ］以下になるように制御する必要がある。

(1) の解答群

ア．fLength － 1　　イ．fLength ＋ 1

ウ．sLength － 1　　エ．sLength ＋ 1

(2) の解答群

ア．fLength － sLength　　イ．fLength － sLength ＋ 1

ウ．sLength － fLength　　エ．sLength － fLength ＋ 1

＜設問２＞　次の文字列を検索する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］

文字列 s の指定した位置以降から,文字列 f を検索する流れ図 findString である。見つかった場合は比較開始位置を，そうでない場合は－1 を返却する。

なお，文字列 s および文字列 f は，配列 s および配列 f に 1 文字ずつ格納されており，配列の格納位置は 0 から始めるものとする。また，この流れ図の呼び出しに際して渡される引数とその並び，および返却値は次のようになっている。

引　数：配列 s[]　…　文字列 s
　　配列 f[]　…　文字列 f
　　sLength　…　文字列 s の長さ
　　fLength　…　文字列 f の長さ
　　startPos　…　比較開始位置

返却値：見つかった文字列 s の開始位置または－1

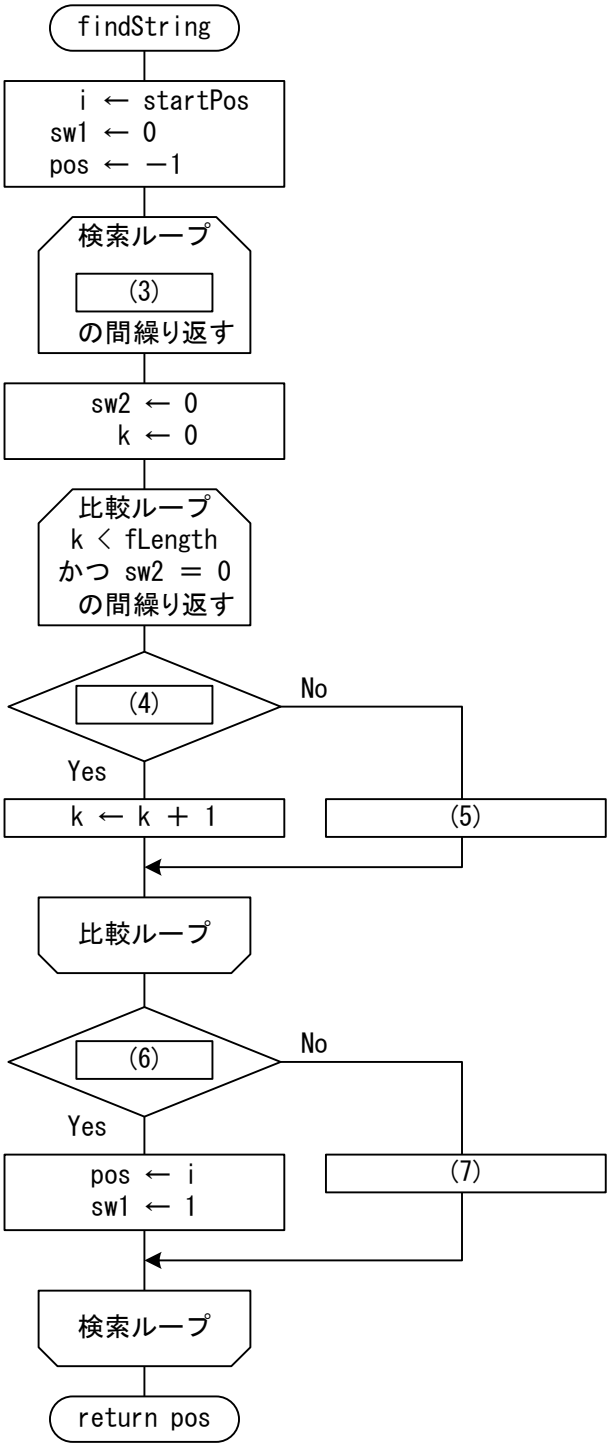


図４　findStringの流れ図

(3) の解答群

ア. i ≦ sLength − fLength かつ sw1 ＝ 0

イ. i ≦ sLength − fLength または sw1 ＝ 0

ウ. i ＞ sLength − fLength かつ sw1 ＝ 0

エ. i ＞ sLength − fLength または sw1 ＝ 0

(4) の解答群

ア．s[i－k] ＝ f[i]　　イ．s[i－k] ＝ f[k]
ウ．s[i＋k] ＝ f[i]　　エ．s[i＋k] ＝ f[k]

(5) の解答群

ア．i ← 1　　イ．k ← 1　　ウ．pos ← 1　　エ．sw2 ← 1

(6) の解答群

ア．sw1 ＝ 0　　イ．sw1 ＝ 1　　ウ．sw2 ＝ 0　　エ．sw2 ＝ 1

(7) の解答群

ア．i ← fLength ＋ 1　　イ．i ← i ＋ 1
ウ．i ← i － 1　　エ．i ← k ＋ 1

＜設問３＞　次の文字列の置換に関する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。なお，解答は重複して選んでもよい。

［流れ図の説明］

文字列 s の中に存在する文字列 f を文字列 r で全て置き換えた文字列を返却する流れ図 replaceString である。

なお，文字列 s および文字列 f，文字列 r は，配列 s および配列 f，配列 r に 1 文字ずつ格納されており，配列の格納位置は 0 から始めるものとする。流れ図の呼び出しに際して渡される引数の並び，および返却値は次のようになっている。

引　数：配列 s[]　…　文字列 s
　　配列 f[]　…　文字列 f
　　配列 r[]　…　文字列 r
　　sLength　…　文字列 s の長さ
　　fLength　…　文字列 f の長さ
　　rLength　…　文字列 r の長さ
返却値：置換後の文字列

また，この流れ図では図４の流れ図 findString を呼び出しており，その時に渡す引数は，配列 s[]，配列 f[]，sLength，fLength，比較開始位置(findString で使用する startPos)である。

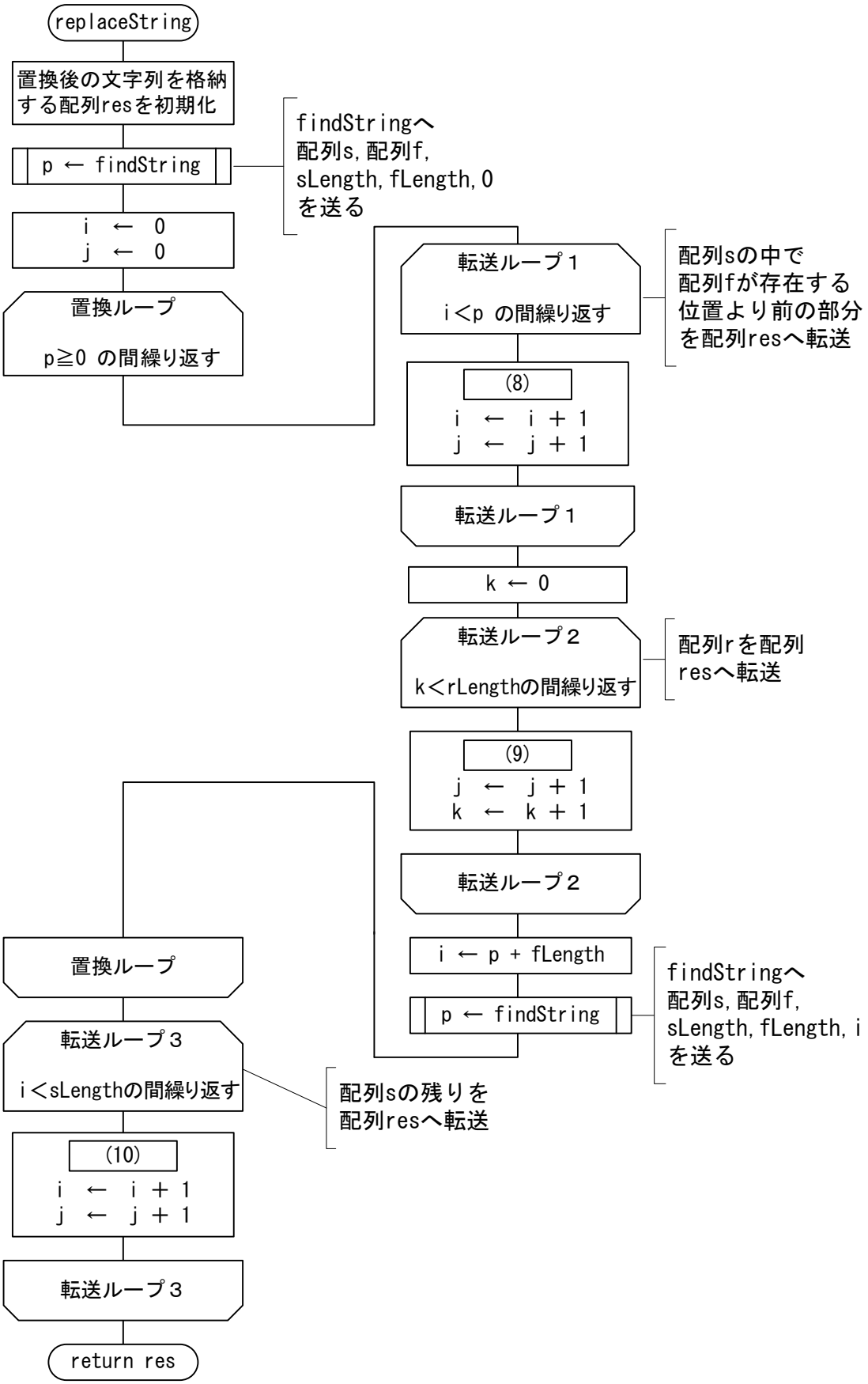


図５　replaceStringの流れ図

(8) ～ (10) の解答群

ア. res[i] ← r[k]　　イ. res[i] ← s[j]

ウ. res[j] ← r[k]　　エ. res[j] ← s[i]

オ. res[k] ← r[j]　　カ. res[k] ← s[i]

問題４　次のプログラムの説明を読み，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。

［プログラムの説明］

異なる数値が昇順に格納されている配列dataの中から，変数Xと同じ数値が格納されている要素を２分探索法を用いて探し，その要素を配列dataから削除するプログラムBinary_sである。なお，変数d_lenには配列dataの要素数が格納されており，配列の添字は0から始まる。

［手順］

① 探索範囲の先頭要素の添字をL，末尾要素の添字をHとする。なお，初期値は，Lは0，Hはd_len－1である。

② 探索範囲の中央要素となるdata[M]と比較する。ただし，Mは(L＋H)÷2とし，小数点以下は切り捨てる。

data[M]＜Xなら，Lを M＋1とし，次の探索範囲を，配列の要素位置がMより大きい方とする。

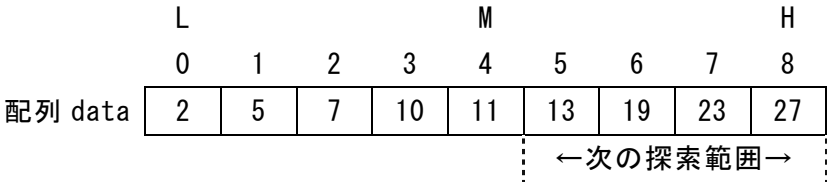


図１　比較例１

data[M]＞Xなら，Hを M－1とし，次の探索範囲を，配列の要素位置がMより小さい方とする。

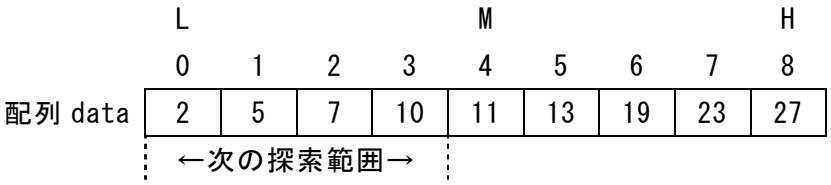


図２　比較例２

③ 変数Xと同じ数値が見つかった場合，その要素を配列dataから削除し，当該要素以降の要素を順に１つずつ前に移動する。また，変数d_lenの値を１減らす。例えば，配列dataの内容が図１と同じ状態で，変数d_len＝9，変数X＝19の場合，変数X＝19と同じ数値が配列data[6]に存在したため，配列data[7]以降の要素を順に１つずつ前に移動し，変数d_lenを８とする。

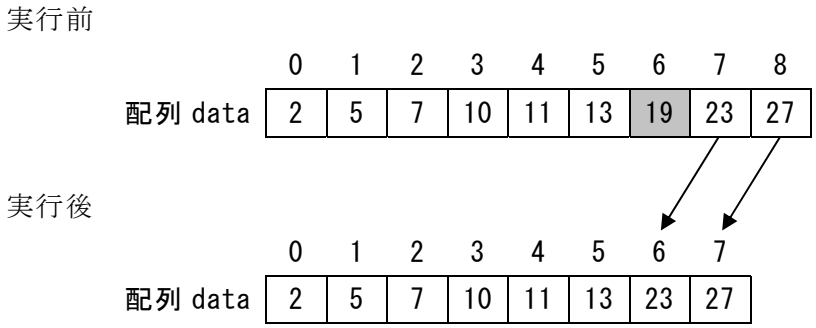


図３　要素の削除例

④ 変数 X と同じ数値がなかった場合，エラーメッセージを表示する。

［擬似言語の記述形式の説明］

記述形式	説明
○	手続き，変数などの名前，型などを宣言する
・変数 ← 式	変数に式の値を代入する
/* 文 */	注釈を記述する
▲ 条件式 ・処理１ ――― ・処理２ ▼	選択処理を示す。 条件式が真の時は処理１を実行し， 偽の時は処理２を実行する。
■ 条件式 ・処理 ■	前判定繰り返し処理を示す。 条件式が真の間，処理を実行する。

［演算子と優先順位］

演算の種類	演算子	優先順位
単項演算	＋，－，not	高 ▲
乗除演算	＊，／，％	
加減演算	＋，－	
関係演算	＞，＜，≧，≦，＝，≠	
論理積	and	
論理和	or	▼ 低

注記　整数同士の除算では，整数の商を結果として返す。％演算子は剰余算を表す。

［プログラム］

○**Binary_s**（整数型：**data[]**，整数型：**d_len**，整数型：**X**）

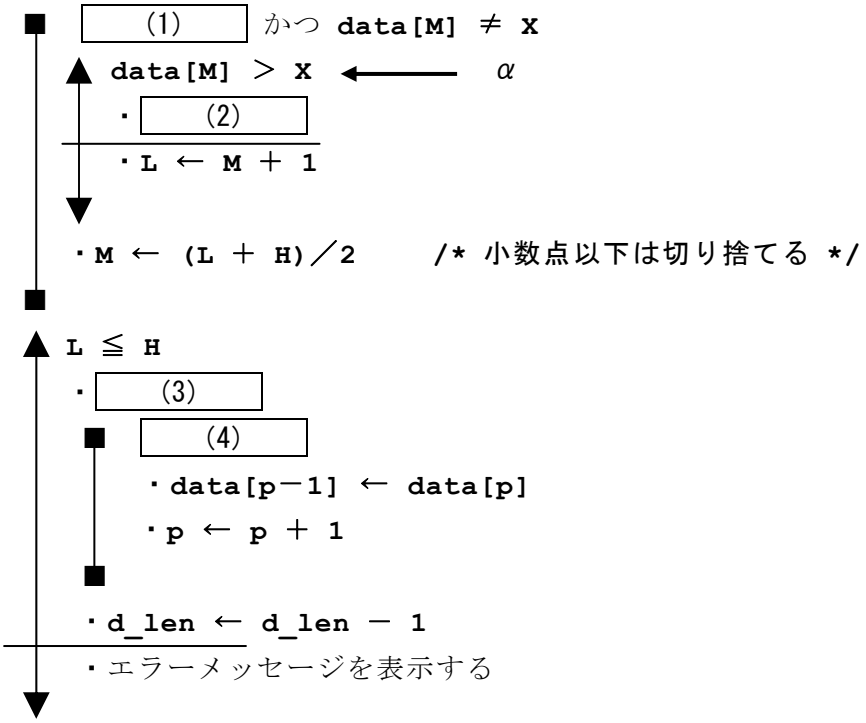
○整数型：**L**，**H**，**M**，**p**

・**L ← 0**

・**H ← d_len − 1**

・**M ← (L ＋ H)／2**　　**/* 小数点以下は切り捨てる */**

/* 配列の中から X を探索する */



＜設問１＞　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)，(4) の解答群

ア．**L ＜ H**　　イ．**L ≦ H**　　ウ．**L ＞ H**

エ．**L ≧ H**　　オ．**p ＜ d_len− 1**　　カ．**p ＜ d_len**

(2)，(3) の解答群

ア．**H ← M − 1**　　イ．**H ← M ＋ 1**　　ウ．**L ← M − 1**

エ．**L ← M ＋ 1**　　オ．**p ← M − 1**　　カ．**p ← M ＋ 1**

＜設問２＞　配列 data の内容が次のような場合，プログラム中のαを実行するときの変数 L，H，M をトレースした表の［　　］に入れるべき適切な字句を解答群から選べ。

X	10

	0	1	2	3	4	5	6	7	8	9
data	2	5	9	10	15	23	29	33	37	42

表　トレースの内容

順番	L	H	M
1	0	9	4
2	0	3	1
3	(5)		
4	3	3	3

(5) の解答群

	L	H	M
ア.	0	2	1
イ.	1	3	2
ウ.	2	3	2
エ.	2	3	3

問題を読みやすくするために,
このページは空白にしてあります。

＜ 選 択 問 題 ＞

選択問題は問題から１つ選択し解答せよ。
選択した問題は必ず，解答用紙「選択欄」にマークすること。
※選択欄にマークがなく，解答のみの場合は採点を行いません。

各構成は以下のとおり。

選択問題

選択問題　C言語の問題

次のＣ言語プログラムの説明を読み，各設問に答えよ。

［プログラムの説明］

データを可逆圧縮するアルゴリズムの一つであるハフマン符号化に用いるハフマン木を作るプログラムである。

ハフマン符号は，データとその出現回数により符号化をするものであり，ここでは8ビット文字を符号化する。ハフマン符号は次の手順で求める。

①　文字ごとの出現回数を取得した節を作成する(図１)。

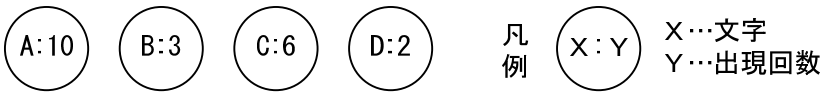


図１　文字と出現回数

②　要素の中で，一番出現回数が少ない節と二番目に出現回数が少ない節を子とした二分木を作成する。親の節には，子の節の出現回数の和を格納する。ここでは，出現回数が少ない方を左，多い方を右の子とする。

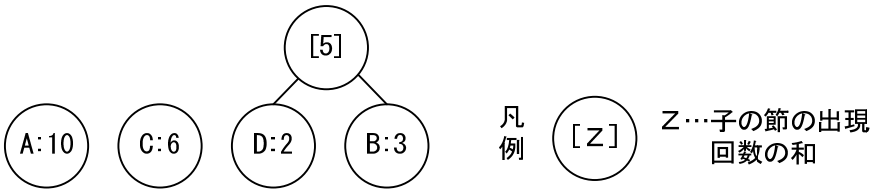


図２　一番少ない出現回数と二番目に少ない出現回数で二分木を作成

③　②の処理を全ての節が二分木構造になるまで繰返す。次に，文字の符号化を行うため，左へたどる枝には0，右へたどる枝には1を割り当てる(図３)。

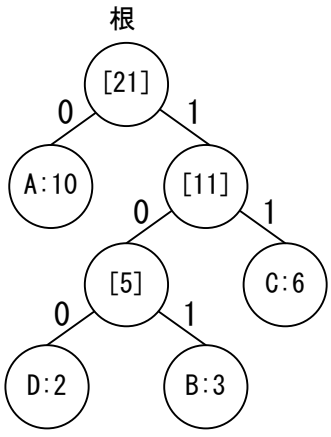


図３　ハフマン木

④ 図３のハフマン木の根から順番に各文字の節へたどった枝にある 0 または 1 を並べたものが，文字を符号化したビット列となる。図３の例では，A は″0″，B は″101″，C は″11″，D は″100″となる。

＜設問１＞　次のハフマン符号に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

文字列″AAAAABBCDDDD″をハフマン符号により符号化する場合を考える。

この文字列は，’A’が 5 個，’B’が 2 個，’C’が 1 個，’D’が 4 個出現するので，各文字を表すビット列は，次の表のようになる。(a)～(d)に入れる字句の組合せを（1）の解答群から選べ。

表　文字と符号化したビット列

文字	’A’	’B’	’C’	’D’
ビット列	(a)	(b)	(c)	(d)

元の文字列は 12 文字であるため，1 文字を 8 ビットとして 96 ビットになるが，符号化後は［(2)］ビットになる。

(1) の解答群

	(a)	(b)	(c)	(d)
ア.	0	101	100	11
イ.	0	100	101	11
ウ.	1	110	011	11
エ.	1	100	101	11

(2) の解答群

ア．20　　イ．22　　ウ．26　　エ．30

＜設問２＞　次のハフマン木を作成するプログラムの［　　］に入れるべき適切な字句を解答群から選べ。

ハフマン木は，次の項目を構造体で表現した配列に格納する。

文字	出現回数	親の節	左の子の節	右の子の節	フラグ

図４　格納要素の構造体

・プログラムが呼び出される前に配列の全要素は初期化されている(文字は'¥0'，出現回数は 0，親の節，左の子の節，右の子の節，フラグは-1 に設定する)。
・子の節を格納する場合は，文字と出現回数，親の節，フラグに値が格納される。
・親の節を格納する場合は，出現回数，左の子の節，右の子の節，フラグに値が格納される。なお，出現回数は，子要素の出現回数の和である。
・親の節，左の子の節，右の子の節には，それぞれが格納されている配列内の位置を格納する。
・フラグは，子の節は 0，親の節または二分木に含まれていない節は 1 を格納する。
・構造体配列の要素数はプログラムの実行に影響がない大きさが確保されている。
・出現回数の総数は 65535 未満とする。

文字列"AAAAABBCDDDD"からハフマン木を作成した場合，次の図５のような配列になる。なお，根は最後の行である。

添字	文字	出現回数	親の節	左の子の節	右の子の節	フラグ
0	A	5	6	-1	-1	0
1	B	2	4	-1	-1	0
2	C	1	4	-1	-1	0
3	D	4	5	-1	-1	0
4	¥0	3	5	2	1	0
5	¥0	7	6	4	3	0
6	¥0	12	-1	0	5	1

図５　ハフマン木を作成した配列

［関数の説明］

countCharacter 関数

引　数：*text(文字列)，*node(ハフマン木を格納する構造体配列)

機　能：文字列の各文字の出現回数をハフマン木に格納する

戻り値：ハフマン木に格納した要素数

makeTree 関数

引　数：n(構造体配列に格納されている要素数)，*node(ハフマン木を格納する構造体配列)

機　能：構造体配列に格納された要素に対してハフマン木を作成する。この関数は，countCharacter を実行後に呼び出される

戻り値：ハフマン木の根の位置

［プログラム］

```
#define DUMMY_POS (-1)          /* 仮の位置として使う */
#define DUMMY_VALUE 0xffff      /* 仮の最小値として使う */
struct NODE {
    char character;
    int count;
    int parentNode;
    int leftChild;
    int rightChild;
    int flag;
};
int countCharacter(char *text, struct NODE *node) {
    int i, k, n, pos;
    n = 0;    /* 配列 node の添え字 */
    i = 0;    /* 配列 text の添え字 */
    while(text[i] != '¥0') {    /* 文字列の最後まで繰返す */
        /* text[i]の文字がハフマン木に格納済みかを調べる */
        pos = DUMMY_POS;
        for(k = 0; k < n && pos < 0; k++) {
            if (text[i] == node[k].character) {
                pos = k;
            }
        }
        if (pos < 0) {    /* 新しい節を登録 */
            node[n].character = text[i];
            (3) ;
            node[n].flag = 1;
            n++;
        } else {     /* 既にある節の出現回数に加算 */
            (4) ;
        }
        (5) ;
    }
    return n;
}
```

```
int makeTree(int p, struct NODE *node) {
    int i, n, min1, min2, min1Value, min2Value, loop_sw;
    n = p;
    loop_sw = 0;
    while(loop_sw == 0) {
        /* 配列内で一番少ない出現回数と二番目に少ない出現回数を求める */
        min1 = min2 = DUMMY_POS;        /* 仮の最小値の位置を設定 */
        min1Value = min2Value = DUMMY_VALUE; /* 仮の最小値を設定 */
        for(i = 0; i < n; i++) {
            if (node[i].flag == 1) {
                if (node[i].count < min1Value) {
                    min2 = min1;
                    min1 = i;
                    min2Value = min1Value;
                    min1Value = node[i].count;
                } else if (node[i].count < min2Value) {
                    min2 = i;
                    min2Value = node[i].count;
                }
            }
        }
        if (min2 == DUMMY_POS) { /* min2 が変更されていなければ終了 */
            loop_sw = 1;
        } else {
            node[n].leftChild = min1;  /* 親の節を追加する */
            node[n].rightChild = min2;
            node[n].count = [ (6) ];
            node[n].flag = 1;
            node[min1].parentNode   /* 子の節に親の節の位置を格納 */
                = node[min2].parentNode = [ (7) ];
            node[min1].flag   /* 二分木の子の節として登録済みとする */
                = node[min2].flag = 0;
            [ (8) ];   /* 次の要素へ進める */
        }
    }
    return n - 1;
}
```

(3) の解答群

ア. `node[n].count = 0`　　イ. `node[n].count = 1`

ウ. `node[pos].count = 0`　　エ. `node[pos].count = 1`

(4) の解答群

ア. `node[i - 1].count++`　　イ. `node[pos - 1].count++`

ウ. `node[i].count++`　　エ. `node[pos].count++`

(5) の解答群

ア. `i++`　　イ. `k++`　　ウ. `n++`　　エ. `pos++`

(6) の解答群

ア. `node[min1].count`

イ. `node[min2].count`

ウ. `node[min1].count + node[min2].count`

エ. `node[n].count + node[min1].count + node[min2].count`

(7) の解答群

ア. `n`　　イ. `node[n].parentNode`

ウ. `node[pos].parentNode`　　エ. `pos`

(8) の解答群

ア. `n++`　　イ. `n = node[n].leftChild`

ウ. `n = node[n].parentNode`　　エ. `n = node[n].rightChild`

選択問題　表計算の問題

次の表計算ソフトの記述を読み，各設問に答えよ。

この問題で使用する表計算ソフトの仕様は下記のとおりである。

AVERAGE 関数
　範囲内のセルに含まれる数値の平均を返す。
　書式：AVERAGE(範囲)

COUNT 関数
　範囲に含まれる数値の個数を返す。
　書式：COUNT(範囲)

COUNTIF 関数
　範囲に含まれるセルのうち，条件に一致するセルの個数を返す。
　書式：COUNTIF(範囲，条件)

IF 関数
　条件が真のときに真の場合，偽のときに偽の場合の計算結果や値を返す。
　書式：IF(条件，真の場合，偽の場合)

LEFT 関数
　文字列の左端から文字数で指定した位置までの文字列を返す。
　書式：LEFT(文字列，文字数)

RIGHT 関数
　文字列の右端から文字数で指定した位置までの文字列を返す。
　書式：RIGHT(文字列，文字数)

RANK 関数
　範囲内の数値を並べたときに何番目になるか(順位)を返す。順序は，降順の場合は 0，昇順の場合は 1 を設定する。なお，範囲内の検査値に同じものがあれば同じ順位を返し，以降の順位に欠番が生じる。
　書式：RANK(検査値，範囲，順序)

ROUNDDOWN 関数
　指定した桁で値を切り捨てる。桁数が正の数であれば小数点以下，負の数であれば小数点以上の桁になる。例えば，１にすると小数点以下第２位以下の桁を切り捨てて小数点以下第１位までを表示し，-1 にすると１の位以下の桁を切り捨てる。

書式：ROUNDDOWN(式または値，桁数)

ROUNDUP 関数
指定した桁で値を切り上げる。桁数が正の数であれば小数点以下，負の数であれば小数点以上の桁になる。例えば，１にすると小数点以下第２位以下の桁を切り上げて小数点以下第１位までを表示し，-1 にすると１の位以下の桁を切り上げる。
書式：ROUNDUP(式または値，桁数)

VLOOKUP 関数
検索範囲から，検索値を探し，位置で指定した列の値を返す。位置は１から始まる相対的な値であり，検索範囲中に見つけた行の中で，左から何番目の列かを示す。検索方法は 0 または 1 を指定し，0 の場合は完全に一致する値を，1 の場合は検索値以下の最大値を探す。
書式：VLOOKUP(検索値，検索範囲，位置，検索方法)

式
＝に続いて計算式や関数などを入力する。

セル番地の絶対参照
セル番地に＄を付けることで，絶対番地(絶対参照)を表す。

別シートの参照
ワークシート名に「！」を付けてセル位置を指定することにより，別のワークシートを参照できる。
例：ワークシート名「集計」のセル A1 を参照する場合は，「集計!A1」と記述する。

＆演算子
文字列を結合する。

J 専門学校のＴ先生は，担当する教科の成績を評価するため，表計算ソフトを利用している。この教科は１～３組の 120 中 50 人が履修しており，期末試験の得点に小テストと課題も含め成績の評価を行う。また，入学年度，組，番号は文字列で入力されている。

	A	B	C	D
1	入学年度	組	番号	氏名
2	2019	1	01	安部　省三
3	2019	1	02	井原　康之
4	2019	1	03	宇野　俊昭
5	2019	1	04	羽田　清志
6	2019	1	05	河本　綾子
:	:	:	:	:
121	2019	3	40	若松　哲朗

図１　「学生名簿」ワークシート

＜設問１＞　次の「学生名簿」ワークシートの拡張に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

	A	B	C	D	E
1	入学年度	組	番号	学生番号	氏名
2	2019	1	01	19101	安部　省三
3	2019	1	02	19102	井原　康之
4	2019	1	03	19103	宇野　俊昭
5	2019	1	04	19104	羽田　清志
6	2019	1	05	19105	河本　綾子
:	:	:	:	:	:
121	2019	3	40	19340	若松　哲朗

図２　「学生名簿」の拡張ワークシート

学生番号を求めるため，氏名の列の前に１列挿入して学生番号の列を作成し，そのセル D2 に次の式を入力し，セル D3～D121 まで複写した。

なお，学生番号は，入学年度の下２桁と組，番号の連結となっている。

= ［　(1)　］

(1) の解答群

ア．LEFT(A$2, 2) & B$2 & C$2　　イ．LEFT(A2, 1) & $B2 & $C2

ウ．RIGHT(A2, 2) & B2 & C2　　エ．RIGHT(A2, 1) & B2 & C2

＜設問２＞　次の「小テスト集計」ワークシートの作成に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

Ｔ先生は，10 点満点の小テストを５回実施しており，図３の「小テスト集計」ワークシートを作成した。

	A	B	C	D	E	F	G	H	I	J
1		小テスト								
2	学生番号	①	②	③	④	⑤	合計	平均	６点以上の個数	判定
3	19101	9	9	8	7	7	40	8	5	A
4	19102	8	7	7	10	8	40	8	5	A
5	19104	6	6	4	9	8	33	6.6	4	C
6	19105	6	5	8	8	4	31	6.2	3	C
7	19107	9	8	7	7	8	39	7.8	5	B
:	:	:	:	:	:	:	:	:	:	:
52	19340	4	5	9	10	9	37	7.4	3	B
53	平　均	7.6	7.3	6.9	7.4	7	36.2	7.24		
54	６点以上の人数	43	39	38	40	40				

図３　「小テスト集計」ワークシート

A 列は学生番号を，B 列から F 列は小テストの結果を入力した。学生ごとの合計を G 列に，平均を H 列に集計した。

I 列は，学生ごとに小テストの合格点(６点以上)の個数を表示するため，セル I3 に次の式を入力し，セル I4～I52 まで複写した。

= (2)

判定には，合計が 40 点以上の場合は「A」，35～39 点の場合は「B」，34～30 点の場合は「C」，30 点未満の場合は「D」を表示するため，セル J3 に次の式を入力し，セル J4～J52 まで複写した。

= (3)

53 行には，各小テストと個人の合計と平均の平均を，54 行には，各小テストにおいて合格点(６点以上)の人数を集計した。

(2) の解答群

ア．COUNT(B$3:F$3, ″>6″)　　イ．COUNT(B3:F3, ″>=6″)

ウ．COUNTIF(B$3:F$3, ″>6″)　　エ．COUNTIF(B3:F3, ″>=6″)

(3) の解答群

ア．IF(G$3 <= 30, ″D″, IF(G$3 <= 35, ″C″, IF(G$3 <= 40, ″B″, ″A″)))

イ．IF(G3 <= 30, ″D″, IF(G3 <= 35, ″C″, IF(G3 <= 40, ″B″, ″A″)))

ウ．IF(G3 >= 40, ″A″, IF(G3 >= 35, ″B″, IF(G3 >= 30, ″C″, ″D″)))

エ．IF(G3 > 40, ″A″, IF(G3 > 35, ″B″, IF(G3 > 30, ″C″, ″D″)))

＜設問３＞　次の「小テストクロス集計」ワークシートの作成に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

「小テスト集計」ワークシートをもとに，図４の「小テストクロス集計」ワークシートを作成した。

	A	B	C	D	E	F
1		小テスト				
2	得点	①	②	③	④	⑤
3	10	9	8	10	8	4
4	9	11	10	3	8	7
5	8	7	7	5	9	13
6	7	5	6	8	11	6
7	6	10	7	11	3	9
:	:	:	:	:	:	:
13	0	0	0	0	0	0
14	合計	50	50	50	50	50

図４　「小テストクロス集計」ワークシート

A 列は 10～0 の得点を，セル B2～F2 には小テストの回数を入力した。

小テストごとに各得点の人数を表示するため，セル B3 に次の式を入力し，セル C3～F3，B4～F13 まで複写した。

= (4)

14 行には，各小テストの人数を集計した。

(4) の解答群

ア．COUNTIF(小テスト集計!$B3:$B52, $A3)

イ．COUNTIF(小テスト集計!B$3:B$52, $A3)

ウ．COUNTIF(小テスト集計!B$3:B$52, A3)

エ．COUNTIF(小テスト集計!B3:B52, A3)

＜設問４＞　次の「成績表」ワークシートの作成に関する記述中の［　　］に入れるべき適切な字句を解答群から選べ。

小テストの合計，課題，期末試験の得点を図５の「成績表」ワークシートにまとめた。なお，課題は 20 点，期末試験は 100 点満点とする。

	A	B	C	D	E	F	G
1	学生番号	氏名	小テストの合計	課題	期末試験	評価	順位
2	19101	安部　省三	40	18	82	83	17
3	19102	井原　康之	40	13	53	60	40
4	19104	羽田　清志	33	13	91	84	15
5	19105	河本　綾子	31	17	95	88	9
6	19107	梶原　英晴	39	16	86	84	15
:	:	:	:	:	:	:	:
51	19340	若松　哲朗	37	15	48	56	48
52	平　均		36.2	17.3	73.1	74.7	

図５　「成績表」ワークシート

A 列は，「小テスト集計」ワークシートから学生番号を複写する。

B 列は，図２の「学生名簿」ワークシートから学生番号で検索し，氏名を表示する。セル B2 に次の式を入力し，セル B3～B51 まで複写した。

= (5)

C 列は，「小テスト集計」ワークシートから学生番号で検索し，合計を表示する。セル C2 に次の式を入力し，セル C3～C51 まで複写した。

= (6)

D 列は課題の得点を，E 列は期末試験の得点を入力した。

F 列は，小テストの合計は 20 点，課題は 10 点，期末試験は 70 点満点と換算し，評価を求める。セル F2 に次の式を入力し，セル F3～F51 まで複写した。なお，小数点以下の数値は切り上げる。

= (7)

G 列は，評価の降順における順位を求める。セル G2 に次の式を入力し，セル G3～G51 まで複写した。

= (8)

52 行は，各得点の平均を求める。セル C52 に次の式を入力し，セル D52～F52 まで複写した。

= (9)

(5) の解答群

ア. VLOOKUP(A2, 学生名簿!D$2:E$121, 2, 0)
イ. VLOOKUP(A2, 学生名簿!D2:E121, 2, 0)
ウ. VLOOKUP(A2, 学生名簿!D$2:E$121, 4, 0)
エ. VLOOKUP(A2, 学生名簿!D2:E121, 4, 0)

(6) の解答群

ア. VLOOKUP(A2,小テスト集計!$A3:$G52, 6, 0)
イ. VLOOKUP(A2,小テスト集計!A3:G52, 6, 1)
ウ. VLOOKUP(A2,小テスト集計!$A3:$G52, 7, 0)
エ. VLOOKUP(A2,小テスト集計!A$3:G$52, 7, 0)

(7) の解答群

ア. ROUNDDOWN((C2 * 0.2 + D2 * 0.1 + E2 * 0.7), 0)
イ. ROUNDDOWN((C2 * 20 / 50 + D2 * 10 / 20 + E2 * 0.7), 0)
ウ. ROUNDUP((C2 * 0.2 + D2 * 0.1 + E2 * 0.7), 0)
エ. ROUNDUP((C2 * 20 / 50 + D2 * 10 / 20 + E2 * 0.7), 0)

(8) の解答群

ア. RANK(F2, F$2:F$51, 0)　　イ. RANK(F2, F2:F51, 0)
ウ. RANK(F2, F$2:F$51, 1)　　エ. RANK(F2, F2:F51, 1)

(9) の解答群

ア. AVERAGE($C2:$C51)　　イ. AVERAGE($C2:C$51)
ウ. AVERAGE(C$2:$C51)　　エ. AVERAGE(C2:C51)

選択問題　アセンブラの問題

次のアセンブラ言語CASLⅡプログラムの説明を読み，各設問に答えよ。

［プログラムの説明］

誤り訂正符号の一つであるハミング符号の冗長ビットを求める副プログラム HUMM である。

［ハミング符号の説明］

ハミング符号は,4 ビットのデータ($X_4X_3X_2X_1$)に対して 3 ビットの冗長ビット($P_3P_2P_1$)を付加することで，1 ビットの誤りを自動訂正でき，補助記憶装置に書き込むデータなどに利用されている。

冗長ビットは，次のように求める

$$P_1 = X_1 \oplus X_3 \oplus X_4$$
$$P_2 = X_1 \oplus X_2 \oplus X_4$$
$$P_3 = X_1 \oplus X_2 \oplus X_3$$

（注）　⊕：排他的論理和

プログラムでは，図に示すように，データは DAT 番地の下位 4 ビットに格納されており，冗長ビットを ECC 番地に求める。図では，見やすいように 4 ビットごとに空白を挿入している。

DAT 番地	0000 0000 0000 $X_4X_3X_2X_1$
ECC 番地	0000 0000 0000 $0P_3P_2P_1$

図　DAT 番地と ECC 番地の内容

冗長ビットを求める手順を次に示す。

［手順］

① X_1～X_4 を 1 ビットずつ取り出して，X1～X4 番地の最下位ビットに格納する。

② P_1～P_3 を計算し，1 ビットずつ GR1～GR3 の対応するビット位置に格納する。

③ P_1～P_3 を合成して ECC 番地に格納する。

［プログラム］

行番号	ラベル	命令	オペランド	コメント
100	HUMM	START		
110		RPUSH		
120		LD	GR1,=1	
130		AND	GR1,DAT	
140		ST	GR1,X1	; X1 番地に 1 ビットを抽出
150		LD	GR2,=2	
160		(1)		
170		SRL	GR2,1	
180		ST	GR2,X2	; X2 番地に 1 ビットを抽出
190		(2)		
200		AND	GR3,DAT	
210		SRL	GR3,2	
220		ST	GR3,X3	; X3 番地に 1 ビットを抽出
230		LD	GR4,=8	
240		AND	GR4,DAT	
250		(3)		
260		ST	GR4,X4	; X4 番地に 1 ビットを抽出
270		XOR	GR1,X3	
280		XOR	GR1,X4	; GR1 に P_1 を生成
290		LD	GR2,X1	
300		XOR	GR2,X2	; P_2 と P_3 の共通部分を生成
310		LD	GR3,GR2	; 共通部分を GR3 へコピー
320		(4)		
330		SLL	GR2,1	; GR2 に P_2 を生成
340		XOR	GR3,X3	
350		(5)		; GR3 に P_3 を生成
360		OR	GR3,GR2	
370		OR	GR3,GR1	
380		ST	GR3,ECC	; P_1～P_3 を合成
390		RPOP		
400		RET		
410	DAT	DC	#000D	
420	ECC	DS	1	
430	X1	DS	1	
440	X2	DS	1	
450	X3	DS	1	
460	X4	DS	1	
470		END		

＜設問１＞　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) の解答群

ア．`AND  GR1,DAT`　　イ．`AND  GR1,GR2`

ウ．`AND  GR2,DAT`　　エ．`AND  GR2,GR1`

(2) の解答群

ア．`LD  GR3,=1`　　イ．`LD  GR3,=2`

ウ．`LD  GR3,=3`　　エ．`LD  GR3,=4`

(3) の解答群

ア．`SRL  GR4,1`　　イ．`SRL  GR4,2`

ウ．`SRL  GR4,3`　　エ．`SRL  GR4,4`

(4) の解答群

ア．`XOR  GR2,X3`　　イ．`XOR  GR2,X4`

ウ．`XOR  GR3,X3`　　エ．`XOR  GR3,X4`

(5) の解答群

ア．`SLL  GR3,1`　　イ．`SLL  GR3,2`

ウ．`SLL  GR3,3`　　エ．`SLL  GR3,4`

＜設問２＞　次のハミング符号の自動訂正機能に関する記述中の［　　］に入れるべき適切なビット列を解答群から選べ。

1ビットの誤りが発生している4ビットのデータ「1101」と，誤りが発生する前のデータに対する3ビットの訂正符号（P_3, P_2, P_1）「100」がある。次の手順により，どのビットに誤りが発生したかを求める。

［手順］

① 4ビットのデータ「1101」の訂正符号を求めると「［　(6)　］」となる。

② ①で求めた訂正符号と，誤り発生前の訂正符号「100」の排他的論理和を求める。

③ ②で得られたビット列と，誤りが発生したビットの対応表を示す。
ただし，データ部分の１ビット誤りに限る。

表　冗長ビットと誤り発生ビットの対応表

②のビット列	誤りの発生したビット
111	X_1
110	X_2
101	X_3
011	X_4

したがって，今回「1101」の誤りが発生する前のデータは「［　(7)　］」である。

(6) の解答群

ア．**001**　　イ．**010**　　ウ．**011**　　エ．**100**

(7) の解答群

ア．**0101**　　イ．**1001**　　ウ．**1100**　　エ．**1111**

＜メモ欄＞

＜メモ欄＞