応用情報技術者試験
午後問題解説

平成27年春試験

●問 1（必須）

問題番号	出題分野	テーマ
問 1	情報セキュリティ	電子メールのセキュリティ

●問 2，問 3（2 問中 1 問選択）

問題番号	出題分野	テーマ
問 2	経営戦略	ブランド戦略
問 3	プログラミング	データ圧縮の前処理として用いられる Block-sorting

問2　　20点
その他　16点

●問 4～問 11（8 問中 4 問選択）

問題番号	出題分野	テーマ
問 4	システムアーキテクチャ	キャンペーンサイトの構築
問 5	ネットワーク	DHCP を利用したサーバの冗長化
問 6	データベース	アクセスログ監査システムの構築
問 7	組込みシステム開発	自動車用衝突被害軽減ブレーキシステム
問 8	情報システム開発	チケット販売システムの在庫調整機能の開発
問 9	プロジェクトマネジメント	プロジェクトの人的資源計画とコミュニケーション計画の策定及び実施
問 10	サービスマネジメント	情報資産の管理
問 11	システム監査	財務会計システムの運用の監査

問1　セキュリティ

設問1　本文中の［　a　］，［　e　］に入れる適切な名称をそれぞれ解答群の中から選び，記号で答えよ。

解答群

ア　OP25B　　イ　送信ドメイン認証　　ウ　フィルタリング

エ　フロー制御　　オ　迷惑メールボックス　　カ　ログ

T君　：メール送信時の内容チェックは，指摘事項の(イ)に対応し，メール受信時の内容チェックは，指摘事項の(カ)に対応します。メールサーバでのメール受信時の送信元メールアドレスが偽称されていないかのチェックは［　a　］と呼ばれ，送信元 IP アドレスを基にチェックする技術（SPF），又は受信メールの中の電子署名を基にチェックする技術（DKIM）を導入します。

S部長：類似の標的型攻撃メールが届いた宛先は，メールサーバの［　e　］から調査できるな。

設問2　本文中の［　b　］～［　d　］に入れる適切な字句を答えよ。

S部長：暗号化方式の変更について説明してくれないか。

T君　：暗号化方式の変更は，指摘事項の(ウ)と(エ)に対応します。現在のメールシステムでは，営業部でのメールの暗号化には，S/MIME を利用することになっています。メール宛先の［　b　］鍵を利用して暗号化する方式で，安全性は高いのですが，先方が［　b　］鍵をもっていなければ使えない方法なので，利用している顧客はごく一部です。

T君　：電子署名の添付は指摘事項の(ア)への対応です。従来，営業部員は個別に電子証明書と暗号鍵を与えられ，本人の電子署名の添付と，公開鍵基盤を導入している宛先へのメールの暗号化ができました。しかし，対象を全社員に広げるとなると，社員の電子証明書の運用コストが掛かってしまいます。そこで，社員の電子署名の添付を廃止し，メールサーバで，全メールに R 社の電子署名を添付して，送信することにします。電子署名はメールの［　c　］値を基に生成されるので，メールの［　d　］検知も可能になります。

設問 3　S 部長が本文中の下線①の指摘を行った理由を，35 字以内で述べよ。

て社外からもメールシステムを利用している。メールシステムが受信した電子メール（以下，メールという）は，メールサーバに保存される。社内 PC で開封したメールは，メールサーバから削除され，社内 PC に保存される。モバイル端末で開封したメールは，削除はされず，メールサーバに残される。

T 君　：当社ではメール受信のプロトコルとして，POP を利用してきました。新メールシステムでは，指摘事項(オ)に対応するために IMAP に変更し，社内 PC で開封したメールも含め，全ての受信メールが一定期間メールサーバに残るようにすることを考えています。

S 部長：なるほど。しかし，そうなると，①パスワードが流出した場合のリスクが高まることを認識しておく必要がある。特にモバイル端末の利用時には盗難なども考えられる。IMAP サーバでのモバイル端末の認証にはワンタイム

パスワードがあれば、
誰でも読める状態。

設問4　本文中の下線②の機能に該当するものを解答群の中から二つ選び，記号で答えよ。

解答群

ア　一時保管メールボックスに転送された受信メールの中から，受信者が必要なメールを取り出す機能

イ　会社や顧客の重要情報を含む送信メールは，フィルタリングの対象となるが，事前に承認されたメールについては宛先に転送されるようにする機能

ウ　フィルタリングによって阻止された全ての送信メールについて，タイトル（主題）だけを宛先に転送する機能

エ　フィルタリングの内容を，社員が設定する機能

S部長：表3に従うと，業務に必要なメールまでチェックによって阻止されてしまうことがある。②それらのメールに対応するための機能も加えるように。

表3　メールの内容チェックの詳細

	チェック項目	チェック後の対応
送信	会社の重要情報が含まれていないか。	問題があったメールは，チェックコメントを付けて，宛先には送信せずに送信元に返送する。
送信	顧客の重要情報が含まれていないか。	
受信	送信元メールアドレスが，迷惑メール送信者としてブラックリストに掲載されていないか。	メールを迷惑メールボックスに転送し，10 日後に自動削除する。
受信	送信元メールアドレスは，偽称されていないか。	本文を削除して宛先メールボックスに転送する。元のメールは一時保管メールボックスに転送し，10 日後に自動削除する。
受信	実行形式や不審なファイルが添付されていないか。	添付ファイルを削除して宛先メールボックスに転送する。元のメールは一時保管メールボックスに転送し，10 日後に自動削除する。
受信	ウイルスに感染していないか。	ウイルスを駆除した上で，宛先メールボックスに転送する。
受信	本文が HTML で記述されていないか。	注意を促すコメントを付けて，宛先メールボックスに転送する。
受信	本文中に URL が記載されていないか。	

宛先に送れない場合がある。

誤検知したメールが
一時保管メールボックスに
転送される場合がある。

設問5　本文中の下線③で不審なメールが届いた全ての受信者に指示すべき事項は何か。15字以内で述べよ。

S部長：標的型攻撃メールには，どのような対応をするのか。

T君　：標的型攻撃メールは，メールシステムでの対応には限界があるので，運用での対応も必要になると考えています。例えば，標的となった組織の複数のメールアドレスに届くことが多いので，一斉に，組織的に対応する必要があります。一人でも標的型攻撃メールと疑われるメールを受信した場合，メールシステムの管理者は，類似のメールが届いていないかを調査し，③不審なメールが届いた全ての受信者に対応を指示します。その後，受信者が添付ファイルを開けていないことやURLをクリックしていないことなどを管理者が確認します。

問3　プログラミング

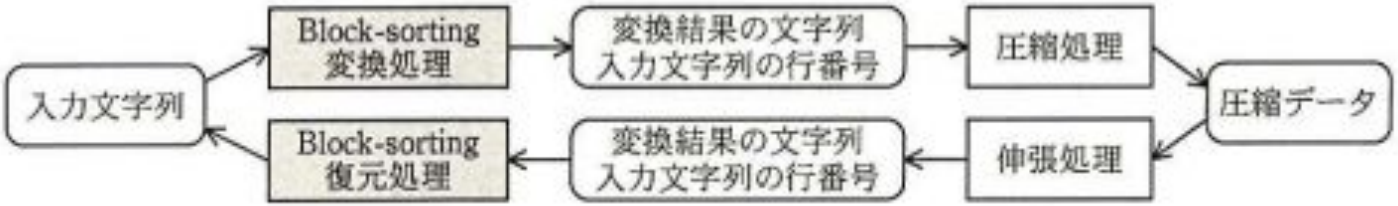
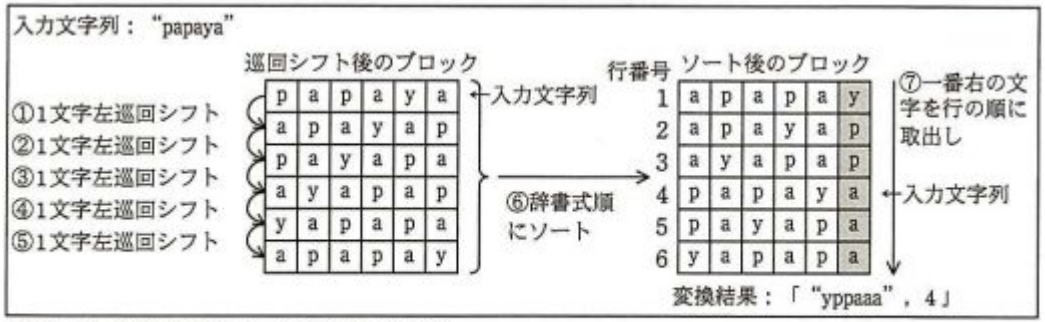


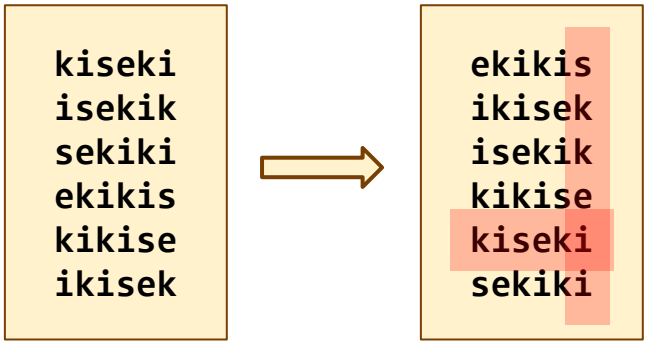
図1　データ圧縮における Block-sorting の使用方法

設問 1　文字列“kiseki”に対して Block-sorting を適用して変換した結果を答えよ。変換結果は図 2 の記法に合わせて記述すること。

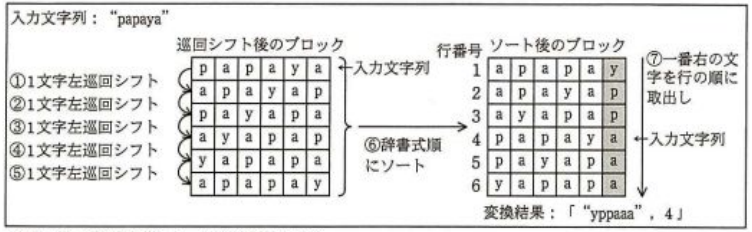


注記　①～⑦は処理順，1～6 は行番号を示す。

図 2　Block-sorting の変換処理



設問2　図3中の［　ア　］～［　ウ　］に入れる適切な字句を答えよ。



注記　①～⑦は処理順，1～6は行番号を示す。

図2　Block-sorting の変換処理

名称	種類	内容
EncodeArray[n]	配列	巡回シフト後のブロックを格納する。ブロックの1行を文字列として，配列の一つの要素に格納する。配列の添字は1から始まる。 例 "papaya" "apayap" "payapa" "ayapap" "yapapa" "apapay"
DecodeArray[2][n]	配列	復元用の文字と添字の組を格納する。配列の添字は1から始まる。 例 "y" "p" "p" "a" "a" "a" / 1 2 3 4 5 6
sort1(Array[])	関数	1次元配列 Array[ ]の要素を辞書式順にソートする。
sort2(Array[][])	関数	2次元配列 Array[ ][ ]を，Array[1]の要素をキーにしてソートする。 例 "y" "p" "p" "a" "a" "a" / 1 2 3 4 5 6 → "a" "a" "a" "p" "p" "y" / 4 5 6 2 3 1
rotation(String)	関数	文字列 String を1文字左に巡回シフトした結果を返す。
InputString	変数	入力文字列。この文字列の長さを"InputString の長さ"とする。他の文字列変数についても，長さを同様に表す。
BlockSortString	変数	変換結果の文字列。
Line	変数	ソート後のブロックでの入力文字列の行番号。
OutputString	変数	復元処理の出力文字列。

注記　nは入力文字列の長さを表す。

変換処理を実装した関数 encode のプログラムを図3に示す。

```
function encode(InputString)
  rString ← InputString
  for( i を［ ア ］から［ イ ］まで1ずつ増やす )
    EncodeArray[i] ← rString
    rString ← rotation(rString)
  endfor
  sort1(EncodeArray)
  BlockSortString を空文字列に初期化する
  for( k を［ ア ］から［ イ ］まで1ずつ増やす )
    BlockSortString の末尾に EncodeArray[k]の末尾の1文字を追加する
    if( ［ ウ ］ )
      Line ← k
    endif
  endfor
endfunction
```

巡回シフトしつつ格納

ソート

末尾の一行を取出し

入力文字列と同じ内容が格納されている場所

手順	処理	内容
1	変換結果の文字列に対して，各文字に 1 から順に添字を付ける。	“yppaaa” → “y(1),p(2),p(3),a(4),a(5),a(6)”
2	文字をソートする。同じ文字の場合は添字の順に並べる。	“y(1),p(2),p(3),a(4),a(5),a(6)” → “a(4),a(5),a(6),p(2),p(3),y(1)”
3	手順 2 でソートした文字を次の手順で並べる。 ・変換結果の行番号“4”から，ソート後の文字列“a(4),a(5),a(6),p(2),p(3),y(1)”の 4 番目の要素“p(2)”を取り出して並べる。 ・“p(2)”の添字が 2 であることから，2 番目の要素“a(5)”を取り出して並べる。 ・“a(5)”の添字が 5 であることから 5 番目の要素の“p(3)”を取り出して並べる。以降，並べた要素の個数が変換結果の文字列の長さと同じになるまで，要素を取り出して並べることを繰り返す。	→ “p(2)” → “p(2),a(5)” → “p(2),a(5),p(3)” → “p(2),a(5),p(3),a(6)” → “p(2),a(5),p(3),a(6),y(1)” → “p(2),a(5),p(3),a(6),y(1),a(4)”
4	手順 3 の結果から添字を取り除く。	“p(2),a(5),p(3),a(6),y(1),a(4)” → “papaya”

名称	種類	内容
EncodeArray[n]	配列	巡回シフト後のブロックを格納する。ブロックの 1 行を文字列として，配列の一つの要素に格納する。配列の添字は 1 から始まる。 例 “papaya” “apayap” “payapa” “ayapap” “yapapa” “apapay”
DecodeArray[2][n]	配列	復元用の文字と添字の組を格納する。配列の添字は 1 から始まる。 例 “y” “p” “p” “a” “a” “a” 1 2 3 4 5 6
sort1(Array[ ])	関数	1 次元配列 Array[ ]の要素を辞書式順にソートする。
sort2(Array[ ][ ])	関数	2 次元配列 Array[ ][ ]を，Array[1]の要素をキーにしてソートする。 例 “y” “p” “p” “a” “a” “a” → “a” “a” “a” “p” “p” “y” 1 2 3 4 5 6 → 4 5 6 2 3 1
rotation(String)	関数	文字列 String を 1 文字左に巡回シフトした結果を返す。
InputString	変数	入力文字列。この文字列の長さを“InputString の長さ”とする。他の文字列変数についても，長さを同様に表す。
BlockSortString	変数	変換結果の文字列。
Line	変数	ソート後のブロックでの入力文字列の行番号。
OutputString	変数	復元処理の出力文字列。

注記 n は入力文字列の長さを表す。

設問 3 〔復元処理関数 decode〕について，(1)，(2)に答えよ。

(1) 図 4 中の［ エ ］～［ カ ］に入れる適切な字句を答えよ。

(2) BlockSortString の長さが p のとき，図 4 中の下線(α)の処理の実行回数を答えよ。

復元処理を実装した関数 decode のプログラムを図 4 に示す。

```
function decode(BlockSortString, Line)
  for( i を 1 から BlockSortString の長さまで 1 ずつ増やす )
    DecodeArray[1][i] ← BlockSortString の i 文字目
    DecodeArray[2][i] ← i
  endfor
  sort2(DecodeArray)
  OutputString を空文字列に初期化する
  OutputString の末尾に［ エ ］に格納されている 1 文字を追加する
  n ← ［ オ ］
  while( ［ カ ］ )
    OutputString の末尾に DecodeArray[1][n]に格納されている 1 文字を追加する ◀ (α)
    n ← DecodeArray[2][n]
  endwhile
endfunction
```

各文字に添え字をつける

ソート

“papaya” の例だと Line = 4なので p と添字 2 を取出す

変換結果の文字列と同じ長さになるまで処理する

設問4 本文中の下線(β)について，ソートアルゴリズムを見直す場合とソートキーを見直す場合のそれぞれについて，どのように見直せばよいかを 30 字以内で述べよ。

〔関数 sort2(Array[ ][ ])の実装〕

関数 decode の処理時間は，使用する関数 sort2(Array[ ][ ])の計算量に大きく依存する。処理時間を短くするためには，sort2(Array[ ][ ])の内部で計算量が少ないソートのアルゴリズムを使用して実装する必要がある。

処理時間の違いを確認するために複数のソートアルゴリズムを使用して関数 sort2(Array[ ][ ])を実装したところ，Array[1]の要素をキーにしてクイックソート（不安定なソート）を使用した場合には復元処理の結果が入力文字列と一致しなかった。

この場合，sort2(Array[ ][ ])が表 1 の手順 2 を正しく実装できていないので，(β)ソートアルゴリズム，ソートキーのいずれかを見直す必要がある。