問2　ハッシュ法と排他制御に関する次の記述を読んで，設問1～3に答えよ。

T社は，ソフトウェア開発を行う会社である。現在，LAN環境で利用するクライアントサーバシステム（以下，本システムという）を開発中である。

本システムは，クライアントPC上で画面にデータを表示する画面プログラムと，画面プログラムが使用するデータをサーバ上で管理するデータ管理プログラムから構成される。

〔データ構造〕

配列arrayは，クライアントPCで使用するデータinfoを格納する配列であり，配列の添え字は1～Nである。infoのデータ型は構造体INFOである。表1に構造体INFOのメンバを示す。構造体メンバの初期値は，全て0（未使用を意味する）を設定する。

表1　構造体INFOのメンバ

メンバ	説明
key	本システムで配列array中のinfoを一意に識別する1～999999999の整数
data	本システムで処理するデータ

注記　メンバの参照は，info.keyのように構造体名の後にピリオドとメンバ名を記述する。

ハッシュ関数Hash(key)は，keyを基にデータの格納位置を算出して，戻り値として戻す。格納位置は1～Nの整数となる。関数Hash(key)が，異なるkeyから同じ格納位置を算出することを，シノニムの発生という。この影響で，格納位置の配列要素が既に使用されていて，データを格納できないことがある。この場合は，配列arrayを格納位置の次から順次検索し，最初に見つかった未使用の配列要素にデータを格納する。配列arrayの最後に到達しても未使用の配列要素がない場合には，配列arrayの先頭に戻り，未使用の配列要素の検索を続ける。

〔データ管理プログラム〕

図1のデータ格納関数Set( )，図2のデータ取得関数Get( )，図3のデータ削除関数Delete( )をデータ管理プログラムと呼ぶ。

関数Get( )は，データ取得に成功すると，取得したデータを引数infoに格納する。関数Set( )と関数Get( )は戻り値として格納位置を戻す。処理結果が正しくない場合は，

戻り値として0を戻す。

```
function Set(info)
    count ← 0
    idx ← Hash(info.key)
    while ((array[idx].key は1以上) かつ (array[idx].key は 999999999 以下)
                                    かつ ( [ ア ] ))
        count ← count + 1
        idx ← idx + 1
        if ( [ イ ] )              // 配列の最後を検出
            idx ← 1                // 配列の先頭に戻る
        endif
    endwhile
    if (count は N より小さい)
        array[idx] ← info          // info を格納
    else
        idx ← 0                    // オーバフロー発生
    endif
    return idx
endfunction
```

図1　データ格納関数 Set( )

```
function Get(key, info)
    count ← 0
    idx ← [ ウ ]
    while ((array[idx].key は1以上) かつ (array[idx].key は 999999999 以下)
                                    かつ ( [ ア ] ) かつ ( [ エ ] ))
        count ← count + 1
        idx ← idx + 1
        if ( [ イ ] )              // 配列の最後を検出
            idx ← 1                // 配列の先頭に戻る
        endif
    endwhile
    if (array[idx].key は key と等しい)
        info ← array[idx]          // データを info に格納
    else
        idx ← 0                    // データの取得失敗
    endif
    return idx
endfunction
```

図2　データ取得関数 Get( )

```
function Delete(key)
    idx ← Get(key, info)
    if (idx は 0 と等しくない)
        array[idx].key ← 0     // 配列要素を未使用に設定する
    endif
endfunction
```

図 3　データ削除関数 Delete( )

〔画面プログラム〕

ユーザは，画面プログラムを使用して，info の作成，検索，編集，削除を行う。

画面プログラムから配列 array へのアクセスにはデータ管理プログラムを使用する。

複数のクライアント PC から同時に配列 array にアクセスできるので，画面プログラムから配列 array へのアクセスには排他制御が必要である。そこで，バイナリセマフォの確保関数 Lock( )，解放関数 Unlock( )を使用した占有ロックを用いて排他制御を実現する。

図 4 は画面プログラムの一部である。ユーザが画面から入力した key と data が配列 array 内に存在しない場合は，データを配列 array に格納している。

```
……  省略  ……
    idx ← Get(key, info)      // データが配列 array に格納済か確認
    Lock()
    if (idx は 0 と等しい)      // データ未格納の場合
        info.key ← key
        info.data ← data
        idx ← Set(info)       // データ格納
    endif
    Unlock()
……  省略  ……
```

図 4　画面プログラムの一部

コーディングを完了したプログラムのテストを実施したところ，次のような障害が発生した。

①あるデータを削除すると，別のデータの取得に失敗した。削除するデータと取得できなくなるデータには関連があり，再現方法は容易に分かった。プログラムを修正して障害は解決した。

②複数のクライアント PC から図 4 の画面プログラムの操作を同時に行った場合に，同じ key をもつデータが重複して配列 array に格納されてしまった。この障害は，再現頻度が低く，原因究明に時間が掛かった。この障害についてもプログラムを修正して障害は解決した。

設問 1　図 1 及び図 2 中の［　ア　］～［　エ　］に入れる適切な字句を答えよ。

設問 2　本文中の下線①の障害について，(1)，(2)に答えよ。

(1)　障害の原因を 40 字以内で述べよ。

(2)　データ管理プログラムの修正の組合せとして適切な文章を解答群の中から二つ選び，記号で答えよ。

解答群

ア　関数 Delete()中の“array[idx].key ← 0”の 0 を－1 に変更する。

イ　関数 Get( )中の if 文の条件“array[idx].key は key と等しい”を削除して，無条件にデータを取得する。

ウ　関数 Get( )中の while 文の条件“(array[idx].key は 1 以上）かつ (array[idx].key は 999999999 以下)”を“(array[idx].key は 0 以外)”に変更する。

エ　関数 Set( )中でオーバフローを検出した場合は，配列 array を動的に拡張してデータを格納する。

オ　関数 Set( )中の if 文の条件“count は N より小さい”を削除して，無条件にデータを格納する。

カ　関数 Set()中の while 文の条件“array[idx].key は 1 以上”を削除する。

設問 3　本文中の下線②の障害について，原因を 25 字以内で述べよ。