次の問８は必須問題です。必ず解答してください。

問８　次のプログラムの説明及びプログラムを読んで，設問１，２に答えよ。

携帯端末上で稼働する簡易メモ帳の機能のうち，メモの編集処理（メモの追加・削除・変更・移動）を行う部分のプログラムである。図１は，簡易メモ帳に４件のメモ“Aoki”，“Imai”，“Uno”及び“Endo”を登録した場合の表示例である。

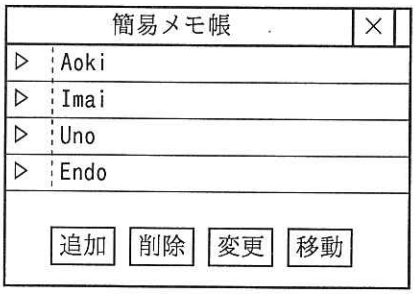


図１　簡易メモ帳の表示例

〔プログラムの説明〕

(1)　メモは，画面に表示可能な１バイトで表現できる文字から成る文字列である。各メモは，文字列の前に，文字列の長さ（0～255）を１バイトの符号なし２進整数の形式で付け加えて格納する。例えば，メモ“Hello!”は，次の形式で格納する（以下，文字列の長さは10進数で表記し，その値に下線を付けて表す）。

| 6 | H | e | l | l | o | ! |
|---|---|---|---|---|---|---|

(2)　メモの格納と管理のために，２個の配列 Memo[]，Data[]と，４個の変数 MemoCnt，MemoMax，DataLen，DataMax を使用する。

各メモは，配列 Data[] の先頭から順に，１要素に１バイトずつ (1) で示した形式で格納し，その格納位置の情報を配列 Memo[] に設定して管理する。

MemoMax は格納できるメモの最大件数（配列 Memo[] の要素数），MemoCnt は現在格納されているメモの件数である。

DataMax は格納できる最大文字数（配列 Data[] の要素数），DataLen は現在格納されている文字数である（文字数には，文字列の長さの情報を含む）。

(3) 簡易メモ帳の画面には，配列 Memo[] の要素番号の昇順に，それが指すメモを取り出して，メモを表示する。図 1 の表示例は，図 3（後出）の状態に対応している。

(4) メモの編集処理を行うための関数の概要は，次の①～⑤のとおりである。これらの関数が呼ばれるとき，引数の内容や配列の空き状態などは事前に検査済みで，正しく実行できるものとする。

なお，以降の図に示す実行例では，MemoMax = 5, DataMax = 25 としている。

① 関数: resetMemo()

全てのメモを消去する。MemoCnt と DataLen に 0 を設定することによって，Memo[] と Data[] の全要素を“空き”の状態にする。

resetMemo() を実行した後の配列・変数の状態を，図 2 に示す。以降の図で，網掛け部分 ▨ は，その配列要素が“空き”であることを表す。

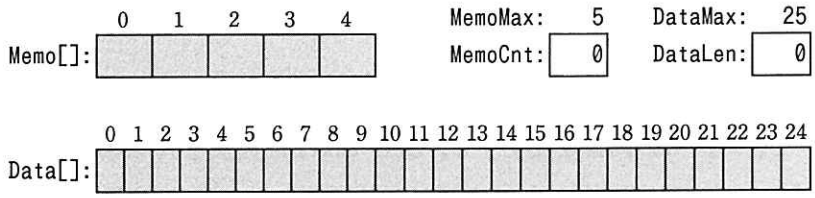


図 2　関数 resetMemo 実行後の配列・変数の状態

② 関数: addMemo(整数型: textLen, 文字列型: text)

1 件のメモを追加する。長さ textLen の文字列 text を Data[] の最初の空き要素以降に格納し，その格納位置の情報を Memo[] に設定する。図 2 の状態から，

```
addMemo(4, "Aoki")
addMemo(4, "Imai")
addMemo(3, "Uno")
addMemo(4, "Endo")
```

をこの順に実行した後の配列・変数の状態を，図 3 に示す。

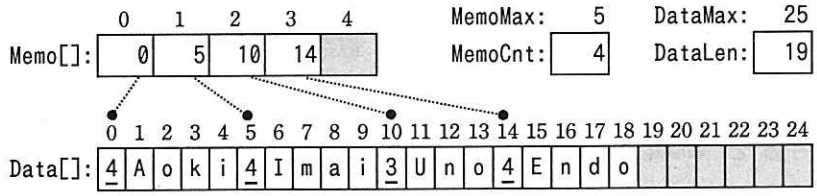


図 3　関数 addMemo（4 件）実行後の配列・変数の状態

E 後

③　関数: deleteMemo(整数型: pos)

1件のメモを削除する。Memo[] の要素番号 pos＋1 以降の内容をそれぞれ一つ前の要素に移し，MemoCnt から 1 を減じることによって，Memo[pos] が指すメモを削除する（表示の対象から除く）。Data[] 中の参照されなくなったメモは，そのまま残す。図 3 の状態から，deleteMemo(0) を実行した後の配列・変数の状態を，図 4 に示す。以降の図で，斜線部分 ▨ は，参照されなくなったメモであることを表す。

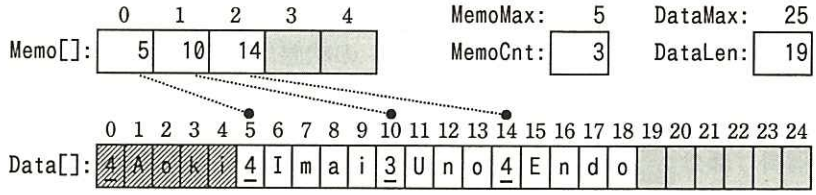


図 4　関数 deleteMemo 実行後の配列・変数の状態

④　関数: changeMemo(整数型: pos, 整数型: textLen, 文字列型: text)

1件のメモの内容を変更する。長さ textLen の文字列 text を Data[] の最初の空き要素以降に格納し，その格納位置の情報を Memo[pos] に設定することによって，Memo[pos] が指すメモの内容を変更する。Data[] 中の参照されなくなったメモは，そのまま残す。図 4 の状態から changeMemo(2, 3, "Abe") を実行した後の配列・変数の状態を，図 5 に示す。

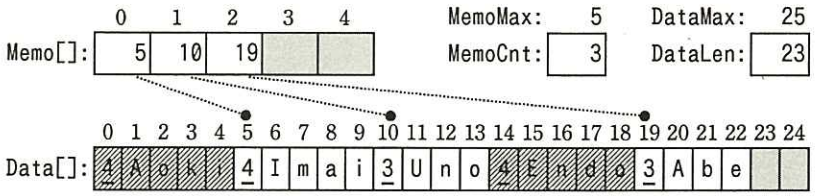


図 5　関数 changeMemo 実行後の配列・変数の状態

⑤　関数: moveMemo(整数型: fromPos, 整数型: toPos)

1 件のメモを移動する。Memo[] の要素の並び順を変えて，Memo[fromPos] の内容を Memo[toPos] の位置に移動する。fromPos ＜ toPos の場合は，Memo[fromPos] の値を取り出し，Memo[fromPos＋1] ～ Memo[toPos] の内容を前方に 1 要素分ずらし，取り出した値を Memo[toPos] に設定するという操作を行

う。fromPos ＞ toPos の場合も，これと同様の操作を行う。図 5 の状態から moveMemo(2, 0) を実行した後の配列・変数の状態を，図 6 に示す。

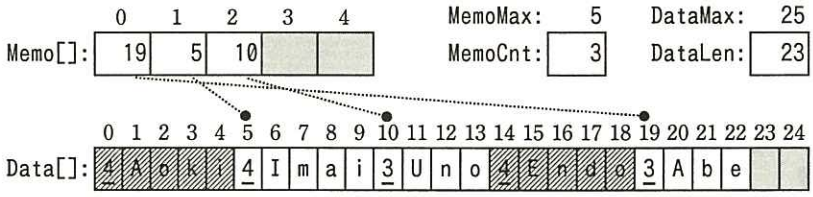


図 6　関数 moveMemo 実行後の配列・変数の状態

〔プログラム〕

○大域　整数型: MemoCnt, MemoMax, Memo[MemoMax]
○大域　整数型: DataLen, DataMax
○大域　8 ビット論理型:　Data[DataMax]

○関数: resetMemo()

- MemoCnt ← 0
- DataLen ← 0

○関数: addMemo(整数型: textLen, 文字列型: text)
　○整数型: i

- Memo[MemoCnt] ← [　　a　　]
- MemoCnt ← MemoCnt ＋ 1
- Data[DataLen] ← textLen
- DataLen ← DataLen ＋ 1
- i: 0, i ＜ textLen, 1
  - Data[DataLen ＋ i] ← text[i]
- DataLen ← [　　b　　]

○関数: deleteMemo(整数型: pos)
　○整数型: i

- i ← [　　c　　]
- i ＜ MemoCnt
  - Memo[i － 1] ← Memo[i]
  - i ← i ＋ 1
- MemoCnt ← MemoCnt － 1

```
○関数: changeMemo(整数型: pos, 整数型: textLen, 文字列型: text)
  ○整数型: i

  ・Memo[pos] ← DataLen
  ・Data[DataLen] ← textLen
  ・DataLen ← DataLen + 1
  ■ i: 0, i < textLen, 1
  |  ・Data[DataLen + i] ← text[i]
  ■
  ・DataLen ← [    b    ]

○関数: moveMemo(整数型: fromPos, 整数型: toPos)
  ○整数型: i, m

  ・m ← Memo[fromPos]
  ▲ fromPos < toPos
  |  ■ i: fromPos, i ≦ toPos − 1, 1
  |  |  ・Memo[i] ← Memo[i + 1]
  |  ■
  ▼
  ▲ fromPos > toPos
  |  ■ i: [    d    ]
  |  |  ・Memo[i] ← Memo[i − 1]
  |  ■
  ▼
  ・Memo[toPos] ← m
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a，bに関する解答群

| | |
|---|---|
| ア　DataLen | イ　DataLen + 1 |
| ウ　DataLen + textLen | エ　DataLen + textLen + 1 |
| オ　textLen | カ　textLen + 1 |

cに関する解答群

| | |
|---|---|
| ア　MemoCnt − pos | イ　pos − 1 |
| ウ　pos | エ　pos + 1 |

dに関する解答群

| | |
|---|---|
| ア　fromPos, i ≧ toPos − 1, −1 | イ　fromPos, i ≧ toPos + 1, −1 |
| ウ　toPos, i ≦ fromPos − 1, 1 | エ　toPos, i ≦ fromPos + 1, 1 |

設問2　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

このメモの管理方法では，削除されたメモや変更前のメモは，Data[] 中に参照されない状態で残っている。その結果，DataLen の値は一方的に増加し，やがて Data[] 中の空き要素が枯渇する。次に示す関数 clearGarbage() は，Data[] 中の参照されなくなったメモを取り除き，空き要素を増やすための関数である。

```
○関数: clearGarbage()
  ○整数型： d, i, m
  ○８ビット論理型： temp[DataMax]

  ・DataLen ← 0
  MemoCnt = 0
  ・return
  m: 0, m < MemoCnt, 1
  ・d ← Memo[m]
  ・Memo[m] ← DataLen
    i: 0, i ≦ Data[d], 1
    ・temp[DataLen] ← Data[d + i]
    ・DataLen ← DataLen + 1
  d: 0, d < DataLen, 1
  ・Data[d] ← temp[d]
```

プログラムの配列・変数が図 6 に示す状態のときに，clearGarbage() を実行すると，実行が終了した時点で，Memo[1] の値は［　e　］，Memo[2] の値は［　f　］，DataLen の値は［　g　］となる。

解答群

| | | | | |
|---|---|---|---|---|
| ア　0 | イ　3 | ウ　4 | エ　5 | オ　7 |
| カ　9 | キ　10 | ク　12 | ケ　13 | コ　23 |