アルゴリズムとデータ構造　授業中練習問題9

次のプログラムは「循環・重連結リストの実現例」である．このプログラムに関して，以下の問いに答えなさい．さらに，このプログラムを入力し，自分のパソコンでコンパイル，実行できることを確認してください．なお，プログラムの日本語部分は，英語，ローマ字に変更してかまいません．

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MEMBER_NO       1       /* 番号を表す定数値 */
#define MEMBER_NAME     2       /* 氏名を表す定数値  */

/*--- 会員データ ---*/
typedef struct {
  int  no;        /* 番号 */
  char name[20];  /* 氏名 */
} Member;
/*--- ノード ---*/
typedef struct __node {
  Member        data;   /* データ */
  struct __node *prev;  /* 先行ノードへのポインタ */
  struct __node *next;  /* 後続ノードへのポインタ */
} Dnode;
/*--- 循環・重連結リスト ---*/
typedef struct {
  Dnode *head;  /* 先頭ダミーノードへのポインタ */
  Dnode *crnt;  /* 着目ノードへのポインタ */
} Dlist;
/*--- 会員の番号の比較関数 ---*/
int MemberNoCmp(const Member *x, const Member *y){
  return x->no < y->no ? -1 : x->no > y->no ? 1 : 0;
}
/*--- 会員の氏名の比較関数 ---*/
int MemberNameCmp(const Member *x, const Member *y){
  return strcmp(x->name, y->name);
}
/*--- 会員データ（番号と氏名）の表示（改行なし）---*/
void PrintMember(const Member *x){
  printf("%d %s", x->no, x->name);
}
/*--- 会員データ（番号と氏名）の表示（改行あり）---*/
void PrintLnMember(const Member *x){
  printf("%d %s¥n", x->no, x->name);
}
/*--- 会員データ（番号と氏名）の読込み ---*/
Member ScanMember(const char *message, int sw){
  Member temp;

  printf("%s するデータを入力してください。¥n", message);

  if (sw & MEMBER_NO)   { printf("番号："); scanf("%d", &temp.no); }
```

```
  if (sw & MEMBER_NAME) { printf("氏名：");  scanf("%s", temp.name); }

  return temp;
}
/*--- 一つのノードを動的に生成 ---*/
static Dnode *AllocDnode(void){
  return calloc(1, sizeof(Dnode));
}
/*--- ノードの各メンバに値を設定 ----*/
static void SetDnode(Dnode *n, const Member *x,
                     const Dnode *prev, const Dnode *next){
  n->data = *x;   /* データ */
  n->prev = (Dnode *) prev; /* 先行ノードへのポインタ */
  n->next = (Dnode *) next; /* 後続ノードへのポインタ */
}
/*--- リストは空か ---*/
static int IsEmpty(const Dlist *list){
  return list->head->next == list->head;
}
/*--- リストを初期化 ---*/
void Initialize(Dlist *list){
  Dnode *dummyNode = AllocDnode();  /* ダミーノードを生成 */
  list->head = list->crnt = dummyNode;
  dummyNode->prev = dummyNode->next = dummyNode;
}
/*--- 着目ノードのデータを表示 ---*/
void PrintCurrent(const Dlist *list){
  if (IsEmpty(list))
    printf("着目要素はありません。");
  else
    PrintMember(&list->crnt->data);
}
/*--- 着目ノードのデータを表示（改行付き）---*/
void PrintLnCurrent(const Dlist *list){
  PrintCurrent(list);
  putchar('¥n');
}
/*--- 関数 compare によって x と一致すると判定されるノードを探索 ---*/
Dnode *Search(Dlist *list, const Member *x,
              int compare(const Member *x, const Member *y)){
  Dnode *ptr = list->head->next;

  while (ptr != list->head) {
    if (compare(&ptr->data, x) == 0) {
      list->crnt = ptr;
      return ptr;  /* 探索成功 */
    }
    ptr = ptr->next;
  }
  return NULL;     /* 探索失敗 */
}
/*--- 全ノードのデータをリスト順に表示 ---*/
```

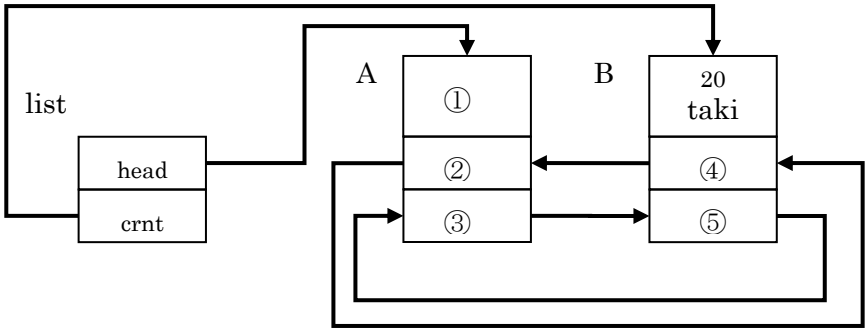
```
100 void Print(const Dlist *list){
      if (IsEmpty(list))
        puts("ノードがありません。");
      else {
        Dnode *ptr = list->head->next;
105
        puts("【一覧表】");
        while (ptr != list->head) {
          PrintLnMember(&ptr->data);
          ptr = ptr->next;  /* 後続ノードに着目 */
110     }
      }
    }
    /*--- p が指すノードの直後にノードを挿入 ---*/
    void InsertAfter(Dlist *list, Dnode *p, const Member *x){
115   Dnode *ptr = AllocDnode();
      Dnode *nxt = p->next;

      p->next = p->next->prev = ptr;
      SetDnode(ptr, x, p, nxt);
120   list->crnt = ptr;     /* 挿入したノードに着目 */
    }
    /*--- 先頭にノードを挿入 ---*/
    void InsertFront(Dlist *list, const Member *x){
      InsertAfter(list, list->head, x);
125 }
    /*--- 末尾にノードを挿入 ---*/void
    InsertRear(Dlist *list, const Member *x){
      InsertAfter(list, list->head->prev, x);
    }
130 /*--- p が指すノードを削除 ---*/
    void Remove(Dlist *list, Dnode *p){
      p->prev->next = p->next;
      p->next->prev = p->prev;
      list->crnt = p->prev;  /* 削除したノードの先行ノードに着目 */
135   free(p);
      if (list->crnt == list->head)
        list->crnt = list->head->next;
    }
    /*--- 先頭ノードを削除 ---*/
140 void RemoveFront(Dlist *list){
      if (!IsEmpty(list))
        Remove(list, list->head->next);
    }
    /*--- 末尾ノードを削除 ---*/
145 void RemoveRear(Dlist *list){
      if (!IsEmpty(list))
        Remove(list, list->head->prev);
    }
    /*--- 着目ノードを削除 ---*/
150 void RemoveCurrent(Dlist *list){
      if (list->crnt != list->head)
```

```
    Remove(list, list->crnt);
}
/*--- 全ノードを削除 ---*/
void Clear(Dlist *list){
  while (!IsEmpty(list)) /* 空になるまで */
    RemoveFront(list);   /* 先頭ノードを削除 */
}
/*--- 循環・重連結リストを後始末 ---*/
void Terminate(Dlist *list){
  Clear(list);      /* 全ノードを削除 */
  free(list->head); /* ダミーノードを削除 */
}
/*--- メニュー ---*/
typedef enum {
  TERMINATE, INS_FRONT, INS_REAR,  RMV_FRONT, RMV_REAR, PRINT_CRNT,
  RMV_CRNT,  SRCH_NO,   SRCH_NAME, PRINT_ALL, CLEAR
} Menu;
/*--- メニュー選択 ---*/
Menu SelectMenu(void){
  int  i, ch;
  char *mstring[] = {
    "先頭にノードを挿入", "末尾にノードを挿入", "先頭のノードを削除",
    "末尾のノードを削除", "着目ノードを表示",   "着目ノードを削除",
    "番号で探索", "氏名で探索", "全ノードを表示", "全ノードを削除"
  };

  do {
    for (i = TERMINATE; i < CLEAR; i++) {
      printf("(%2d) %-18.18s  ", i + 1, mstring[i]);
      if ((i % 3) == 2)
        putchar('¥n');
    }
    printf("( 0) 終了 :");
    scanf("%d", &ch);
  } while (ch < TERMINATE || ch > CLEAR);

  return (Menu)ch;
}
/*--- メイン ---*/
int main(void){
  Menu menu;
  Dlist list;

  Initialize(&list); /* 循環・重連結リストの初期化 */

  do {
    int n;
    Member x;
    Member *ptr;

    switch (menu = SelectMenu()) {
    case INS_FRONT :/* 先頭にノードを挿入 */
```

```
        x = ScanMember("先頭に挿入", MEMBER_NO | MEMBER_NAME);
205     InsertFront(&list, &x);
        break;
      case INS_REAR :/* 末尾にノードを挿入 */
        x = ScanMember("末尾に挿入", MEMBER_NO | MEMBER_NAME);
        InsertRear(&list, &x);
210     break;
      case RMV_FRONT : /* 先頭ノードを削除 */
        RemoveFront(&list);
        break;
      case RMV_REAR : /* 末尾ノードを削除 */
215     RemoveRear(&list);
        break;
      case PRINT_CRNT : /* 着目ノードのデータを表示 */
        PrintLnCurrent(&list);
        break;
220   case RMV_CRNT :   /* 着目ノードを削除 */
        RemoveCurrent(&list);
        break;
      case SRCH_NO :    /* 番号による探索 */
        x = ScanMember("探索", MEMBER_NO);
225     if (Search(&list, &x, MemberNoCmp) != NULL)
          PrintLnCurrent(&list);
        else
          puts("その番号のデータはありません。");
        break;
230   case SRCH_NAME :  /* 氏名による探索 */
        x = ScanMember("探索", MEMBER_NAME);
        if (Search(&list, &x, MemberNameCmp) != NULL)
          PrintLnCurrent(&list);
        else
235       puts("その名前のデータはありません。");
        break;
      case PRINT_ALL :   /* 全ノードのデータを表示 */
        Print(&list);
        break;
240   case CLEAR : /* 全ノードを削除 */
        Clear(&list);
        break;
      }
    } while (menu != TERMINATE);
245
    Terminate(&list); /* 循環・重連結リストの後始末 */

    return 0;
  }
250
```

1）このプログラムの動作直後に，「先頭にノードを挿入」を指示し，データ（番号：20，氏名： taki）を入力しました．このとき，次の問いに答えなさい．

(ア) この状態での循環・重連結リストの各ノードの関係を示す図を以下に示す．この図中の空欄①～⑤に入る最も適切な語句を答えなさい．



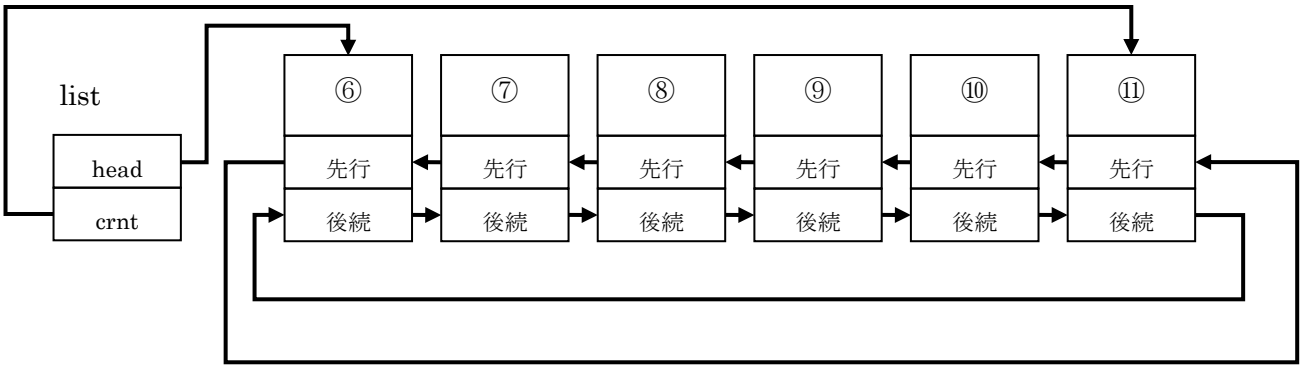
(イ) main 関数中の list.head が指すノードを(ア)の図の記号 A または B の記号で答えなさい．

(ウ) main 関数中の list.head ->next が指すノードを(ア)の図の記号 A または B の記号で答えない．

（エ）main 関数中の((list.head -> next) -> prevn)-> prev が指すノードを(ア)の図の記号 A または B の記号で答えない．

2）このプログラムの動作直後に，「末尾にノードを挿入」を連続し 5 回指示し，5 つのデータを[5, konishi]，[25，takahashi]，[20，ueda]，[25，kita]，[20, kita]の順番で入力しました．このとき，次の問に答えなさい．

（ア）この状態での循環・重連結リストの関係を示す図を以下に示す．この図の空欄⑥～⑪に入る最も適切な語句，またはノートのデータを答えなさい．



(イ) （ア)の状態で，さらに「氏名で探索」で[kita]を探索し，その後「着目ノードを削除」を指示しました．この時の削除されるノードを（ア)の図の⑥～⑪の記号で答えない．

(ウ) （ア)の状態で，さらに「番号で探索」で[20]を探索し，その後「着目ノードを削除」を指示しました．この時の削除されるノードを（ア)の図の⑥～⑪の記号で答えない．