プログラミング基礎 後期期末試験

全てのプログラムファイルの先頭行に、コメントとして自分の番号と名前を書くこと。

準備 この試験に利用する関数や構造体の宣言が、事前にファイルに用意されている。端末内で、以下のコマンドを実行してコピーしておくこと。

```
$ cp /usr/local/common/kogai/kiso201801.c . （←ここにピリオド）
```

1 ファイル「in.txt」に以下の様な記述が保存されているとする。

in.txt（読み込むファイル）

```
int a, b, c;
a = 30;
b = 50;
c = a + b;
```

このファイルを読み込み、**コピーした行数が最後の行に付加された**ファイル「out.txt」を生成するコピー処理をしたい。(以下参照)

out.txt（生成されたファイル）

```
int a, b, c;
a = 30;
b = 50;
c = a + b;
コピーした行数: 4
```

以下の補足を参考に、このようなコピー処理をするプログラムを作成しなさい。

- 1 行ずつコピーする処理をもとに作成できる
- 読み込む行をカウントする変数を用意しておき、1 行ずつ読み込む繰り返し処理において、読み込むごとにこの変数を値を 1 つ増やす
- 繰り返し処理が終了した後に、カウントした行数をファイルに書き込む

2 次のような「商品の単価と個数」を表す構造体型を考える。この構造体型の宣言は「kiso201801.c」に用意されている。

```
typedef struct Item {
    int tanka;  /* 単価 */
    int kosu;   /* 個数 */
} Item;
```

「この構造体型の配列（要素数は 5 個）に対して、各要素が保持している商品の情報を出力し、全ての総額を求める」関数 show_total() を作成しなさい。この関数のプロトタイプ宣言は以下のようになる。

```
void show_total(Item *p);
 /* 総額を求める変数を用意する */
 /* 仮引数 p が参照している配列に対する繰り返し処理を作る */
 /* 各要素に対して、単価，個数，単価×個数を出力し、 単価×個数を総額に加算する*/
 /* 繰り返し処理後に、全ての要素の総額を出力する */
 /* p が参照している配列の要素数は、5 個を前提に作成してよい */
```

main() での動作確認の例とその実行結果は以下のようになる。なお、ここで使っている main() は「kiso201801.c」に用意されている。

```
[main での処理]
  Item items1[NUM] = {
      {500, 3},
      {100, 2},
      {210, 1},
      {980, 4},
      {320, 3}
  };
  show_total(items1);
  items1[2].kosu = 3;
  items1[3].kosu = 1;
  show_total(items1);
[実行結果]
  500 円 * 3 個 = 1500 円       (←最初の配列に対して合計金額を出力している)
  100 円 * 2 個 = 200 円
  210 円 * 1 個 = 210 円
  980 円 * 4 個 = 3920 円
  320 円 * 3 個 = 960 円
  合計: 6790 円
  500 円 * 3 個 = 1500 円       (← 2 番目と 3 番目の要素の個数を変更した後の出力)
  100 円 * 2 個 = 200 円
  210 円 * 3 個 = 630 円
  980 円 * 1 個 = 980 円
  320 円 * 3 個 = 960 円
  合計: 4270 円
```

3 次のような「商品の単価と個数」を表すリスト用の構造体を考える。
前問で使用した構造体 Item とは異なることに注意すること。この構造体の宣言は「kiso201801.c」に用意されている。

```
typedef struct Item2 {
    int tanka;           /* 単価 */
    int kosu;            /* 個数 */
    struct Item2 *next;  /* 線形リスト用 */
} Item2;
```

この構造体 Item2 を使った線形リストに対して、「全ての要素の個数を増やす」関数 add_kosu() を作成しなさい。この関数のプロトタイプ宣言は以下のようになる。

```
void add_kosu(Item2 *start, int n);
  /* リストの全要素に対する繰り返し処理の中で、各要素のメンバ kosu に仮引数 n を加算する */
```

main() での動作確認の例とその実行結果は以下のようになる。なお、ここで使っている main() と出力用関数 show_itemlist() は「kiso201801.c」に用意されている。

```
[main での処理]
  Item2 it0, it1, it2, it3;
  it0.tanka = 300; it0.kosu = 2;
  it1.tanka = 800; it1.kosu = 3;
  it2.tanka = 400; it2.kosu = 0;
  /* リストを作る */
```

```
  it0.next = &it1;
  it1.next = &it2;
  it2.next = NULL;
  /* 動作を確認する */
  show_itemlist(&it0, "初期状態");
  add_kosu(&it0, 3);
  show_itemlist(&it0, "3 個追加");
  it3.tanka = 300; it3.kosu = 2;
  it2.next = &it3;
  it3.next = NULL;
  add_kosu(&it0, 2);
  show_itemlist(&it0, "さらに 2 個追加");
[実行結果]
  --- 初期状態 ---
  単価 300 円, 個数 2 個
  単価 800 円, 個数 3 個
  単価 400 円, 個数 0 個
  --- 3 個追加 ---
  単価 300 円, 個数 5 個
  単価 800 円, 個数 6 個
  単価 400 円, 個数 3 個
  --- さらに 2 個追加 ---
  単価 300 円, 個数 7 個
  単価 800 円, 個数 8 個
  単価 400 円, 個数 5 個
  単価 300 円, 個数 4 個
```

4 構造体 Item2 の要素が格納されているリストに対して、「引数で与えられた単価と個数が負の値の場合は、正の値にしてリストに追加する」関数 add_item() を作成しなさい。この関数のプロトタイプ宣言は以下のようになる。

```
void add_item(Item2 *p, int t, int k);
  /* まず、新しい要素の領域を確保してから・・・ */
  /*    ・t が負の値ならば、正の値にしてメンバ tanka に代入する */
  /*    ・k が負の値ならば、正の値にしてメンバ kosu に代入する */
  /* どちらの引数も正の値の場合は、そのままそれぞれのメンバに代入する */
  /* 新しい要素がリストの先頭に追加されるように next をつなぐ */
```

main() での動作確認の例とその実行結果は以下のようになる。なお、ここで使っている main() と出力用関数 show_itemlist() は「kiso201801.c」に用意されている。

```
[main での処理]
  Item2 head;
  head.tanka = 0; head.kosu = 0; head.next = NULL;
  show_itemlist(&head, "head (1)");
  add_item(&head, 2000, 3);
  show_itemlist(&head, "head (2)");
  add_item(&head, 4000, -5);
  show_itemlist(&head, "head (3)");
  add_item(&head, -1000, 2);
  show_itemlist(&head, "head (4)");
[実行結果]
  --- head (1) ---
  単価 0 円, 個数 0 個
```

```
--- head (2) ---          (←要素を 1 つ追加した)
単価 0 円, 個数 0 個
単価 2000 円, 個数 3 個
--- head (3) ---          (←個数が負の値の場合)
単価 0 円, 個数 0 個
単価 4000 円, 個数 5 個
単価 2000 円, 個数 3 個
--- head (4) ---          (←単価が負の値の場合)
単価 0 円, 個数 0 個
単価 1000 円, 個数 2 個
単価 4000 円, 個数 5 個
単価 2000 円, 個数 3 個
```

(各 25 点)

問題はここまで

定期試験の実施について

試験中に使用できるもの

- 筆記用具（メモ用紙は必要な人に配布）
- 演習室のコンピューター一台（一つの机に一人の配置で、座る場所はどこでもよい）

試験中に参照できるもの

- 自分のホームディレクトリ（ホームフォルダ）以下に保存されているファイル
  (定期試験では紙媒体のものは参照不可)

* **上記以外の情報を参照することは不正行為とする**
  (例：USB で接続された機器に保存されているファイルの参照など)

* 試験中は、演習室外へのネットワークアクセスは遮断される

答案の提出

- 提出する全てのプログラムファイルの先頭行に、**自分の学科の出席番号と氏名**をコメントとして書く
- 保存したファイルは次のように「report」コマンドで提出する
  (ちゃんと提出できた場合は、「Succeed.」と画面に表示される)

```
$ ~kogai/report  kiso2term  「プログラムファイル」
```

- 複数のファイルを提出する場合は、report コマンドを分けて提出する
  例えば、test1.c と test2.c のファイルを提出したい場合は、次のように 2 回に分けて提出する

```
$ ~kogai/report  kiso2term  test1.c
$ ~kogai/report  kiso2term  test2.c
```

- 同じ問題に対して、複数の提出ファイルが存在した場合は、更新日時が新しい方を提出ファイルとする
- 提出するファイルは、誰から提出されたのか区別されるため、ファイル名は各自で自由に決めて良い

後期期末試験 模範解答（平均 78.9 点）

採点について コンパイル時にエラーとなる箇所は −4 点, 実行可能だか処理内容が問題の意図と違う箇所は −2 点を基本とする。

問 1 のプログラム

```
#include <stdio.h>
#define NUM 100

int main(void)
{
    FILE *in, *out;
    char str[NUM];
    int l = 0;
    /* コピー元とコピー先のファイルを開く */
    in = fopen("in.txt", "r");
    out = fopen("out.txt", "w");
    if(in == NULL) {
        printf("入力ファイルを開けませんでした\n");
        return 1;
    }
    if(out == NULL) {
        printf("出力ファイルを開けませんでした\n");
        return 1;
    }
    /* コピー元から 1 文字ずつ読み込む */
    while(fgets(str, NUM, in) != NULL) {
        /* コピー先に書き込む */
        fputs(str, out);
        l++;
    }
    /* コピーした行数を書き込む */
    fprintf(out, "コピーした行数: %d\n", l);
    /* ファイルを閉じる */
    fclose(in);
    fclose(out);

    return 0;
}
```

問 2〜4 のプログラム

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

/* 配列の要素数 */
#define NUM 5

/* 商品の情報を保持する構造体 */
typedef struct Item {
    int tanka;
    int kosu;
} Item;

typedef struct Item2 {
    int tanka;
    int kosu;
    struct Item2 *next;
} Item2;

/* 関数のプロトイタプ宣言 */
void show_itemlist(Item2 *start, char *str);
void show_total(Item *p);
void add_kosu(Item2 *start, int n);
void add_item(Item2 *p, int t, int k);

/* 構造体 Item2 の線形リストを出力する */
void show_itemlist(Item2 *start, char *str)
{
    Item2 *p;
    printf("--- %s ---\n", str);
    for(p = start; p!=NULL; p = p->next) {
        printf("単価 %d 円, 個数 %d 個\n",
               p->tanka, p->kosu);
    }
}

/* 総額を計算する */
void show_total(Item *p)
{
    int i, sum = 0;
    for(i=0; i<NUM; i++) {
        printf("%d 円 * %d 個 = %d 円\n",
               (p+i)->tanka, (p+i)->kosu,
               (p+i)->tanka * (p+i)->kosu);
        sum += (p+i)->tanka * (p+i)->kosu;
    }
    printf("合計: %d 円\n", sum);
}

/* 構造体 Item2 の線形リストを出力する */
/*
   ・リストに対する繰り返し処理→ 13 点
   ・メンバ kosu に対する加算処理→ 13 点
 */
void add_kosu(Item2 *start, int n)
{
    Item2 *p;
    for(p = start; p!=NULL; p = p->next) {
        p->kosu += n;
    }
}

/* リストに商品を push する */
void add_item(Item2 *p, int t, int k)
{
    Item2 *new;

    /* 構造体の領域を確保する */
    new = (Item2 *)malloc(sizeof(Item2));
    /* 確保した領域のメンバに引数の値を代入する */
```

```
    if(t > 0) {
        new->tanka = t;
    } else {
        new->tanka = -1 * t;
    }
    if(k > 0) {
        new->kosu = k;
    } else {
        new->kosu = -1 * k;
    }
    /* 確保した領域の next を更新する */
    new->next = p->next;
    /* p の next は確保した領域を参照する */
    p->next = new;
}

int main(void)
{
    /* 問 2 */
    Item items1[NUM] = {
        {500, 3},
        {100, 2},
        {210, 1},
        {980, 4},
        {320, 3}
    };
    show_total(items1);
    items1[2].kosu = 3;
    items1[3].kosu = 1;
    show_total(items1);
    /* 問 3 */
    Item2 it0, it1, it2, it3;
    it0.tanka = 300; it0.kosu = 2;
    it1.tanka = 800; it1.kosu = 3;
    it2.tanka = 400; it2.kosu = 0;
    /* リストを作る */
    it0.next = &it1;
    it1.next = &it2;
    it2.next = NULL;
    show_itemlist(&it0, "初期状態");
    /* 動作を確認する */
    add_kosu(&it0, 3);
    show_itemlist(&it0, "3 個追加");

    it3.tanka = 300; it3.kosu = 2;
    it2.next = &it3;
    it3.next = NULL;
    add_kosu(&it0, 2);
    show_itemlist(&it0, "さらに 2 個追加");
    /* 問 4 */
    Item2 head;
    head.tanka = 0; head.kosu = 0; head.next = NULL;
    show_itemlist(&head, "head (1)");
    add_item(&head, 2000, 3);
    show_itemlist(&head, "head (2)");
    add_item(&head, 4000, -5);
    show_itemlist(&head, "head (3)");
    add_item(&head, -1000, 2);
    show_itemlist(&head, "head (4)");

    return 0;
}
```