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、C の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の一部や構造体の宣言が、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/kiso201701.c . （←ここにピリオド）
```

1 ファイル「in.txt」に以下の様なプログラム処理が保存されているとする。

in.txt（読み込むファイル）

```
int x;
x = 10;
x++;
```

このファイルを読み込み、**4 文字分字下げされ、main 関数の記述が付加された**ファイル「out.c」を自動生成したい。(以下参照)

out.c（生成されたファイル）

```
int main(void)
{
    int x;
    x = 10;
    x++;
    return 0;
}
```

このようなファイルを自動生成するプログラムの一部が以下の様になった。不足している部分を書き足してプログラムを完成させなさい。なお、このプログラムは「kiso201701.c」に用意されているため、必要に応じて利用よい。

作成したプログラムの一部

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

int main(void)
{
    FILE *in, *out;
    char str[NUM];
    in = fopen("in.txt", "r");
    out = fopen("out.c", "w");
    fputs("int main(void)\n", out);
    fputs("{\n", out);
    while((fgets(str, NUM-1, in)) != NULL) {

    }

    fclose(in);
    fclose(out);
    return 0;
}
```

2 次のような分数を表す構造体を考える。この構造体の宣言は「kiso201701.c」に用意されている。

```
typedef struct Frac {
    int bunshi;  /* 分子の値 */
    int bunbo;   /* 分母の値 */
} Frac;
```

「この構造体が保持している分数の情報を、指定された形式で出力する」関数 show() 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(Frac *p, int mode);
  /* 仮引数 mode の値が 1 の場合、「"分子"/"分母"」の形式で出力する */
  /* 仮引数 mode の値がそれ以外の場合、「"分母" 分の "分子"」の形式で出力する */
  /* 出力の書式は実行結果の様子を参照 */
```

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。

```
[main での処理]
  Frac f1 = {2, 5};
  Frac f2 = {1, 3};
  show(&f1, 1);
  show(&f2, 0);
[実行結果]
  2/5
  3 分の 1
```

3 前問に示した構造体 Frac の配列（3 個分の分数）に対して、「指定した要素に整数値を加算する」関数 add_frac() 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_frac(Frac *p, int i, int num);
  /* 仮引数 p が参照している配列の i 番目の分数に、整数値 num を加算する */
  /* この処理によって、i 番目の分数は加算後の値となる */
  /* 加算後の分数に対して約分は考慮しなくてよい */
  /* p が参照している配列の要素数は、3 個を前提に作成してよい */
```

例えば、i 番の分数が $\frac{2}{5}$、num の値が 4 の場合、

$$\frac{2}{5} + \frac{4}{1} = \frac{22}{5} \text{が } i \text{ 番の値となる}$$

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、ここで使っている出力用関数 show_fracs() は「kiso201701.c」に用意されている。

```
[main での処理]
  Frac fracs1[3] = {
      {2, 5},
      {1, 3},
      {4, 7}
  };
  add_frac(fracs1, 0, 4);
  add_frac(fracs1, 2, 3);
  show_fracs(fracs1, "fracs1");
```

```
[実行結果]
  --- fracs1 ---
  [0] 22/5    (← 2/5 + 4)
  [1] 1/3
  [2] 25/7    (← 4/7 + 3)
```

4 次のような分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 Frac とは異なることに注意すること。
この構造体の宣言は「kiso201701.c」に用意されている。

```
typedef struct Frac2 {
    int bunshi;          /* 分子 */
    int bunbo;           /* 分母 */
    struct Frac2 *next;  /* 線形リスト用 */
} Frac2;
```

この構造体 Frac2 を使った線形リストに対して、「スタックの push 操作をする」関数 push_frac() 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 push_frac(Frac2 *p, int s, int b);
  /* 仮引数 p が参照しているリストに新しい要素を push する */
  /* 仮引数 s, b は、push される構造体 Frac2 のメンバ bunshi, bunbo にそれぞれ格納される */
  /* push の処理は、第 12 回または第 13 回と同様に、リストの先頭に格納する */
```

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、ここで使っている出力用関数 show_fraclist() は「kiso201701.c」に用意されている。

```
[main での処理]
  Frac2 head1;
  head1.bunshi = 0; head1.bunbo = 0;   /* 空のスタックを準備する */
  head1.next = NULL;
  show_fraclist(&head1, "head1 (1)");  /* 初期状態を出力する */
  push_frac(&head1, 3, 8);             /* 3 回 push する */
  push_frac(&head1, 2, 9);
  push_frac(&head1, 7, 2);
  show_fraclist(&head1, "head1 (2)");  /* push 後の状態を出力する */
[実行結果]
  --- head1 (1) ---
  0/0
  --- head1 (2) ---
  0/0
  7/2
  2/9
  3/8
```

5 構造体 Frac2 の要素が格納されているリストに対して、「仮分数または真分数のみを出力する」関数 select_frac() 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 select_frac(Frac2 *start, char *str, int mode);
  /* show_fraclist() と同様に、リスト全体に対して繰り返し処理をする */
  /* 仮引数 mode が 1 の場合、仮分数（つまり分子の値が分母以上の分数）のみが出力される */
  /* 仮引数 mode が 2 の場合、真分数（つまり分子の値が分母未満の分数）のみが出力される */
  /* 仮引数 mode がこれら以外の場合、何も出力されない */
```

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。

```
[main での処理（問 4 に示した main の続きに書くとする）]
  push_frac(&head1, 5, 5);
  select_frac(&head1, "仮分数", 1);
  select_frac(&head1, "真分数", 2);
  select_frac(&head1, "1, 2 以外の指定", 3);
[実行結果]
  --- 仮分数 ---            (←仮分数だけが出力される)
  0/0
  5/5
  7/2
  --- 真分数 ---            (←真分数だけが出力される)
  2/9
  3/8
  --- 1, 2 以外の指定 ---   (← 1, 2 以外の場合は何も出力されない)
```

（各 20 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

* **上記以外の情報を参照することはカンニング行為とする**
（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 「提出先」 「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
（ただし、提出するファイルの先頭には、出席番号と氏名を記入する）
- 「提出先」への提出は試験時のみ可能である

後期期末試験 模範解答（平均 65.5 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];
    /* ファイルを開く */
    in = fopen("in.txt", "r");
    out = fopen("out.c", "w");
    /* プログラムの先頭部分を書き込む */
    fputs("int main(void)\n", out);
    fputs("{\n", out);
    while((fgets(str, NUM-1, in)) != NULL) {
        /* 字下げをして 1 行書き込む */
        fprintf(out, "    %s", str);
    }
    /* プログラムの末尾部分を書き込む */
    fputs("    return 0;\n", out);
    fputs("}\n", out);
    /* ファイルを閉じる */
    fclose(in);
    fclose(out);

    return 0;
}
```

問 2〜5 のプログラム

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

/* 分数の構造体 */
typedef struct Frac {
    int bunshi;
    int bunbo;
} Frac;

/* 分数の構造体（リスト用） */
typedef struct Frac2 {
    int bunshi;
    int bunbo;
    struct Frac2 *next;
} Frac2;

/* 関数のプロトタイプ宣言 */
void show(Frac *p, int mode);
void add_frac(Frac *p, int i, int num);
void show_fracs(Frac *p, char *str);
void push_frac(Frac2 *p, int s, int b);
void show_fraclist(Frac2 *p, char *str);
void select_frac(Frac2 *p, char *str, int mode);

/* 分数を指定した形式で出力する */
void show(Frac *p, int mode)
{
    if(mode == 1) {
        printf("%d/%d\n", p->bunshi, p->bunbo);
    } else {
        printf("%d 分の %d\n", p->bunbo, p->bunshi);
    }
}

/* 配列に対して分数の加算をする */
void add_frac(Frac *p, int i, int num)
{
    (p+i)->bunshi = (p+i)->bunshi + num * (p+i)->bunbo;
}

/* 分数の配列を出力する */
void show_fracs(Frac *p, char *str)
{
    int i = 0;
    printf("--- %s ---\n", str);
    for(i=0; i<3; i++) {
        printf("[%d] %d/%d\n",
               i, (p+i)->bunshi, (p+i)->bunbo);
    }
}

/* スタックを出力する */
void show_fraclist(Frac2 *start, char *str)
{
    Frac2 *pfrac;
    printf("--- %s ---\n", str);
    for(pfrac = start; pfrac!=NULL;
          pfrac = pfrac->next) {
        printf("%d/%d\n", pfrac->bunshi, pfrac->bunbo);
    }
}

/* スタックに要素を追加する */
void push_frac(Frac2 *p, int s, int b)
{
    Frac2 *new;
    /* 構造体 Frac2 の領域を確保する */
    new = (Frac2 *)malloc(sizeof(Frac2));
    /* 確保した領域のメンバに引数の値を代入する */
    new->bunshi = s;
    new->bunbo = b;
    /* 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;
}
```

```
/* 真分数または仮分数を出力する */
void select_frac(Frac2 *start, char *str, int mode)
{
    Frac2 *pfrac;
    printf("--- %s ---\n", str);
    for(pfrac = start; pfrac!=NULL;
          pfrac = pfrac->next) {
        if(mode == 1) {
            /* 仮分数を出力 */
            if(pfrac->bunshi >= pfrac->bunbo) {
                printf("%d/%d\n",
                       pfrac->bunshi, pfrac->bunbo);
            }
        } else if(mode == 2) {
            /* 真分数を出力 */
            if(pfrac->bunshi < pfrac->bunbo) {
                printf("%d/%d\n",
                       pfrac->bunshi, pfrac->bunbo);
            }
        }
    }
}

int main(void)
{
    Frac f1 = {2, 5};
    Frac f2 = {1, 3};
    Frac fracs1[3] = {
        {2, 5},
        {1, 3},
        {4, 7}
    };

    /* [2] */
    show(&f1, 1);
    show(&f2, 0);

    /* [3] */
    add_frac(fracs1, 0, 4);
    add_frac(fracs1, 2, 3);
    show_fracs(fracs1, "fracs1");

    /* [4] */
    Frac2 head1;
    /* 空のスタックを準備する */
    head1.bunshi = 0; head1.bunbo = 0;
    head1.next = NULL;
    show_fraclist(&head1, "head1 (1)");
    /* push_frac() の動作確認 */
    push_frac(&head1, 3, 8);
    push_frac(&head1, 2, 9);
    push_frac(&head1, 7, 2);
    show_fraclist(&head1, "head1 (2)");

    /* [5] */
    push_frac(&head1, 5, 5);
    select_frac(&head1, "仮分数", 1);
    select_frac(&head1, "真分数", 2);
    select_frac(&head1, "1, 2 以外の指定", 3);

    return 0;
}
```