プログラミング基礎 後期中間試験

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。

1 次のような int 型のポインタを使った処理を main() 内に作ることを考える。

```
  int x, y;
  int *max, *min;
  x = 300;
  y = 200;

  /* ----- if 文の処理 ----- */

  printf("大きい方の値は %d\n", *max);
  printf("小さい方の値は %d\n", *min);
```

この時、「変数 x と y に格納されている値を比較して、大きい方の変数をポインタ max が参照し、小さい方の変数をポインタ min が参照する」処理となるように、**コメントで示した箇所に「if 文の処理」を追加して**、このプログラムを完成させなさい。実行結果は以下のようになる。

```
[実行結果]
  $ ./a.out              (← x に 300、y に 200 を代入した場合の実行)
  大きい方の値は 300
  小さい方の値は 200

  $ ./a.out              (← x に 100、y に 200 を代入した場合の実行)
  大きい方の値は 200
  小さい方の値は 100
```

2 仮引数 x, y, z で与えられた 3 つの整数に対して、「3 つの整数の合計と平均を求める」関数 sum_avg() 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 sum_avg(int x, int y, int z, int *s, double *a);
  /*  x, y, z の合計を *s に代入する  */
  /*  平均を計算して、 *a に代入する（double 型で計算する必要がある）  */
```

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。

```
[main での処理]
  int sum;
  double avg;
  sum_avg(3, 5, 7, &sum, &avg);
  printf("sum: %d\n", sum);
  printf("avg: %lf\n", avg);
  sum_avg(6, 4, 7, &sum, &avg);
  printf("sum: %d\n", sum);
  printf("avg: %lf\n", avg);
[実行結果]
  sum: 15
  avg: 5.000000
  sum: 17
  avg: 5.666667
```

3 仮引数のポインタが参照する int 型の配列（要素数は 5）に対して、「全ての要素の中から、仮引数 n と値が等しい要素を探し、何番目に見つかったのか場所を返す」関数 find_num() を作成しなさい。

ただし、以下のような動作となること。

- 参照する配列の要素数は 5 であることを前提にする
- 仮引数 n と値が等しい要素が複数ある場合は、**先に見つかった方の場所を返す**
- 値が等しい要素が見つからなかった場合は、−1 を返す

この関数のプロトタイプ宣言は以下のようになる。

```
int find_num(int *ptr, int n);
  /* 何番目に見つかったのかを格納する変数を用意する（初期値を-1 にしておくと作りやすい） */
  /* ptr が参照する配列の全要素（要素数は 5）に対する繰り返し処理を作る */
  /* 繰り返し処理の中で、ptr が参照している配列の i 番目の値が、n と等しい場合、
        見つかった場所を i 番目にして、繰り返し処理を終了する */
  /* 繰り返し処理終了後に、見つかった場所を戻り値とする */
```

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。

```
[main での処理]
  int a[5] = {5, 2, 3, 1, 3};
  printf("%d\n", find_num(a, 3));
  printf("%d\n", find_num(a, 7));
[実行結果]
  2       (← 配列 a で、3 の値が先に見つかった場所が 2 番目)
  -1      (← 配列 a で、7 の値が見つからなかった)
```

4 仮引数のポインタが参照する文字列に対して、「アルファベット小文字を仮引数 ch の文字に置き換え、置き換えた文字数を返す」関数 replace_lower() 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。

```
int replace_lower(char *s, char ch);
  /* 個数を数えるための変数を用意する */
  /* s が参照する文字列全体の繰り返し処理を作る */
  /* 繰り返し処理の中で、s+i が参照する文字が小文字がどうか調べる
        (小文字の判定は「 'a' 以上 かつ 'z' 以下 かどうか」でできる) */
  /* 小文字の場合は、s+i が参照する文字を ch に置き換えて、数えている個数を 1 つ増やす */
  /* 関数内で文字列 s を出力するのではなく、置換した結果が文字列 s に反映されるように作る */
  /* 繰り返し処理終了後に、数えた個数を戻り値とする */
```

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。

```
[main での処理]
  char str1[] = "ComputerScience";
  char str2[] = "C-Java-PHP";
  int r;
  r = replace_lower(str1, '*');
  printf("文字列: %s, 個数: %d\n", str1, r);
  r = replace_lower(str2, '?');
  printf("文字列: %s, 個数: %d\n", str2, r);
[実行結果]
  文字列: C******S******, 個数: 13        (← 13 個の小文字が * に置き換えられた)
  文字列: C-J???-PHP, 個数: 3             (← 3 個の小文字が ? に置き換えられた)
```

5 char 型の配列のためのメモリを確保し、仮引数 ch で与えられたアルファベット小文字から 'z'（小文字の z）までがアルファベット順に格納された文字列を作る関数 fill_alpha() を作成しなさい。

ただし、仮引数 ch は、アルファベット小文字 'a'〜'z' であることを前提に作ってよい。この関数のプロトタイプ宣言は以下のようになる。

```
char *fill_alpha(char ch);
  /* 必要な配列の要素数分（終端文字分も含めると、「'z'-ch+2」個分）のメモリを確保する */
  /* 繰り返し処理によって、確保した配列の先頭から、ch, ch+1, ch+2, ... の文字を'z'まで代入していく */
  /* 文字の最後に終端文字を代入する */
  /* 作成した配列のアドレスを return で返す */
```

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]
  char *str3;
  str3 = fill_alpha('a');
  printf("%s\n", str3);
  str3 = fill_alpha('p');
  printf("%s\n", str3);
[実行結果]
  abcdefghijklmnopqrstuvwxyz     (← 'a'〜'z' が格納された文字列が作られた)
  pqrstuvwxyz                    (← 'p'〜'z' が格納された文字列が作られた)
```

問題はここまで

（各 20 点）

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 kiso2mid 「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 kiso2mid test1.c
$ ~kogai/report kiso2mid 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

後期中間試験 模範解答（平均 88.7 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。
配点：1 〜 5 各 20 点

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

/* 関数のプロトタイプ宣言 */
void sum_avg(int x, int y, int z, int *s, double *a);
int find_num(int *ptr, int n);
int replace_lower(char *str, char ch);
char *fill_alpha(char ch);

/* [2] */
void sum_avg(int x, int y, int z, int *s, double *a)
{
    /* 合計を求める */
    *s = x + y + z;
    /* 平均を求める（double 型にキャストする） */
    *a = (double)*s / (double)3;
}

/* [3] */
int find_num(int *ptr, int n)
{
    /* 戻り値の初期値を-1 とする */
    int result = -1;
    int i;
    /* 配列の全要素に対する繰り返し処理 */
    for(i=0; i<5; i++) {
        /* i 番目の文字が n と等しいかどうか */
        if( *(ptr+i) == n) {
            /* 戻り値を「i 番目」にする */
            result = i;
            /* 繰り返し処理を中断する */
            break;
        }
    }
    return result;
}

/* [4] */
int replace_lower(char *s, char ch)
{
    /* 個数を数える変数を初期化する */
    int result = 0;
    int i;
    /* 文字列に対する繰り返し処理 */
    for(i=0; *(s+i) != '\0'; i++) {
        /* i 番目の文字が小文字かどうか */
        if( *(s+i) >= 'a' && *(s+i) <= 'z' ) {
            /* i 番目の文字を ch に置き換える */
            *(s+i) = ch;
            /* 置き換えた個数を 1 つ増やす */
            result++;
        }
    }
    return result;
}

/* [5] */
/*
   問 5 については以下の部分点を考慮する
   malloc 部分 5 点，for 文 5 点，
   終端文字の処理 5 点，return 部分 5 点
*/
char *fill_alpha(char ch)
{
    int i;
    char *str;
    /* ch から'z' までの文字列に必要なメモリを確保する */
    str = (char *)malloc(sizeof(char)* ('z' - ch + 2));
    if(str==NULL) {
        printf("not allocated.\n");
        return NULL;
    }
    /* ch から'z' までの繰り返し処理 */
    for(i=0; ch+i <= 'z'; i++) {
        /* i 番目にアルファベット ch+i を代入する */
        *(str+i) = ch+i;
    }
    /* 最後に終端文字を代入する */
    *(str+i) = '\0';
    return str;
}

int main(void)
{
    /*  [1]  */
    int x, y;
    int *max, *min;
    x = 100; //= 100;
    y = 200;

    /* x と y の値を比較する */
    if(x>=y) {
        /* x が大きい場合 */
        max = &x;
        min = &y;
    } else {
```

```
        /* y が大きい場合 */
        min = &x;
        max = &y;
    }
    printf("大きい方の値は %d\n", *max);
    printf("小さい方の値は %d\n", *min);

    /* [2] の動作確認 */
    int sum;
    double avg;
    sum_avg(3, 5, 7, &sum, &avg);
    printf("sum: %d\n", sum);
    printf("avg: %lf\n", avg);
    sum_avg(6, 4, 7, &sum, &avg);
    printf("sum: %d\n", sum);
    printf("avg: %lf\n", avg);

    /* [3] の動作確認 */
    int a[5] = {5, 2, 3, 1, 3};
    printf("%d\n", find_num(a, 3));
    printf("%d\n", find_num(a, 7));

    /* [4] の動作確認 */
    char str1[] = "ComputerScience";
    char str2[] = "C-Java-PHP";
    int r;
    r = replace_lower(str1, '*');
    printf("文字列: %s, 個数: %d\n", str1, r);
    r = replace_lower(str2, '?');
    printf("文字列: %s, 個数: %d\n", str2, r);

    /* [5] の動作確認 */
    char *str3;
    str3 = fill_alpha('a');
    printf("%s\n", str3);
    str3 = fill_alpha('p');
    printf("%s\n", str3);

    return 0;
}
```