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、C のコメントとして自分の番号と名前を書くこと。

1 以下のようにポインタ ptr を宣言し、代入をする。

```
char *ptr;
char ch = 'Z';
ptr = &ch;
```

この後に、「Z（大文字）から、順に A（大文字）までのアルファベットを出力する」処理を main() に作りなさい。ただし、処理に使える変数は、ポインタ ptr のみとする（ch も含め他の変数は使わない）。

2 仮引数 num1, num2 で与えられた 2 つの整数に対して「num1 の平方根（$\sqrt{num1}$）と num2 の平方根（$\sqrt{num2}$）の計算結果を求める」関数 calc_sqrt() 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 calc_sqrt(int num1, int num2, float *sqrt1, float *sqrt2);
  /* num1 と num2 に対して平方根を計算し、num1 の平方根は *sqrt1、num2 の平方根は *sqrt2 に代入する */
  /* なお、平方根は、関数 sqrt() を使って計算できる */
```

関数 sqrt() については以下を参考にすること。（前期第 5 回を参照）

- プログラムの先頭で、math.h を読み込む必要がある
- コンパイルの際に、「-lm」を付ける必要がある
  ［例］ `$ cc test2.c -lm`
- 関数 sqrt() は引数に指定した値の平方根を返してくれる
  ［例］ `sqrt(2) /* 1.41421356237 が返される */`

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]
  float s1, s2;
  calc_sqrt(10, 3, &s1, &s2);
  printf("sqrt1: %f, sqrt2: %f\n", s1, s2);
  calc_sqrt(2, 9, &s1, &s2);
  printf("sqrt1: %f, sqrt2: %f\n", s1, s2);
[実行結果]
  sqrt1: 3.162278, sqrt2: 1.732051
  sqrt1: 1.414214, sqrt2: 3.000000
```

3 仮引数で与えられたポインタが参照する int 型の配列（要素数は 8）に対して、全ての要素の中から、**low 以上かつ up 以下の範囲の値となる**要素の個数を求める関数 count_num() を作成しなさい。
ただし、仮引数 low, up は、$low \leq up$ であり、参照する配列の要素数は 8 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。

```
int count_num(int *ptr, int low, int up);
  /* 個数を数えるための変数を用意する */
  /* ptr が参照する配列の全要素（要素数は 8）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、low〜up の範囲内である場合、個数を 1 つ増やす */
  /* 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]
  int a1[8] = {-2, 3, 17, -7, 11, -13, 19, 5};
  int a2[8] = {16, 6, -10, -4, 8, 12, 2, -14};
  printf("%d\n", count_num(a1, 5, 19));
  printf("%d\n", count_num(a1, -2, 5));
  printf("%d\n", count_num(a2, -10, 6));
[実行結果]
  4       (← 配列 a1 で、5 以上かつ 19 以下となる要素の個数)
  3       (← 配列 a1 で、-2 以上かつ 5 以下となる要素の個数)
  4       (← 配列 a2 で、-10 以上かつ 6 以下となる要素の個数)
```

4 仮引数に与えられたポインタが参照する文字列に対して、「それぞれ文字を、引数で指定した分だけ文字コードを加算してシフトする」関数 shift_char() 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 shift_char(char *s, int n);
  /* 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 が参照している各文字に引数 n を加算する */
  /* 加算の際は、文字コードが ASCII 文字コード表内に収まっているかどうかを調べる必要はない */
  /* ちなみに、これは「シーザー暗号」として知られている古典的な暗号化の手法である */
```

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]
    char str0[20] = "PL!Hpphmf"; /* 暗号文その 1 */
    char str1[20] = "Li}$Wmvm";  /* 暗号文その 2 */
    shift_char(str0, -1);
    shift_char(str1, -4);
    printf("%s, %s\n", str0, str1);
    shift_char(str0, 1);
    shift_char(str1, 4);
    printf("%s, %s\n", str0, str1);
[実行結果]
  OK Google, Hey Siri     (←復号化された)
  PL!Hpphmf, Li}$Wmvm     (←暗号文にもどった)
```

5 int 型の配列のためのメモリを確保し、仮引数で与えられた整数 s から e までの整数が昇順で格納された配列を作る関数 fill_range() 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 *fill_range(int s, int e);
  /* s と e から、必要な配列の要素数（main の処理を参照）を求めて、配列のためのメモリを確保する */
  /* 作成した配列の先頭から、s, s+1, s+2, ..., e の整数を格納していく */
  /* 作成した配列のアドレスを return で戻す */
  /* なお、「配列に格納する場所（?番目)」と「格納する値」は異なるため、別々の変数を用意した方が作りやすい */
```

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]

```
  int *a;
  int i;
  a = fill_range(5, 20);      /* 処理その 1 */
  for(i=0; i<(20-5)+1; i++) {
      printf("%d ", *(a+i));
  }
  printf("\n");
  free(a);
  a = fill_range(100, 1000);  /* 処理その 2 */
  for(i=0; i<(1000-100)+1; i++) {
      printf("%d ", *(a+i));
  }
  printf("\n");
  free(a);
```

[実行結果]

```
  5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20              (←処理その 1：5〜20 の整数が格納されている)
  100 101 102 103 104 105 106 107 108（長いので以下省略）    (←処理その 2)
```

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 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
（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

前期期末試験 模範解答（平均 81.9 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。
配点：1 〜 5 各 20 点

```
/* 自分の番号と名前をここに書く */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* 関数のプロトタイプ宣言 */
void calc_sqrt(int num1, int num2,
               float *sqrt1, float *sqrt2);
int count_num(int *ptr, int low, int up);
void shift_char(char *s, int n);
int *fill_range(int s, int e);

/* [2] 仮引数の平方根を計算する */
void calc_sqrt(int num1, int num2,
               float *sqrt1, float *sqrt2)
{
    /* 平方根の結果を代入する */
    *sqrt1 = sqrt(num1);
    *sqrt2 = sqrt(num2);
}
/* [3] 要素の値が up〜low の範囲となる個数を求める */
int count_num(int *ptr, int low, int up)
{
    int i, result;
    result = 0;
    /* 配列全体を繰り返す */
    for(i=0; i<8; i++) {
        if( *(ptr+i) <= up && *(ptr+i) >= low ) {
            result ++;
        }
    }
    return result;
}
/* [4] 文字列の各文字の文字コードをシフトする */
void shift_char(char *s, int n)
{
    int i;
    for(i=0; *(s+i)!='\0'; i++) {
        /* 文字コードに n を加算する */
        *(s+i) += n;
    }
}
/* [5] s〜e の整数が格納された配列を作る */
int *fill_range(int s, int e)
{
    int *array;
    int n, i;
    /* int 型のメモリを確保する */
    array = (int *)malloc(sizeof(int)*(e-s)+1);
    if(array==NULL) {
        printf("not allocated.\n");
        return NULL;
    }
    /* 確保する整数の初期値は s */
    n = s;
    /* 確保したメモリに s〜e の整数を入れる */
    for(i=0; i<=(e-s)+1; i++) {
        *(array+i) = n;
        n++;
    }
    /* 確保したメモリの先頭アドレスを戻す */
    return array;
}

int main(void)
{
    /* [1] Z から A までの文字を順に出力する */
    char *ptr;
    char ch = 'Z';
    ptr = &ch;
    /* ptr のみを使って 'Z' から'A' まで
       減算しながら出力する */
    for(; *ptr>='A'; (*ptr)--) {
        printf("%c ", *ptr);
    }
    printf("\n");

    /* [2] の動作確認 */
    float s1, s2;
    calc_sqrt(10, 3, &s1, &s2);
    printf("sqrt1: %f, sqrt2: %f\n",
           s1, s2);
    calc_sqrt(2, 9, &s1, &s2);
    printf("sqrt1: %f, sqrt2: %f\n",
           s1, s2);

    /* [3] の動作確認 */
    int a1[8] = {-2, 3, 17, -7, 11, -13, 19, 5};
    int a2[8] = {16, 6, -10, -4, 8, 12, 2, -14};
    printf("%d\n", count_num(a1, 5, 19));
    printf("%d\n", count_num(a1, -2, 5));
    printf("%d\n", count_num(a2, -10, 6));

    /* [4] の動作確認 */
    char str0[20] = "PL!Hpphmf";
    char str1[20] = "Li}$Wmvm";
    shift_char(str0, -1);
    shift_char(str1, -4);
    printf("%s, %s\n", str0, str1);
    shift_char(str0, 1);
    shift_char(str1, 4);
    printf("%s, %s\n", str0, str1);

    /* [5] の動作確認 */
    int *a;
    int i;
    a = fill_range(5, 20);
    for(i=0; i<(20-5)+1; i++) {
        printf("%d ", *(a+i));
    }
    printf("\n");
    free(a);
    a = fill_range(100, 1000);
    for(i=0; i<(1000-100)+1; i++) {
        printf("%d ", *(a+i));
    }
    printf("\n");
    free(a);

    return 0;
}
```