プログラミング応用 前期期末試験

※提出する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 再帰呼び出しを利用して、2 つの整数 x, y に対する x^y（x の y 乗）を求める関数 power() 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 power(int x, int y);
  /* y が 0 より大きい場合は再帰呼び出しをするように if 文を作る */
  /* y が 0 より大きい場合、x と再帰呼び出しの結果をかけ算する  */
  /* y が 1 ずつ減るように再帰呼び出しをする */
```

main の例とその実行結果は以下のようになる。

```
[main での処理]
  printf("%d\n", power(2, 3));
  printf("%d\n", power(10, 5));
  printf("%d\n", power(3, 0));

[実行結果]
  8
  100000
  1
```

2 数列の第 n 項の値を求める関数 term() が、次のような性質を持っているとする。

- `term(0) = 1`
- `term(1) = 3`
- `term(n) = 3*term(n-1) - 2*term(n-2)`

再帰呼び出しを利用して、この数列の第 n 項の値を求める関数 term() 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 term(int x);
  /* 上記の性質で場合分けした if 文を作り、その中で再帰呼び出しをする */
```

main の例とその実行結果は以下のようになる。

```
[main での処理]
  int i;
  for(i=0; i<20; i++) {
      printf("%d ", term(i));
  }
  printf("\n");

[実行結果]（最初から 20 項分を表示している）
  1 3 7 15 31 63 127 255 511 1023 2047 4095 8191 16383 32767 65535 131071 262143 524287 1048575
```

3 線形探索によって「指定した範囲内に含まれる値を全て見つける」関数 search_range_all() 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 search_range_all(char *label[], int value[], int s, int e, int size);
  /* 条件を満たす値を全て見つけるように線形探索する */
  /* 見つける条件を「s から e までの範囲に含まれているかどうか」にする */
```

main の例とその実行結果は以下のようになる。

```
[main での処理]
  char *months[7] = {"1 月", "2 月", "3 月", "4 月", "5 月", "6 月"};  //データのラベル（各月の降水量という意味）
  int newyork[7]  = {54, 151, 104, 128, 82, 92};    //ニューヨークの月別降水量
  int honolulu[7] = {2, 105, 65, 30, 6, 4};         //ホノルルの月別降水量
  search_range_all(months, newyork, 104, 128, 6);   //実行その 1
  search_range_all(months, newyork, 54, 92, 6);     //実行その 2
  search_range_all(months, honolulu, 6, 30, 6);     //実行その 3
  search_range_all(months, honolulu, 7, 29, 6);     //実行その 4

[実行結果]
  label: 3 月, value: 104          (←実行その 1 の結果)
  label: 4 月, value: 128
  label: 1 月, value: 54           (←実行その 2 の結果)
  label: 5 月, value: 82
  label: 6 月, value: 92
  label: 4 月, value: 30           (←実行その 3 の結果)
  label: 5 月, value: 6
  見つかりませんでした              (←実行その 4 の結果)
```

4 降順であらかじめ並んでいる float 型の配列を二分探索する関数 binary_f() 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 binary_f(char *label[], float value[], float n, int left, int right);
  /* 降順に並んでいる配列 value を二分探索するように if 文の条件を作る */
  /* 配列 value の要素は float 型であることに注意する */
```

main の例とその実行結果は以下のようになる。

```
[main での処理]
  char *l[7] = {"num0", "num1", "num2", "num3", "num4", "num5", "num6"};
  float f[7] = {18.5, 16.4, 15.9, 13.1, 11.8, 8.4, 3.0};
  binary_f(l, f, 15.9, 0, 6);
  binary_f(l, f, 18.5, 0, 6);
  binary_f(l, f, 3.0, 0, 6);
  binary_f(l, f, 10.5, 0, 6);

[実行結果]
  label: num2, value: 15.900000
  label: num0, value: 18.500000
  label: num6, value: 3.000000
  見つかりませんでした
```

（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

- 提出する全てのファイル内に、**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 ouyou1term 「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 ouyou1term test1.c
$ ~kogai/report ouyou1term 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

前期期末試験 模範解答（平均 96.6 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。

```
#include <stdio.h>

int power(int x, int y);
int term(int x);
void search_range_all(char *label[], int value[], int s, int e, int size);
void binary_f(char *label[], float value[], float n, int left, int right);

int power(int x, int y)
{
    if(y > 0) {
        return x * power(x, y-1);
    } else {
        return 1;
    }
}

int term(int x)
{
    if(x==0) {
        return 1;
    } else if(x==1) {
        return 3;
    } else {
        return 3*term(x-1) - 2*term(x-2);
    }
}

void search_range_all(char *label[], int value[], int s, int e, int size)
{
    int i = 0;
    int found = 0;
    while(1) {
        if(s<=value[i] && value[i]<=e) {
            printf("label: %s, ", label[i]);
            printf("value: %d\n", value[i]);
            found = 1;
            //break;
        }
        if(i==size-1) {
            if(found==0) {
                printf("見つかりませんでした\n");
            }
            break;
        }
        i++;
    }
}

void binary_f(char *label[], float value[], float n, int left, int right)
{
    int center;
    if(left>right) {
        printf("見つかりませんでした\n");
        return;
    }
    center = (left+right)/2;
    if(value[center]==n) {
```

```
        printf("label: %s, ", label[center]);
        printf("value: %f\n", value[center]);
        return;
    }
    if(value[center]<n) {
        binary_f(label, value, n, left, center-1);
    }
    if(value[center]>n) {
        binary_f(label, value, n, center+1, right);
    }
}

int main(void)
{
    //power
    printf("---power---\n");
    printf("%d\n", power(2, 3));
    printf("%d\n", power(10, 5));
    printf("%d\n", power(3, 0));
    //term
    printf("---term---\n");
    int i;
    for(i=0; i<20; i++) {
        printf("%d ", term(i));
    }
    printf("\n");
    //search_range
    printf("---search_range---\n");
    char *months[7] = {"1 月", "2 月", "3 月", "4 月", "5 月", "6 月"};
    int newyork[7]  = {54, 151, 104, 128, 82, 92};
    int honolulu[7] = {2, 105, 65, 30, 6, 4};
    search_range_all(months, newyork, 104, 128, 6);
    search_range_all(months, newyork, 54, 92, 6);
    search_range_all(months, honolulu, 6, 30, 6);
    search_range_all(months, honolulu, 7, 29, 6);
    //binary_f
    printf("---binary_f---\n");
    char *l[7] = {"num0", "num1", "num2", "num3", "num4", "num5", "num6"};
    float f[7] = {18.5, 16.4, 15.9, 13.1, 11.8, 8.4, 3.0};
    binary_f(l, f, 15.9, 0, 6);
    binary_f(l, f, 18.5, 0, 6);
    binary_f(l, f, 3.0, 0, 6);
    binary_f(l, f, 10.5, 0, 6);

    return 0;
}
```