プログラミング応用 後期中間試験

※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。

準備 **試験で利用可能なコードが事前に用意されている。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/ouyou201811.c  .  (←ここにピリオド)
```

1 引数で与えられた int 型配列の値を**降順（大きい順）に**バブルソートする 関数 bubble() を作成しなさい。なお、必要に応じて、変数の値を交換する関数 swap(), 配列の全要素を出力する関数 show() を利用してよい。(これらの関数は準備でコピーしたファイルから入手可能だが、授業で扱った関数と内容は同じ)

この問で作成する関数 bubble() のプロトタイプ宣言は以下のようになる。

```
void bubble(int value[], int size);
  /* 配列 value の要素数は size で与えられる */
```

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 v[10] = {18, 12, 13, 17, 15, 16, 10, 14, 19, 11};
  show(v, 10);
  bubble(v, 10);
  show(v, 10);

[実行結果]
  18 12 13 17 15 16 10 14 19 11      (←ソート前の出力)
  19 18 17 16 15 14 13 12 11 10      (←ソート後の出力)
```

2 引数で与えられた int 型配列に対して、**再帰呼び出しされた際に、配列の左端から右端までの要素を出力して**クイックソートをする関数 quick() を作成して下さい。なお、必要に応じて、変数の値を交換する関数 swap(), 配列の指定した範囲の要素を出力する関数 show2() を利用してよい。(これらの関数は準備でコピーしたファイルから入手可能)

関数 swap() は問 **1** で使う関数と同じだが、関数 show2() のプロトタイプ宣言は以下のようになる。

```
void show2(int value[], int left, int right);
  /* 配列 value の left 番目から right 番目までの値を出力する */
```

この問で作成する関数 quick() のプロトタイプ宣言は以下のようになる。

```
void quick(int value[], int left, int right);
  /* left と right の値を出力した後に、 */
  /* 配列 value の left 番目から right 番目までの範囲の要素を show2() で出力する処理を追加する */
```

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 v[10] = {9, 7, 6, 0, 1, 5, 2, 3, 4, 8};
  show2(v, 0, 9);
  quick(v, 0, 9);
  show2(v, 0, 9);
```

```
[実行結果]
  9 7 6 0 1 5 2 3 4 8                          (←ソート前の出力)
  left: 0, right: 9 → 9 7 6 0 1 5 2 3 4 8      (「→」以降の配列の出力は show2 を呼びだしている)
  left: 0, right: 1 → 1 0
  left: 2, right: 9 → 6 7 9 5 2 3 4 8
  left: 2, right: 4 → 4 3 2
  left: 6, right: 9 → 9 7 6 8
  left: 8, right: 9 → 9 8
  0 1 2 3 4 5 6 7 8 9                          (←ソート後の出力)
```

3 ニュートン法を使って仮引数 c の平方根を求める関数 mysqrt() を作成しなさい。ただし、仮引数 n の値によって以下のような動作し、x2 が変化する様子が分かるように出力される。

- n の値が 0 より大きい場合は、x1 から x2 を求める処理を n 回繰り返す
- n の値が 0 以下の場合は、x1 と x2 の差分が 0.0001 未満になったら終了する

この問で作成する関数 mysqrt() のプロトタイプ宣言は以下のようになる。

```
double mysqrt(double c, int n);
  /* n の値によって、平方根を求める処理が上記となるように場合分けする */
```

main の例とその実行結果は以下のようになる。

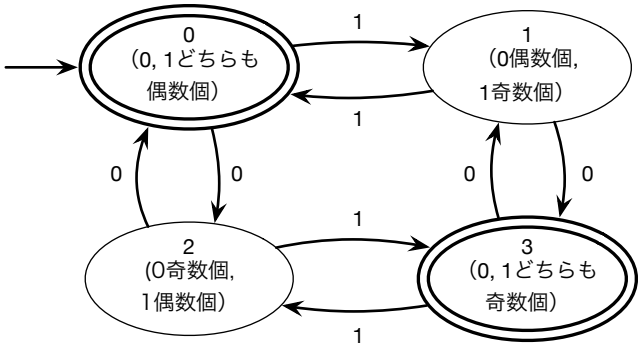
```
[main での処理]
  printf("----------\n");
  printf("%lf\n", mysqrt(2, 1));
  printf("----------\n");
  printf("%lf\n", mysqrt(2, 3));
  printf("----------\n");
  printf("%lf\n", mysqrt(10, 15));
  printf("----------\n");
  printf("%lf\n", mysqrt(10, 0));

[実行結果]
----------        (← n が 1 で、2 の平方根を求める場合)
0: 50.010000
50.010000
----------        (← n が 3 で、2 の平方根を求める場合)
0: 50.010000
1: 25.024996
2: 12.552458
12.552458
----------        (← n が 15 で、10 の平方根を求める場合)
0: 50.050000
1: 25.124900
2: 12.761456
3: 6.772533
4: 4.124543
5: 3.274527
6: 3.164202
7: 3.162278
8: 3.162278
```

```
9: 3.162278
10: 3.162278
11: 3.162278
12: 3.162278
13: 3.162278
14: 3.162278
3.162278
----------        (← n が 0 で、10 の平方根を求める場合)
0: 50.050000
1: 25.124900
2: 12.761456
3: 6.772533
4: 4.124543
5: 3.274527
6: 3.164202
7: 3.162278
8: 3.162278
3.162278
```

4 「0 または 1 の 2 種類のみの入力を先頭から読み込み、以下の条件をどれか 1 つ満たすならば受理」するオートマトンを考える。

- 入力された 0 と 1 の個数がどちらも奇数個ならば受理する
- 入力された 0 と 1 の個数がどちらも偶数個ならば受理する
- それ以外は受理しない

このオートマトンの状態遷移図は以下のようになった。



この状態遷移図に基づいた動作をするプログラムを作成しなさい。main の例とその実行結果は以下のようになる。

```
[実行結果]
$ ./a.out                    (← 0 と 1 がどちらも偶数個の場合)
数字を入力してください。
0101
読み込んだ数値 : 0 遷移先 : 2
読み込んだ数値 : 1 遷移先 : 3
読み込んだ数値 : 0 遷移先 : 1
読み込んだ数値 : 1 遷移先 : 0
受理する。
$ ./a.out                    (← 0 が奇数個、1 が偶数個の場合)
数字を入力してください。
101
```

```
読み込んだ数値 : 1 遷移先 : 1
読み込んだ数値 : 0 遷移先 : 3
読み込んだ数値 : 1 遷移先 : 2
受理しない。
$ ./a.out                    (← 0 が偶数個、1 が奇数個の場合)
数字を入力してください。
001
読み込んだ数値 : 0 遷移先 : 2
読み込んだ数値 : 0 遷移先 : 0
読み込んだ数値 : 1 遷移先 : 1
受理しない。
$ ./a.out                    (← 0 と 1 がどちらも奇数個の場合)
数字を入力してください。
01
読み込んだ数値 : 0 遷移先 : 2
読み込んだ数値 : 1 遷移先 : 3
受理する。
```

（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  ouyou2mid  「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  ouyou2mid  test1.c
$ ~kogai/report  ouyou2mid  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 93.3 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

```
#include <stdio.h>

void show(int value[], int size);
void swap(int *x, int *y);
void bubble(int value[], int size);

void show(int value[], int size)
{
    int i;
    for(i=0; i<size; i++) {
        printf("%d ", value[i]);
    }
    printf("\n");
}

void swap(int *x, int *y)
{
    int tmp;
    tmp = *x;
    *x = *y;
    *y = tmp;
}

void bubble(int value[], int size)
{
    int i, j, n;
    n = size-1;
    for(i=0; i<size-1; i++) {
        for(j=0; j<n; j++) {
            if(value[j] < value[j+1]) {
                swap(&value[j], &value[j+1]);
            }
        }
        n--;
    }
}

int main(void)
{
    int v[10] = {18, 12, 13, 17, 15,
                 16, 10, 14, 19, 11};
    show(v, 10);
    bubble(v, 10);
    show(v, 10);

    return 0;
}
```

■問 2

```
#include <stdio.h>

void show2(int value[], int left, int right);
void swap(int *x, int *y);
void quick(int value[], int left, int right);

void show2(int value[], int left, int right)
{
    int i;
    for(i=left; i<=right; i++) {
        printf("%d ", value[i]);
    }
    printf("\n");
}

void swap(int *x, int *y)
{
    int tmp;
    tmp = *x;
    *x = *y;
    *y = tmp;
}

void quick(int value[], int left, int right)
{
    int x = value[(left+right)/2];
    int pl = left;
    int pr = right;

    printf("left: %d, right: %d → ", left, right);
    show2(value, left, right);//
    while(1) {
        while(value[pl] < x) pl++;
        while(value[pr] > x) pr--;
        if(pl > pr) {
            break;
        }
        if(pl <= pr) {
            swap(&value[pl], &value[pr]);
            pl++;
            pr--;
        }
    }
    if(left < pr) quick(value, left, pr);
    if(pl < right) quick(value, pl, right);
}

int main(void)
{
    int v[10] = {9, 7, 6, 0, 1, 5, 2, 3, 4, 8};
    show2(v, 0, 9);
    quick(v, 0, 9);
    show2(v, 0, 9);

    return 0;
}
```

■問 3

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

```
double mysqrt(double c, int n)
{
    int i;
    double x1, x2;
    if(n>0) {
        x1 = 100;
        for(i=0; i<n; i++) {
            x2 = x1 - (x1*x1 - c) / (2*x1);
            x1 = x2;
            printf("%d: %lf\n", i, x2);
        }
        return x2;
    } else {
        x1 = 100;
        for(i=0; 1; i++) {
            x2 = x1 - (x1*x1 - c) / (2*x1);
            printf("%d: %lf\n", i, x2);
            if(x1-x2<0.0001) break;
            x1 = x2;
        }
        return x2;
    }
}

int main(void)
{
    printf("----------\n");
    printf("%lf\n", mysqrt(2, 1));
    printf("----------\n");
    printf("%lf\n", mysqrt(2, 3));
    printf("----------\n");
    printf("%lf\n", mysqrt(10, 15));
    printf("----------\n");
    printf("%lf\n", mysqrt(10, 0));

    return 0;
}
```

■問 4

```
#include <stdio.h>

int isaccept(int c, int fin_states[]);

int isaccept(int c, int fin_states[])
{
    int i, result;
    result = 0;
    for(i=0; fin_states[i] != -1; i++) {
        if(fin_states[i]==c) return 1;
    }
    return 0;
}

int main(void)
{
    char input[100];     //入力を格納する配列
    int i=0;             //入力数をカウントする
    //状態は、0: 0/1 偶数個、1: 0 偶数個, 1 奇数個
    //        2: 0 奇数個, 1 偶数個, 3: 0/1 奇数個
    int current_state=0; //現在の状態（初期状態）
    int fin_state=1;     //終了状態
    int fin_states[3] = {0, 3, -1};
    printf("数字を入力してください。\n");
    scanf("%s", input);
    while(input[i]!='\0') {
        switch(current_state) {
        case 0:
            if(input[i]=='0') current_state = 2;
            if(input[i]=='1') current_state = 1;
            break;
        case 1:
            if(input[i]=='0') current_state = 3;
            if(input[i]=='1') current_state = 0;
            break;
        case 2:
            if(input[i]=='0') current_state = 0;
            if(input[i]=='1') current_state = 3;
            break;
        case 3:
            if(input[i]=='0') current_state = 1;
            if(input[i]=='1') current_state = 2;
            break;
        }
        printf("読み込んだ数値 : %c 遷移先 : %d\n",
               input[i], current_state);
        i++;
    }
    if(isaccept(current_state, fin_states)) {
        printf("受理する。\n");
    } else {
        printf("受理しない。\n");
    }
    return 0;
}
```