プログラミング II 前期期末試験

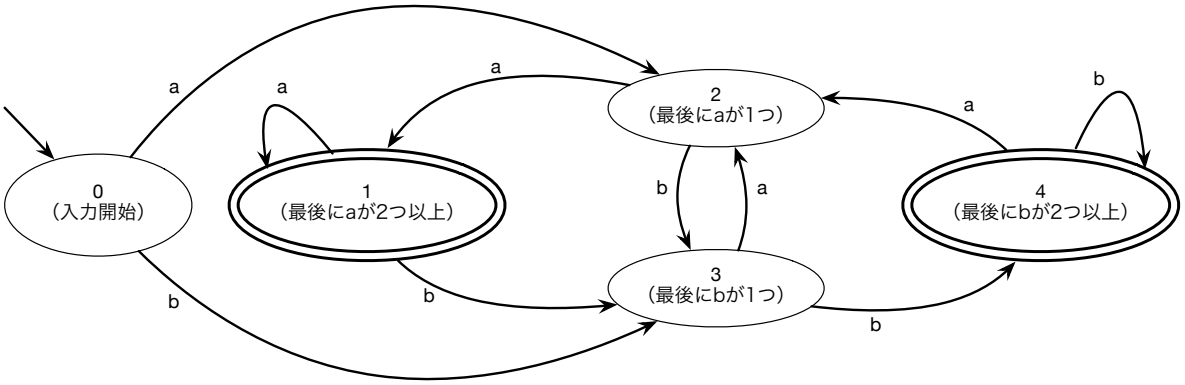
準備　プログラムを作る前に、以下の操作をしてファイルの準備をしておくこと。

1. 授業の配付資料を全てダウンロードする場合は、以下を実行する（既に実行済みの場合は不要）
   `$ mygitclone-p2`
2. GitHub から自分のリポジトリを clone しておく（既に実行済みの場合は不要）
   `$ mygitclone （自分の GitHub ユーザ名）`
3. リポジトリのフォルダに移動して、設定用のスクリプトを実行する
   `$ cd ~/prog3i-(ユーザ名)`
   `$ ./myconf`
4. 今回の定期試験用のフォルダをコピーする
   `$ cp -r /usr/local/common/kogai/p2/test1term .`（←ここにピリオド）
   `$ cd test1term`　（コピーしたフォルダに移動する）
   `$ ls`　（フォルダ内のファイルを確認すると、以下のファイルがコピーされている）
   `1_ab.c  2_array.lex  2_array.yacc  2_myproc.h  2_myproc.c  3_newton.c  4_err.c`
5. テキストエディタでプログラムを開き、**まず先頭行に C のコメントとして自分の番号と名前を書いて**、解答を始める
   `$ gedit` 各問のファイル名 `&`

1　'a' または'b' の 2 種類のみの文字を先頭から読み込み、「最後に 2 つ以上同じ文字が連続した入力を受理する」オートマトンを考える。入力例を以下に示す。

- 受理される入力例："abaa", "babb", "baaa", "aabbb"
- 受理されない入力例："abaab", "babba"

このオートマトンの状態遷移図は以下のようになった。（状態の中には、状態を識別するための整数と括弧内にその状態の説明が示されている。）



この状態遷移図に基づいた動作をするプログラムをファイル「1_ab.c」に作成しなさい。また、関数 isaccept() が同ファイル内に定義済みなので利用してよい。

実行例は以下のようになる。実行中に、「読み込んだ文字と遷移先」の出力は入れても構わない。

```
[実行結果]
$ ./a.out                    (←最後に a が 2 つ続いた場合)
a または b を入力してください。
abaa
受理する。
$ ./a.out                    (←最後に b が 2 つ続いた場合)
a または b を入力してください。
```

```
babb
受理する。
$ ./a.out                    (←最後に a が 3 つ続いた場合)
a または b を入力してください。
baaa
受理する。
$ ./a.out                    (←最後に b が 3 つ続いた場合)
a または b を入力してください。
aabbb
受理する。
$ ./a.out                    (←最後に b が 1 つの場合)
a または b を入力してください。
abaab
受理しない。
$ ./a.out                    (←最後に a が 1 つの場合)
a または b を入力してください。
babba
受理しない。
```

2 整数の配列に対して操作するコマンドを実行するプログラムが以下のファイルとして用意されている。

- 字句解析の定義「2_array.lex」
- 構文解析の定義「2_array.yacc」
- ヘッダファイル「2_myproc.h」
- 関数定義「2_myproc.c」

以下に従って、これに「配列内の指定した値を全て別の値に置き換えて、置き換えた要素数を出力する」コマンド replace を追加しなさい。

- コマンド「replace」の字句定義の識別子は REPLACE とする
- コマンド replace の後には、NUMBER が 2 個続く
- この演算で呼び出す関数のプロトタイプ宣言は以下のようにする `int replace(int x, int y);`
- このコマンドで呼び出される関数 replace の定義は、以下のような処理となる
  1. 配列 a の全要素に対する繰り返し処理の中で、i 番目の値と x を比較して、等しい場合は i 番目に y を代入する
  2. 置き換えた要素数をカウントして、最後に出力する
  3. 最後に 0 を戻す

```
[実行結果]（実際に出力される空行は省略している）
$ ./a.out
set 0 2                  (←コマンド set でいくつか値を変更する)
set 2 6
set 3 9
set 8 2
set 9 2
show                     (← set 後の配列を確認する)
2 0 6 9 0 0 0 0 2 2
replace 2 1              (←配列内の 2 を全て 1 に変更する)
  置き換えた要素数: 3
show                     (← replace 後の配列を確認する)
1 0 6 9 0 0 0 0 1 1
exit
```

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() が、以下のような動作をするようなプログラムをファイル「3_newton.c」に作成しなさい。

- 仮引数 d で与えられた数値を用いて、x1 と x2 の差分が d 未満になったら近似値を求める繰り返し処理を終了する
- 近似値を求める繰り返し処理の中で、「x1, x2, x1 と x2 の差分」の 3 つの情報が分かるように出力する
- x1 と x2 の差分は常に正の値となるものとし、仮引数 d は常に正の値が与えられるものとしてよい

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, double d);
  /* d の値を用いて、平方根を求める処理が上記となるようにニュートン法で計算する */
```

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(3, 1.0));
  printf("----------\n");
  printf("%lf\n", mysqrt(3, 0.1));
  printf("----------\n");
  printf("%lf\n", mysqrt(3, 0.01));
[実行結果]
----------                          (← d が 1.0 で、3 の平方根を求める場合)
x1: 100.000000, x2: 50.015000, x1-x2: 49.985000
x1: 50.015000, x2: 25.037491, x1-x2: 24.977509
x1: 25.037491, x2: 12.578656, x1-x2: 12.458835
x1: 12.578656, x2: 6.408577, x1-x2: 6.170078
x1: 6.408577, x2: 3.438350, x1-x2: 2.970227
x1: 3.438350, x2: 2.155431, x1-x2: 1.282919
x1: 2.155431, x2: 1.773632, x1-x2: 0.381799
1.773632
----------                          (← d が 0.1 で、3 の平方根を求める場合)
x1: 100.000000, x2: 50.015000, x1-x2: 49.985000
x1: 50.015000, x2: 25.037491, x1-x2: 24.977509
x1: 25.037491, x2: 12.578656, x1-x2: 12.458835
x1: 12.578656, x2: 6.408577, x1-x2: 6.170078
x1: 6.408577, x2: 3.438350, x1-x2: 2.970227
x1: 3.438350, x2: 2.155431, x1-x2: 1.282919
x1: 2.155431, x2: 1.773632, x1-x2: 0.381799
x1: 1.773632, x2: 1.732538, x1-x2: 0.041094
1.732538
----------                          (← d が 0.01 で、3 の平方根を求める場合)
x1: 100.000000, x2: 50.015000, x1-x2: 49.985000
x1: 50.015000, x2: 25.037491, x1-x2: 24.977509
x1: 25.037491, x2: 12.578656, x1-x2: 12.458835
x1: 12.578656, x2: 6.408577, x1-x2: 6.170078
x1: 6.408577, x2: 3.438350, x1-x2: 2.970227
x1: 3.438350, x2: 2.155431, x1-x2: 1.282919
x1: 2.155431, x2: 1.773632, x1-x2: 0.381799
x1: 1.773632, x2: 1.732538, x1-x2: 0.041094
x1: 1.732538, x2: 1.732051, x1-x2: 0.000487
1.732051
```

4 「丸め誤差」の動作を確認するプログラムを作るために、以下のような処理をファイル「4_err.c」に作成した。

```
#include <stdio.h>

int main(void)
{
    int n, i;
    float x = 0.1, result;
    printf("n > ");
    scanf("%d", &n);
    result = n;
    /* ----- ここに繰り返し処理を作る ----- */
    printf("result: %f\n", result);
    return 0;
}
```

この時、変数 x には 0.1 が代入されているので、数式上では「$n - n * 10 * x$」は 0 となるはずだが、「$n * 10$ 回、n から x を引く処理を繰り返す」と、変数 x の丸め誤差が原因で結果は 0 とはならず、変数 result の出力が 0 にはならない事が確認できる。この繰り返し処理を作り、このプログラムを完成させなさい。実行例は以下のようになる。

```
[実行結果]（n が大きくなると、つまり、繰返し回数が増えると誤差も大きくなる）
$ ./a.out            （← n が 5 の場合）
n > 5
result: 0.000003
$ ./a.out            （← n が 50 の場合）
n > 50
result: 0.000188
$ ./a.out            （← n が 500 の場合）
n > 500
result: -0.021320
```

問題はここまで

（各 25 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
  （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、Web ブラウザやネットワークを介した情報の参照、自分の PC を使用する、など）
  - 試験中（開始 5 分後〜開始 60 分後）は、演習室外へのネットワークアクセスは遮断される
  - GitHub への提出のためのコマンドに限ってネットワーク利用が可能（それ以外は不正行為とする）

答案の提出

1. **提出する全ての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**
2. 端末内で、以下のコマンドで課題を提出

   ```
   $ git add -A
   $ git commit -m "前期期末提出"
   $ git push origin master
   ```

   （push が成功すると「done」が含まれたメッセージが表示される）
3. 提出が完了しているかを確認したい人は声をかけて下さい。（その場で教員側の画面で確認します）

模範解答（試験時間 90 分, 平均 86.4 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>

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];
    int i=0;
    int current_state=0;
    int fin_states[3] = {1, 4, -1};
    printf("a または b 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]=='a') {
                current_state = 2;
            } else {
                current_state = 3;
            }
            break;
        case 1:
            if(input[i]=='a') {
                current_state = 1;
            } else {
                current_state = 3;
            }
            break;
        case 2:
            if(input[i]=='a') {
                current_state = 1;
            } else {
                current_state = 3;
            }
            break;
        case 3:
            if(input[i]=='a') {
                current_state = 2;
            } else {
                current_state = 4;
            }
            break;
        case 4:
            if(input[i]=='a') {
                current_state = 2;
            } else {
                current_state = 4;
            }
            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;
}
```

■問 2

```
//---------------------------
//2_array.lex
//---------------------------
%%
"clear" return CLEAR;
"exit" return EXIT;
"show" return SHOW;
"set"  return SET;
"replace"   return REPLACE;
"\n" return NL;

[0-9]+ {
  yylval = atoi(yytext);
  return NUMBER;
}
%%
//---------------------------
//2_array.yacc
//---------------------------
%token NL NUMBER
%token CLEAR EXIT SHOW SET REPLACE //追加
%%
list :
     | list cmd NL       { printf("\n"); }
     ;
cmd  : CLEAR             { clear(); }
     | EXIT              { exit(0); }
     | SHOW              { show(); }
     | SET NUMBER NUMBER { set($2, $3); }
```

```
     | REPLACE NUMBER NUMBER { replace($2, $3); }
     ;
%%
#include <stdio.h>
#include "lex.yy.c"
#include "2_myproc.h"
//---------------------------
//2_myproc.h
//---------------------------
int a[10];

int clear();
int show();
int set(int i, int n);
int replace(int x, int y);
//---------------------------
//2_myporc.c
//---------------------------
#include <stdio.h>
#include "2_myproc.h"

int clear()
{
    int i;
    for(i=0; i<10; i++) {
        a[i] = 0;
    }
    return 0;
}

int show(void)
{
    int i;
    for(i=0; i<10; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
    return 0;
}

int set(int i, int n)
{
    a[i] = n;
    return 0;
}

int replace(int x, int y)
{
    int i;
    int count;
    count = 0;
    for(i=0; i<10; i++) {
        if(a[i]==x) {
            a[i] = y;
            count++;
        }
    }
    printf("置き換えた要素数: %d\n", count);
    return 0;
}
```

■問 3

```
#include <stdio.h>

double mysqrt(double c, double d);

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

int main(void)
{
    printf("----------\n");
    printf("%lf\n", mysqrt(3, 1.0));
    printf("----------\n");
    printf("%lf\n", mysqrt(3, 0.1));
    printf("----------\n");
    printf("%lf\n", mysqrt(3, 0.01));

    return 0;
}
```

■問 4

```
#include <stdio.h>

int main(void)
{
    int n, i;
    float x = 0.1, result;
    printf("n > ");
    scanf("%d", &n);
    result = n;
    for(i=0; i<10*n; i++) {
        result -= x;
    }
    printf("result: %f\n", result);

    return 0;
}
```