プログラミング応用 後期期末試験

※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。

準備 この試験で使用するテンプレートファイル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、テンプレートファイルが入ったフォルダをコピーしておくこと。

```
$  cp  -r /usr/local/common/kogai/ouyou201902  .  (←ここにピリオド)
$  cd ouyou201902      (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、ファイルがコピーされている)
1_calc.lex      2_array.lex     2_myproc.c      3_4_uml.asta    5_server.c
1_calc.yacc     2_array.yacc    2_myproc.h      5_client.c      todo.java
```

1 加算と減算の処理をするプログラムの字句解析「1_calc.lex」と構文解析「1_calc.yacc」が用意されている。以下に従って、これに「左シフト演算」「右シフト演算」を追加しなさい。

- 左シフトの演算子は「<<」、字句定義の識別子は LSF とする（つまり、演算子は C 言語と同じ）
- 右シフトの演算子は「>>」、字句定義の識別子は RSF とする（つまり、演算子は C 言語と同じ）
- これらの演算子を追加すると構文は以下のようになる。

```
式 ::= 式 + 式
   ::= 式 - 式
   ::= 式 << 式
   ::= 式 >> 式
   ::= ( 式 )
   ::= 数値
```

実行結果は以下のようになる。

```
[実行結果]
 $ ./a.out
 2<<1       (← 2 を 1 つ左シフトする)
 4          (←その結果、2 が 2 倍された)
 2<<3       (← 2 を 3 つ左シフトする)
 16         (←その結果、2 が 8 倍（2 の 3 乗倍）された)
 8>>1       (← 8 を 1 つ右シフトする)
 4          (←その結果、2 が 1/2 倍された)
 8>>2       (← 8 を 2 つ右シフトする)
 2          (←その結果、2 が 1/4 倍（1/2 の 2 乗倍）された)
```

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」

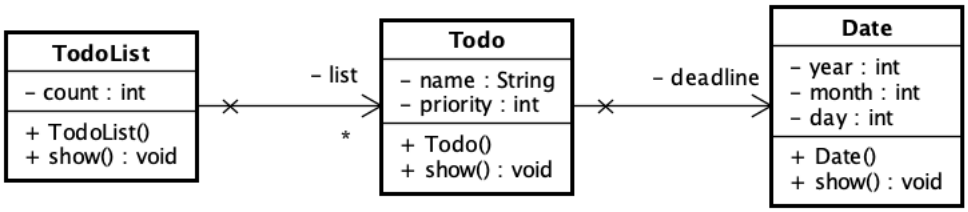
以下に従って、これに「配列内の指定した値を全て別の値に置き換える」コマンド replace を追加しなさい。

- 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
  配列 a の 0〜9 番目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
  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
show                    (←初期値を確認する)
0 0 0 0 0 0 0 0 0 0
set 0 3                 (←コマンド set でいくつか値を変更する)
set 2 5
set 3 8
set 7 3
set 9 3
show                    (← set 後の配列を確認する)
3 0 5 8 0 0 0 3 0 3
replace 3 1             (←配列内の 3 を全て 1 に変更する)
show                    (← replace 後の配列を確認する)
1 0 5 8 0 0 0 1 0 1
exit
```

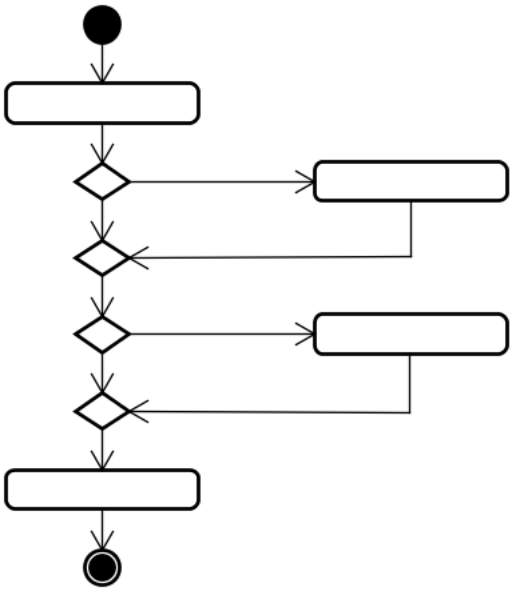
3 Java プログラム「todo.java」が予め用意されている。astah を起動して、用意されたファイル「3_4_uml.asta」を開き、用意されたクラス図「todo」に、この Java プログラムを読み込んで以下のようなクラス図を作成しなさい。
作成したクラス図には「ノート」を貼り自分の番号と名前を入力しておくこと。



4 次のような C プログラムの関数 tri() を考える。

```
int tri(int x, int y, int z)
{
    int r;
    r = x;
    if(y > r) r = y;
    if(z > r) r = z;
    return r;
}
```

astah ファイル「3_4_uml.asta」に、関数 tri() のアクティビティ図の一部が以下のように用意されている。



このアクティビティ図に「アクション」と「ガード」を追加して完成させなさい。ただし、変数宣言はアクションとして描く必要はない。

作成したアクティビティ図には「ノート」を貼り自分の番号と名前を入力しておくこと。

5 ソケットを使ったネットワークプログラムのクライアント「5_client.c」とサーバ「5_server.c」が用意されている。(構造体 Msg を使って送受信するプログラム)

このサーバ「5_server.c」に対して、クライアントから受信した構造体に以下のような処理をして送信する機能を追加しなさい。(サーバのみの変更で完成する)

- 受信した構造体のメンバ num の値が偶数の場合、
  - 受信した構造体のメンバ message の文字列の末尾に「"偶数"」を結合する
  - 受信した構造体のメンバ num の値を 0 にする
- 受信した構造体のメンバ num の値が奇数の場合、
  - 受信した構造体のメンバ message の文字列の末尾に「"奇数"」を結合する
  - 受信した構造体のメンバ num の値を 1 にする

受信した構造体のメンバ num の値が偶数の場合の実行結果

```
[実行結果](サーバ側:IP アドレスが 192.168.3.15 の場合)
  $ ./server
  ポート番号 >5000
  クライアントの受付を開始しました。

  Handling client 192.168.3.15
  受信した文字列: 結果?   (←受信した文字列は「結果?」)
  受信した整数: 8         (←受信した整数は「8」)
```

```
[実行結果](クライアント側)


  $ ./client 192.168.3.15
  ポート番号 > 5000
  整数 > 8

  受信した文字列: 結果? 偶数  (←「偶数」が追加された)
  受信した整数: 0             (←整数が 0 になった)
```

受信した構造体のメンバ num の値が奇数の場合の実行結果

[実行結果]（サーバ側：IP アドレスが 192.168.3.15 の場合）

```
$ ./server
ポート番号 >5000
クライアントの受付を開始しました。

Handling client 192.168.3.15
受信した文字列: 結果？　（←受信した文字列は「結果？」）
受信した整数: 7　　（←受信した整数は「7」）
```

[実行結果]（クライアント側）

```
$ ./client 192.168.3.15
ポート番号 > 5000
整数 > 7

受信した文字列: 結果？ 奇数　（←「奇数」が追加された）
受信した整数: 1　　（←整数が 1 になった）
```

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 ouyou2term 「プログラムファイル」
```

- 今回の試験は以下の**8 個のファイル**を提出すること

```
$ ~kogai/report ouyou2term 1_calc.lex
$ ~kogai/report ouyou2term 1_calc.yacc
$ ~kogai/report ouyou2term 2_array.lex
$ ~kogai/report ouyou2term 2_array.yacc
$ ~kogai/report ouyou2term 2_myproc.c
$ ~kogai/report ouyou2term 2_myproc.h
$ ~kogai/report ouyou2term 3_4_uml.asta
$ ~kogai/report ouyou2term 5_server.c　（問 5 はクライアントの提出は不要）
```

- 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

後期期末試験 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

```
//--------------------
//1_calc.lex
//--------------------
/* ここに番号と名前 */
%%
"+" return ADD;
"-" return SUB;
"<<" return LSF;
">>" return RSF;
"(" return LP;
")" return RP;
"\n" return NL;

[0-9]+ {
  yylval = atoi(yytext);
  return NUMBER;
}
%%
//--------------------
//1_calc.yacc
//--------------------
/* ここに番号と名前 */
%token NL LP RP NUMBER
%token ADD SUB LSF RSF
%%
list :
     | list expr NL { printf("%d\n", $2);}
     ;
expr : expr ADD expr { $$ = $1 + $3;}
     | expr SUB expr { $$ = $1 - $3;}
     | expr LSF expr { $$ = $1 << $3;}
     | expr RSF expr { $$ = $1 >> $3;}
     | LP expr RP    { $$ = $2;}
     | NUMBER        { $$ = $1;}
     ;
%%
#include "lex.yy.c"
```

■問 2

```
//--------------------
//2_array.lex
//--------------------
/* ここに番号と名前 */
%%
"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
//--------------------
/* ここに番号と名前 */
%token NL NUMBER
%token CLEAR EXIT SHOW SET REPLACE
%%
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
//--------------------
/* ここに番号と名前 */
int a[10];

int clear();
int show();
int set(int i, int n);
int replace(int x, int y);
//--------------------
//2_myproc.c
//--------------------
/* ここに番号と名前 */
#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;
    for(i=0; i<10; i++) {
```

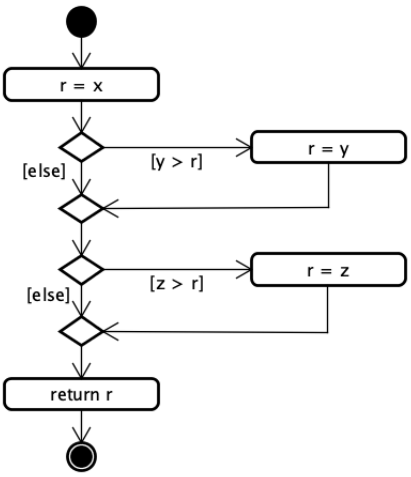
```
        if(a[i]==x) {
            a[i] = y;
        }
    }
    return 0;
}
```

■問 3

astah にて、以下の手順で操作する

1.「ツール」メニューから「Java」を選択する
2.「Java ソースコードの読み込み」を選択する
3. ファイル選択画面で「todo.java」を選択する
4.「todo.java」を候補リストから選択リストへ移動し「了解」を押す
5. 関連にしたい属性として、「deadline」「list」のチェックを付ける
6. クラス図を開き、左の構造ツリーから、クラス Date, Todo, TodoList を編集画面へドラッグ&ドロップする

■問 4



■問 5

```
//--------------------
//5_server.c
//--------------------
/* ここに番号と名前 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* ソケット用にインクルードする */
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

typedef struct Msg {
    char message[50];
    int num;
} Msg;

/* プロトタイプ宣言 */
void showError(char *message);

/* エラー処理のための関数の定義 */
void showError(char *message)
{
    fprintf(stderr, "%s", message);
    fprintf(stderr, "\n");
    exit(1);
}

int main(void)
{
    struct sockaddr_in sAddr;
    struct sockaddr_in cAddr;
    int sSock;
    int cSock;
    int sPort;
    int result, cLen;
    Msg msg1, msg2;
    /* サーバのポート番号を入力する */
    printf("ポート番号 >");
    scanf("%d", &sPort);
    /* (1) TCP とアプリケーション間の通信路を作る */
    sSock =socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (sSock < 0) {
        showError("socket() failed");
    }
    sAddr.sin_family = AF_INET;
    sAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    sAddr.sin_port = htons(sPort);
    /* (2) ポート番号と IP アドレスを指定する */
    result = bind(sSock, (struct sockaddr *)&sAddr,
                  sizeof(struct sockaddr));
    if (result < 0) {
        showError("bind() failed.");
    }
    /* (3) 接続の受付を開始する */
    result = listen(sSock, 5);
    if (result < 0) {
        showError("listen() failed.");
    }
    printf("クライアントの受付を開始しました。\n");
    /* クライアントからの受付を待ち続ける */
    while(1) {
        cLen = sizeof(struct sockaddr);
        /* (4) 受付けた接続用のソケットを作る */
        cSock =accept(sSock, (struct sockaddr *)&cAddr,
                      &cLen);
        if (cSock < 0) {
            showError("accept() failed");
        }
        printf("Handling client %s\n",
               inet_ntoa(cAddr.sin_addr));
        /* (5) クライアントから要求を受信する */
        recv(cSock, &msg1, sizeof(Msg), 0);
        /* 受信した文字列データを表示する */
        printf("受信した文字列: %s \n", msg1.message);
        printf("受信した整数: %d \n", msg1.num);

        if(msg1.num % 2 == 0) {
            strcat(msg1.message, "偶数");
            msg1.num = 0;
        }
        if(msg1.num % 2 == 1) {
            strcat(msg1.message, "奇数");
            msg1.num = 1;
        }

        /* (6) クライアントに応答メッセージを送信する */
        send(cSock, &msg1, sizeof(Msg), 0);
        /* クライアント用ソケットを閉じる */
        close(cSock);
    }
    return 0;
}
```