din\_philo.c

[ryonosuke\_araki@RyonosukenoMacBook-Air 13:27:37] $ cat din\_philo.c [ ~/OneDrive - The University of Tokyo/soubunkiso2 ]

#include <pthread.h>

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <assert.h>

#define PHILOS 5

#define DELAY 5000

#define FOOD 50

void \*philosopher (void \*id);

void grab\_chopstick (int, int, char \*);

void down\_chopsticks (int, int);

int food\_on\_table ();

pthread\_mutex\_t chopstick[PHILOS];

pthread\_t philo[PHILOS];

pthread\_mutex\_t food\_lock;

int sleep\_seconds = 0;

int main (int argn, char \*\*argv)

{

int i;

if (argn == 2)

sleep\_seconds = atoi (argv[1]);

pthread\_mutex\_init (&food\_lock, NULL);

for (i = 0; i < PHILOS; i++)

pthread\_mutex\_init (&chopstick[i], NULL);

for (i = 0; i < PHILOS; i++)

pthread\_create (&philo[i], NULL, philosopher, (void \*)i);

for (i = 0; i < PHILOS; i++)

pthread\_join (philo[i], NULL);

return 0;

}

void \*philosopher (void \*num)

{

int id;

int i, left\_chopstick, right\_chopstick, f;

id = (int)num;

printf("Philosopher %d is done thinking and now ready to eat.\n", id);

right\_chopstick = id;

left\_chopstick = id + 1;

/\* 箸が一巡した \*/

if (left\_chopstick == PHILOS)

left\_chopstick = 0;

while (f = food\_on\_table ()) {

/\* 箸を手に取る前にうたた寝をする哲学者 1 のおかげで、ほかの

\* 哲学者達は、デッドロックに陥ることなく食事をすることが

\* できます。 \*/

if (id == 1)

sleep (sleep\_seconds);

grab\_chopstick (id, right\_chopstick, "right ");

grab\_chopstick (id, left\_chopstick, "left");

printf ("Philosopher %d: eating.\n", id);

usleep (DELAY \* (FOOD - f + 1));

down\_chopsticks (left\_chopstick, right\_chopstick);

}

printf ("Philosopher %d is done eating.\n", id);

return (NULL);

}

int food\_on\_table ()

{

static int food = FOOD;

int myfood;

pthread\_mutex\_lock (&food\_lock);

if (food > 0) {

food--;

}

myfood = food;

pthread\_mutex\_unlock (&food\_lock);

return myfood;

}

void

grab\_chopstick (int phil, int c, char \*hand)

{

pthread\_mutex\_lock (&chopstick[c]);

printf ("Philosopher %d: got %s chopstick %d\n", phil, hand, c);

}

void

down\_chopsticks (int c1, int c2)

{

pthread\_mutex\_unlock (&chopstick[c1]);

pthread\_mutex\_unlock (&chopstick[c2]);

}

din\_philoの実行結果

[ryonosuke\_araki@RyonosukenoMacBook-Air 13:17:54] $ cc din\_philo.c [ ~/OneDrive - The University of Tokyo/soubunkiso2 ]

din\_philo.c:33:55: warning: cast to 'void \*' from smaller integer type 'int' [-Wint-to-void-pointer-cast]

pthread\_create (&philo[i], NULL, philosopher, (void \*)i);

^

din\_philo.c:53:14: warning: using the result of an assignment as a condition without parentheses [-Wparentheses]

while (f = food\_on\_table ()) {

~~^~~~~~~~~~~~~~~~~~

din\_philo.c:53:14: note: place parentheses around the assignment to silence this warning

while (f = food\_on\_table ()) {

^

( )

din\_philo.c:53:14: note: use '==' to turn this assignment into an equality comparison

while (f = food\_on\_table ()) {

^

==

2 warnings generated.

[ryonosuke\_araki@RyonosukenoMacBook-Air 13:19:34] $ ./a.out [ ~/OneDrive - The University of Tokyo/soubunkiso2 ]

Philosopher 0 is done thinking and now ready to eat.

Philosopher 1 is done thinking and now ready to eat.

Philosopher 2 is done thinking and now ready to eat.

Philosopher 2: got right chopstick 2

Philosopher 4 is done thinking and now ready to eat.

Philosopher 4: got right chopstick 4

Philosopher 1: got right chopstick 1

Philosopher 3 is done thinking and now ready to eat.

Philosopher 2: got left chopstick 3

Philosopher 2: eating.

Philosopher 0: got right chopstick 0

Philosopher 2: got right chopstick 2

Philosopher 3: got right chopstick 3

^Z

zsh: suspended ./a.out

[ryonosuke\_araki@RyonosukenoMacBook-Air 13:24:02] $

トークンを用いたシステムを使用して、哲学者がトークンを受け取ってから食事をとるようにする。用意するトークンの数は、席についている哲学者の人数より少ない。哲学者はトークンを受け取ったあと、テーブルの規則に従って食事をとることができる。食事をしたあと、哲学者はトークンを返却し、プロセスを繰り返す。

din\_philo\_fix.c

[ryonosuke\_araki@RyonosukenoMacBook-Air 13:49:17] $ cat din\_philo\_fix.c [ ~/OneDrive - The University of Tokyo/soubunkiso2 ]

#include <pthread.h>

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <assert.h>

#define PHILOS 5

#define DELAY 5000

#define FOOD 50

void \*philosopher (void \*id);

void grab\_chopstick (int, int, char \*);

void down\_chopsticks (int, int);

int food\_on\_table ();

int get\_token ();

void return\_token ();

pthread\_mutex\_t chopstick[PHILOS];

pthread\_t philo[PHILOS];

pthread\_mutex\_t food\_lock;

pthread\_mutex\_t num\_can\_eat\_lock;

int sleep\_seconds = 0;

uint32\_t num\_can\_eat = PHILOS - 1;

int main (int argn, char \*\*argv)

{

int i;

pthread\_mutex\_init (&food\_lock, NULL);

pthread\_mutex\_init (&num\_can\_eat\_lock, NULL);

for (i = 0; i < PHILOS; i++)

pthread\_mutex\_init (&chopstick[i], NULL);

for (i = 0; i < PHILOS; i++)

pthread\_create (&philo[i], NULL, philosopher, (void \*)i);

for (i = 0; i < PHILOS; i++)

pthread\_join (philo[i], NULL);

return 0;

}

void \*philosopher (void \*num)

{

int id;

int i, left\_chopstick, right\_chopstick, f;

id = (int)num;

printf ("Philosopher %d is done thinking and now ready to eat.\n", id);

right\_chopstick = id;

left\_chopstick = id + 1;

/\* 箸が一巡した \*/

if (left\_chopstick == PHILOS)

left\_chopstick = 0;

while (f = food\_on\_table ()) {

get\_token ();

grab\_chopstick (id, right\_chopstick, "right ");

grab\_chopstick (id, left\_chopstick, "left");

printf ("Philosopher %d: eating.\n", id);

usleep (DELAY \* (FOOD - f + 1));

down\_chopsticks (left\_chopstick, right\_chopstick);

return\_token ();

}

printf ("Philosopher %d is done eating.\n", id);

return (NULL);

}

int food\_on\_table ()

{

static int food = FOOD;

int myfood;

pthread\_mutex\_lock (&food\_lock);

if (food > 0) {

food--;

}

myfood = food;

pthread\_mutex\_unlock (&food\_lock);

return myfood;

}

void grab\_chopstick (int phil, int c, char \*hand)

{

pthread\_mutex\_lock (&chopstick[c]);

printf ("Philosopher %d: got %s chopstick %d\n", phil, hand, c);

}

void down\_chopsticks (int c1, int c2)

{

pthread\_mutex\_unlock (&chopstick[c1]);

pthread\_mutex\_unlock (&chopstick[c2]);

}

int get\_token ()

{

int successful = 0;

while (!successful) {

pthread\_mutex\_lock (&num\_can\_eat\_lock);

if (num\_can\_eat > 0) {

num\_can\_eat--;

successful = 1;

}

else {

successful = 0;

}

pthread\_mutex\_unlock (&num\_can\_eat\_lock);

}

}

void return\_token ()

{

pthread\_mutex\_lock (&num\_can\_eat\_lock);

num\_can\_eat++;

pthread\_mutex\_unlock (&num\_can\_eat\_lock);

}

din\_philo\_fix.cの実行結果

[ryonosuke\_araki@RyonosukenoMacBook-Air 13:48:17] $ ./a.out [ ~/OneDrive - The University of Tokyo/soubunkiso2 ]

Philosopher 1 is done thinking and now ready to eat.

Philosopher 0 is done thinking and now ready to eat.

Philosopher 2 is done thinking and now ready to eat.

Philosopher 3 is done thinking and now ready to eat.

Philosopher 0: got right chopstick 0

Philosopher 3: got right chopstick 3

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 2: got right chopstick 2

Philosopher 4 is done thinking and now ready to eat.

Philosopher 1: got right chopstick 1

Philosopher 3: got right chopstick 3

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 4: got right chopstick 4

Philosopher 2: got left chopstick 3

Philosopher 2: eating.

Philosopher 1: got left chopstick 2

Philosopher 1: eating.

Philosopher 3: got right chopstick 3

Philosopher 2: got right chopstick 2

Philosopher 0: got left chopstick 1

Philosopher 0: eating.

Philosopher 0: got right chopstick 0

Philosopher 0: got left chopstick 1

Philosopher 0: eating.

Philosopher 1: got right chopstick 1

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 3: got right chopstick 3

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 3: got right chopstick 3

Philosopher 4: got right chopstick 4

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 4: got right chopstick 4

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 3: got right chopstick 3

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 4: got right chopstick 4

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 3: got right chopstick 3

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 4: got right chopstick 4

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 2: got left chopstick 3

Philosopher 2: eating.

Philosopher 3: got right chopstick 3

Philosopher 1: got left chopstick 2

Philosopher 1: eating.

Philosopher 2: got right chopstick 2

Philosopher 0: got right chopstick 0

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 1: got right chopstick 1

Philosopher 3: got right chopstick 3

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 2: got left chopstick 3

Philosopher 2: eating.

Philosopher 3: got right chopstick 3

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 1: got left chopstick 2

Philosopher 1: eating.

Philosopher 2: got right chopstick 2

Philosopher 0: got left chopstick 1

Philosopher 0: eating.

Philosopher 4: got right chopstick 4

Philosopher 2: got left chopstick 3

Philosopher 2: eating.

Philosopher 1: got right chopstick 1

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 1: got left chopstick 2

Philosopher 1: eating.

Philosopher 0: got right chopstick 0

Philosopher 4: got right chopstick 4

Philosopher 2: got right chopstick 2

Philosopher 0: got left chopstick 1

Philosopher 0: eating.

Philosopher 2: got left chopstick 3

Philosopher 2: eating.

Philosopher 1: got right chopstick 1

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 3: got right chopstick 3

Philosopher 1: got left chopstick 2

Philosopher 1: eating.

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 0: got right chopstick 0

Philosopher 0: got left chopstick 1

Philosopher 0: eating.

Philosopher 2: got right chopstick 2

Philosopher 4: got right chopstick 4

Philosopher 2: got left chopstick 3

Philosopher 2: eating.

Philosopher 1: got right chopstick 1

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 1: got left chopstick 2

Philosopher 1: eating.

Philosopher 4: got right chopstick 4

Philosopher 0: got right chopstick 0

Philosopher 2: got right chopstick 2

Philosopher 0: got left chopstick 1

Philosopher 0: eating.

Philosopher 2: got left chopstick 3

Philosopher 2: eating.

Philosopher 1: got right chopstick 1

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 3: got right chopstick 3

Philosopher 1: got left chopstick 2

Philosopher 1: eating.

Philosopher 0: got right chopstick 0

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 2: got right chopstick 2

Philosopher 0: got left chopstick 1

Philosopher 0: eating.

Philosopher 3: got right chopstick 3

Philosopher 4: got right chopstick 4

Philosopher 1: got right chopstick 1

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 0: got right chopstick 0

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 3: got right chopstick 3

Philosopher 3: got left chopstick 4

Philosopher 3: eating.

Philosopher 3 is done eating.

Philosopher 4: got right chopstick 4

Philosopher 2: got left chopstick 3

Philosopher 2: eating.

Philosopher 2 is done eating.

Philosopher 1: got left chopstick 2

Philosopher 1: eating.

Philosopher 1 is done eating.

Philosopher 0: got left chopstick 1

Philosopher 0: eating.

Philosopher 0 is done eating.

Philosopher 4: got left chopstick 0

Philosopher 4: eating.

Philosopher 4 is done eating.

din\_philo.java

[ryonosuke\_araki@RyonosukenoMacBook-Air 14:28:01] $ cat din\_philo.java [ ~/OneDrive - The University of Tokyo/soubunkiso2 ]

// 共有オブジェクト

class Fork {

// どの哲学者の左手にあるか識別する番号

private int id;

// いずれかの哲学者の手にとられているかどうか

private boolean eating = false;

// コンストラクタ

Fork(int i) {

// メソッド引数の哲学者の左手に置かれている

id = i;

}

// 手にとられる

public synchronized void pick(int i) {

while (eating == true) {

// 隣の哲学者の手に取られている間は繰り返し

try {

System.out.println(i + " is starving.");

// 待機プールへ

wait();

} catch (InterruptedException e) {

System.out.println(e);

}

}

// 手に取れたら true をセット

eating = true;

}

// 手から離される

public synchronized void down() {

// 食事が済んだら false をセット

eating = false;

// 待機プールのスレッドをロック探索状態に

notifyAll();

}

}

class Philosopher implements Runnable {

int id; // 哲学者の識別番号

int eatTime; // 食事時間

int thinkTime; // 思索時間

int left; // 左手のフォークの番号

int right; // 右手のフォークの番号

Fork[] forks = new Fork[4]; // 共有オブジェクト

// コンストラクタ

Philosopher(int i) {

id = i;

}

// 哲学者のプロパティのセット

public void setProperties(int eating, int thinking, Fork[] objs) {

left = id;

if (id != 0) {

right = id - 1;

} else {

right = 4;

}

eatTime = eating;

thinkTime = thinking;

forks = objs;

}

// 空腹を感じるとフォークを手に取る

public void feelHungry() {

// 左手のフォークを手に取る

forks[left].pick(id);

// ここの待機時間が長いとデッドロック発生

try {

Thread.sleep(500);

} catch (InterruptedException e) {

System.out.println(e);

}

// 右手のフォークを手に取る

forks[right].pick(id);

System.out.println(id + " is eating.");

try {

// 食事中

Thread.sleep(eatTime);

} catch (InterruptedException e) {

System.out.println(e);

}

// 食事終了

// 左手のフォークを離す

forks[left].down();

// 右手のフォークを離す

forks[right].down();

}

// 思索

public void think() {

try {

// 思索中

Thread.sleep(thinkTime);

} catch (InterruptedException e) {

System.out.println(e);

}

}

// スレッドの run() メソッド

public void run() {

while (true) {

// 思索中

System.out.println(id + " is thinking.");

think();

// 空腹を感じる

System.out.println(id + " feels hungry.");

feelHungry();

}

}

}

class Dining {

public static void main(String[] args) {

// 共有オブジェクト

Fork[] forks = new Fork[5];

// 哲学者

Philosopher[] phils = new Philosopher[5];

// インスタンス化

for (int i=0; i<5; i++) {

forks[i] = new Fork(i);

phils[i] = new Philosopher(i);

}

// 哲学者のプロパティ

// eating, thinking, shared object

phils[0].setProperties(2000, 1000, forks);

phils[1].setProperties(1900, 1100, forks);

phils[2].setProperties(1800, 1200, forks);

phils[3].setProperties(1700, 1300, forks);

phils[4].setProperties(1600, 1400, forks);

// スレッド

Thread[] thres = new Thread[5];

for (int i=0; i<5; i++) {

// 哲学者をスレッドに委譲

thres[i] = new Thread(phils[i]);

}

// スレッドの開始

for (int i=0; i<5; i++) {

thres[i].start();

}

}

}

din\_philo.javaの実行結果

[ryonosuke\_araki@RyonosukenoMacBook-Air 14:25:42] $ javac din\_philo.java [ ~/OneDrive - The University of Tokyo/soubunkiso2 ]

[ryonosuke\_araki@RyonosukenoMacBook-Air 14:25:57] $ java Dining [ ~/OneDrive - The University of Tokyo/soubunkiso2 ]

1 is thinking.

2 is thinking.

0 is thinking.

4 is thinking.

3 is thinking.

0 feels hungry.

1 feels hungry.

2 feels hungry.

3 feels hungry.

4 feels hungry.

0 is starving.

1 is starving.

2 is starving.

3 is starving.

4 is starving.

^Z

zsh: suspended java Dining

DinningPhilosophers.java（共有資源に優先順位を与えるという手法によって、デッドロックを回避）

public class DiningPhilosophers {

// 哲学者 (箸) の数

static final int N = 5;

// 箸の配列

static ChopStick[] chopsticks;

// 哲学者の配列

static Philosopher[] philosophers;

public static void main(String[] args) {

// 箸オブジェクトを N 本用意する

chopsticks = new ChopStick[N];

// オブジェクトの初期化

for (int i = 0; i < N; i++) {

chopsticks[i] = new ChopStick();

}

// 哲学者オブジェクトの生成と、

// 各自が使える箸の登録

philosophers = new Philosopher[N];

for (int i = 0; i < N; i++) {

philosophers[i] = new Philosopher(chopsticks[i],

chopsticks[(i+1)%N]);

}

// 表示用

for (int i = 0; i < N; i++) {

System.out.printf("Philosopher %d ", i);

}

System.out.println();

for (int i = 0; i < N; i++) {

System.out.print("-------------------- ");

}

System.out.println();

// 哲学者達に食事を始めさせる。

for (int i = 0; i < N; i++) {

philosophers[i].start();

}

}

}

// 箸クラス

class ChopStick {

static int counter = 0;

// 使用中か否か

boolean isUsed;

// 優先度

int rank;

ChopStick() {

isUsed = false;

// それぞれの箸に異なる優先度を与える

rank = counter++;

}

}

// 哲学者クラス

class Philosopher extends Thread {

static int counter = 0;

// 待ち時間の最大値。適当に設定してください。

final long WAITTIME = 100;

// 識別番号。哲学者オブジェクトを複数作るので、

// 表示の際に見やすいよう、各々に異なるIDを与えます。

int number;

// 自分のテリトリーにある2本の箸。

// lower : 優先度の低い箸,

// higher: 優先度の高い箸

ChopStick lowerStick;

ChopStick higherStick;

// 自分が箸を持っているか否か

boolean hasLowerStick;

boolean hasHigherStick;

// message

String msg;

Philosopher(ChopStick c1, ChopStick c2) {

// IDの付与

number = counter++;

// 自分が使える箸を登録

if (c1.rank < c2.rank) {

lowerStick = c1;

higherStick = c2;

} else {

lowerStick = c2;

higherStick = c1;

}

// 最初は箸を持っていない状態

hasLowerStick = false;

hasHigherStick = false;

}

// 哲学者の行動を登録したメソッド群

public void run () {

for (int i = 0; i < 5; i++) {

// 優先度の高い箸を取る

picUpHigherStick();

// 考え事をする

if(hasHigherStick)

think();

// 優先度の低い箸を取る

picUpLowerStick();

// 食べる

eat();

// また考える

think();

// 優先度の高い箸を置く

putDownHigherStick();

// 優先度の低い箸を置く

putDownLowerStick();

// またまた考え事をする

think();

}

}

// 優先度の高い箸を取る

synchronized void picUpHigherStick() {

// 優先度の高い箸が空くまで待つ

while(higherStick.isUsed)

await();

higherStick.isUsed = true;

hasHigherStick = true;

printAnEvent("pick up stick No." + higherStick.rank);

}

// 優先度の低い箸を取る

synchronized void picUpLowerStick() {

if (hasHigherStick) {

// 優先度の低い箸が空くまで待つ

while(lowerStick.isUsed)

await();

lowerStick.isUsed = true;

hasLowerStick = true;

printAnEvent("pick up stick No." + lowerStick.rank);

}

}

// 食事をするメソッド

void eat() {

if (hasLowerStick && hasHigherStick) {

printAnEvent("\*\*\*eating\*\*\*");

// ランダム時間だけ待機

waitRandom();

}

}

// 考え事

void think() {

printAnEvent(" \*\*\*thinking\*\*\*");

waitRandom();

}

// 優先度の高い箸を置く

synchronized void putDownHigherStick() {

// この条件は、単に (hasHigherStick) だけでもよい。

if (hasLowerStick && hasHigherStick) {

higherStick.isUsed = false;

hasHigherStick = false;

printAnEvent("put down stick No." + higherStick.rank);

}

}

// 優先度の低い箸を置く

synchronized void putDownLowerStick() {

if (hasLowerStick) {

lowerStick.isUsed = false;

hasLowerStick = false;

printAnEvent("put down stick No." + lowerStick.rank);

}

}

// 表示用

synchronized void printAnEvent(String str) {

msg = "";

for(int i = 0; i < 22 \* number; i++) {

msg += " ";

}

msg += str;

System.out.println(msg);

}

// 適当な時間だけ待つ

void waitRandom() {

try {

sleep((long)(Math.random() \* WAITTIME));

} catch (InterruptedException e) { }

}

// 一瞬だけ待つ (これ、別にwaitRandom()メソッドでも代用可)

synchronized void await() {

try {

sleep(1);

} catch (InterruptedException e) { }

}

}

DinningPhilosophers.javaの実行結果

[ryonosuke\_araki@RyonosukenoMacBook-Air 18:46:25] $ java DiningPhilosophers [ ~/OneDrive - The University of Tokyo/soubunkiso2 ]

Philosopher 0 Philosopher 1 Philosopher 2 Philosopher 3 Philosopher 4

-------------------- -------------------- -------------------- -------------------- --------------------

pick up stick No.1

pick up stick No.3

pick up stick No.2

\*\*\*thinking\*\*\*

pick up stick No.4

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

pick up stick No.0

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

put down stick No.1

put down stick No.0

\*\*\*thinking\*\*\*

pick up stick No.1

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

put down stick No.2

put down stick No.1

\*\*\*thinking\*\*\*

pick up stick No.1

\*\*\*thinking\*\*\*

pick up stick No.2

\*\*\*eating\*\*\*

pick up stick No.0

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

put down stick No.3

put down stick No.2

\*\*\*thinking\*\*\*

pick up stick No.2

pick up stick No.3

\*\*\*thinking\*\*\*

\*\*\*eating\*\*\*

put down stick No.1

put down stick No.0

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

pick up stick No.1

\*\*\*eating\*\*\*

put down stick No.4

put down stick No.3

\*\*\*thinking\*\*\*

pick up stick No.3

pick up stick No.4

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

pick up stick No.0

\*\*\*eating\*\*\*

put down stick No.2

put down stick No.1

pick up stick No.1

pick up stick No.2

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

put down stick No.4

put down stick No.0

\*\*\*thinking\*\*\*

pick up stick No.0

\*\*\*eating\*\*\*

pick up stick No.4

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

put down stick No.1

put down stick No.0

\*\*\*thinking\*\*\*

put down stick No.3

put down stick No.2

\*\*\*thinking\*\*\*

pick up stick No.2

pick up stick No.3

\*\*\*thinking\*\*\*

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

pick up stick No.1

\*\*\*thinking\*\*\*

pick up stick No.0

\*\*\*eating\*\*\*

put down stick No.4

put down stick No.3

\*\*\*thinking\*\*\*

pick up stick No.3

pick up stick No.4

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

put down stick No.1

put down stick No.0

\*\*\*thinking\*\*\*

pick up stick No.1

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

pick up stick No.0

\*\*\*eating\*\*\*

put down stick No.2

put down stick No.1

\*\*\*thinking\*\*\*

pick up stick No.1

\*\*\*thinking\*\*\*

pick up stick No.2

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

put down stick No.4

pick up stick No.4

put down stick No.0

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

pick up stick No.0

\*\*\*eating\*\*\*

put down stick No.3

put down stick No.2

\*\*\*thinking\*\*\*

pick up stick No.2

\*\*\*thinking\*\*\*

pick up stick No.3

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

put down stick No.1

put down stick No.0

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

pick up stick No.1

\*\*\*eating\*\*\*

put down stick No.4

put down stick No.3

\*\*\*thinking\*\*\*

pick up stick No.3

pick up stick No.4

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

pick up stick No.0

\*\*\*eating\*\*\*

put down stick No.2

put down stick No.1

\*\*\*thinking\*\*\*

pick up stick No.2

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

put down stick No.3

put down stick No.2

\*\*\*thinking\*\*\*

pick up stick No.2

\*\*\*thinking\*\*\*

put down stick No.4

put down stick No.0

\*\*\*thinking\*\*\*

pick up stick No.4

\*\*\*thinking\*\*\*

pick up stick No.3

\*\*\*thinking\*\*\*

pick up stick No.1

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

put down stick No.2

put down stick No.1

\*\*\*thinking\*\*\*

pick up stick No.2

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

pick up stick No.3

put down stick No.3

\*\*\*eating\*\*\*

put down stick No.2

\*\*\*thinking\*\*\*

\*\*\*thinking\*\*\*

put down stick No.4

put down stick No.3

\*\*\*thinking\*\*\*

pick up stick No.4

\*\*\*thinking\*\*\*

pick up stick No.0

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

put down stick No.4

put down stick No.0

\*\*\*thinking\*\*\*

pick up stick No.4

\*\*\*thinking\*\*\*

pick up stick No.3

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

put down stick No.4

put down stick No.3

\*\*\*thinking\*\*\*

pick up stick No.4

\*\*\*thinking\*\*\*

pick up stick No.0

\*\*\*eating\*\*\*

\*\*\*thinking\*\*\*

put down stick No.4

put down stick No.0

\*\*\*thinking\*\*\*