universidade de aveiro deti departamento de electrónica, telecomunicações e informática

RELATÓRIO DO TRABALHO 2
JANTAR DE AMIGOS (RESTAURANT)

Trabalho Realizado por:
Roberto Rolão de Castro 107133
Tiago Caridade Gomes 108307

Índice

1. Introdução

O trabalho proposto, no contexto da disciplina de Sistemas Operativos, tem como objetivo a compreensão dos mecanismos associados à execução e sincronização de processos e ***threads***, na linguagem de programação ***C*** que simule um jantar.

Desta forma, é nos informado que um grupo de amigos combinou um jantar num restaurante que tem apenas um empregado de mesa (waiter) e um cozinheiro (chef). A interação destes intervenientes é regida pelas seguintes linhas, o primeiro cliente é responsável pelo pedido da comida, mas apenas depois de todos chegarem. O empregado de mesa deve informar o cozinheiro do pedido e, consecutivamente, trazer a comida para a mesa quando esta estiver pronta. No final, os amigos apenas devem abandonar a mesa depois de todos terminarem de comer, sendo que é função do último cliente a chegar ao restaurante de pagar a conta. O tamanho da mesa está ajustado ao número de amigos.

Como referido anteriormente, face ao código fornecido é pedido que se desenvolva uma solução em ***C*** que simule este jantar. Sabemos que os intervenientes são processos independentes, sendo a sua combinação efetuada através de semáforos e memória partilhada.

A apresentação desta informação é realizada através da impressão de uma tabela devidamente formatada que atualiza os estados dos intervenientes (***client, waiter*** e ***chef***) ao longo do jantar.

Ao longo deste relatório iremos discriminar as metodologias que utilizámos para a realização do projeto.

2. Estados dos Intervenientes

Tal como mencionado anteriormente ao longo do decorrer do jantar o estado dos intervenientes é atualizado, podendo estes tomar os seguintes estados:

clientStat[id]		
INIT	1	Estado Inicial
WAIT_FOR_FRIENDS	2	Espera dos restantes dos clientes
FOOD_REQUEST	3	Pede a comida ao ***waiter***
WAIT_FOR_FOOD	4	Espera pela Comida
EAT	5	Come a sua refeição
WAIT_FOR_OTHERS	6	Espera pelos outros acabarem de comer
WAIT_FOR_BILL	7	Espera para completar o pagamento
FINISHED	8	Cliente terminou a sua refeição
chefStat		
WAIT_FOR_ORDER	0	Espera pelo pedido de comida
COOK	1	Cozinha
REST	2	Descansa
waiterStat		
WAIT_FOR_REQUEST	0	Espera por ser chamado
INFORM_CHEF	1	Leva o pedido ao ***chef***
TAKE_TO_TABLE	2	Leva a comida à mesa
RECEIVE_PAYMENT	3	Recebe o pagamento

3. Abordagem utilizada

De modo a uma melhor compreensão do código fornecido, analisamos de forma detalhada todas as funções para compreender o que cada realiza.

Dessa forma, surgiu a elaboração da uma tabela de dados com o intuito de suporte para a verificação do comportamento de cada processo, para isso começamos por averiguar os semáforos disponibilizados:

Semáforo	Objetivo
mutex (valor inicial = 1)	Identifica uma determinada região crítica, que bloqueia qualquer processo que tente entrar numa região de modo a proteger a execução do processo atual. Dentro desta também é atualizado os estados dos diversos intervenientes.
friendsArrived (valor inicial = 0)	Usado pelos clientes de modo a esperar os restantes dos amigos.
requestReceived (valor inicial = 0)	Utilizado pelo cliente para esperar pelo empregado de mesa de modo a realizar o pedido.
foodArrived (valor inicial = 0)	Tem o intuito de informar os clientes quando é que o seu pedido chega à mesa, este quando está ativo advertem para o estado de espera pela comida.
allFinished (valor inicial = 0)	Aplicado pelos clientes de forma a espera que os outros acabem a sua refeição
waiterRequest (valor inicial = 0)	Manuseado pelo empregado de modo a obter a informação de quando é chamado tanto pelos clientes como pelo chef.
waitOrder (valor inicial = 0)	Utilizado pelo chef para esperar pelo pedido.

Dada tal informação e observação do código fonte chegamos à elaboração da seguinte tabela:

Semáforo	*down*			*up*		
	Entidade	**Função**	**Quantidade**	**Entidade**	**Função**	**Quantidade**
friendsArrived	cliente	waitFriends()	1	cliente	waitFriends()	1
requestReceived	cliente	orderFood() waitAndPay()	2	empregado	informChef() receivePayment()	2
foodArrived	cliente	waitFood()	1	empregado	takeFoodToTable()	1
allFinished	cliente	waitAndPay()	1	cliente	waitAndPay()	1
waiterRequest	empregado	waitForClientOrChef()	1	cliente, chefe	orderFood() waitAndPay() processOrder()	3
waitOrder	chefe	waitForOrder()	1	empregado	informChef()	1

4. Estrutura do Código

Nesta secção do relatório iremos explicar o código, que foi necessário acrescentar de maneira que a aplicação funcione sem problemas. Vamos dividir a explicação por cada ficheiro que compõem a pasta comprimida que nos foi fornecida.

4.1. *semSharedMemClient.c*

4.1.1. Função *waitFriends()*

Esta função simula a ação dos clientes que forem chegando ao restaurante esperam pelos outros clientes que ainda não chegaram.

No código completado, a variável ***tableClients*** foi incrementada, a medida que os clientes foram chegando, e representa o número de clientes presentes no momento na mesa. Se o número de clientes sentados na mesa for igual a 1, guarda-se na variável ***tableFirst*** o *id* do primeiro cliente que chegou e a variável booleana ***first*** passa a ser verdadeira. Depois, caso o número de clientes for igual ao tamanho da mesa, previamente definido na variável ***TABLESIZE***, é guardado na variável ***tableLast*** o *id* do último cliente que chegou, o seu estado muda para ***WAIT_FOR_FOOD*** e o novo estado é guardado. A seguir com a ajuda de um ciclo ***for***, que percorre o tamanho da mesa, vamos sair da zona crítica do semáforo do ***friendsArrived***. No caso de a mesa não estar completa, ou seja, se ainda não chegaram todos os clientes, o estado do cliente cujo *id* foi chamado como argumento atualiza para ***WAIT_FOR_FRIENDS*** e é guardado.

```
/* insert your code here */
sh->fSt.tableClients++;

if(sh->fSt.tableClients == 1){
    sh->fSt.tableFirst = id;
    first = true;
}

if (sh->fSt.tableClients == TABLESIZE){
    sh->fSt.tableLast = id;
    sh->fSt.st.clientStat[sh->fSt.tableLast] = WAIT_FOR_FOOD ;
    saveState(nFic, &sh->fSt);
    for(int i = 0; i < TABLESIZE-1; i++){
        if (semUp (semgid, sh->friendsArrived) == -1)                    /* exit critical region */
        { perror ("error on the up operation for semaphore access (CT)");
            exit (EXIT_FAILURE);
        }
    }

}
else  {
    sh->fSt.st.clientStat[id] = WAIT_FOR_FRIENDS ;
    saveState(nFic, &sh->fSt);

}
```

Figura 1 - Código completado na função *waitFriends()*

Mais abaixo no código, se a mesa não estiver cheia, ou seja, se os clientes não tiverem todos chegado, vamos entrar na zona crítica do semáforo do ***friendsArrived.*** E por fim vamos devolver a variável ***first***.

```
/* insert your code here */

if(sh->fSt.tableClients != TABLESIZE){
    if (semDown (semgid, sh->friendsArrived) == -1) {                                    /* enter critical region */
        perror ("error on the down operation for semaphore access (CT)");
        exit (EXIT_FAILURE);
    }
}

return first;
```

Figura 2 - Código completado na função *waitFriends()*

4.1.2. Função *orderFood()*

A função ***orderFood()*** encena a ação de um cliente pedir a comida para os outros clientes do grupo.

No código que foi necessário completar, começamos por atribuir à *flag* ***foodRequest*** o valor de 1. A seguir, saímos da zona crítica do semáforo do ***waiterRequest*** e atualizamos o status do cliente que pede a comida para o resto do grupo, neste caso o cliente é o primeiro que chegou cujo *id* está guardado na variável ***tableFirst***, para o estado ***FOOD_REQUEST*** e guardamos o novo estado.

```
sh->fSt.foodRequest=1;

if (semUp (semgid, sh->waiterRequest) == -1)                                         /* exit critical region */
{ perror ("error on the up operation for semaphore access (CT)");
    exit (EXIT_FAILURE);
}

sh->fSt.st.clientStat[sh->fSt.tableFirst] = FOOD_REQUEST ;
saveState(nFic, &sh->fSt);
```

Figura 3 - Código completado na função *orderFood()*

A seguir no código acrescentado, entramos na zona crítica do semáforo do ***requestReceived()***.

```
/* insert your code here */

if (semDown (semgid, sh->requestReceived) == -1) {                                   /* enter critical region */
    perror ("error on the down operation for semaphore access (CT)");
    exit (EXIT_FAILURE);
}
```

Figura 4 - Código completado na função *orderFood()*

4.1.3. Função *waitFood()*

Esta função simula a espera que os clientes têm de efetuar até a sua comida chegar.

Na parte onde foi necessário acrescentar código, atualizamos o estado do cliente, através do *id*, para ***WAIT_FOR_FOOD*** e guardamos o novo estado.

```
sh->fSt.st.clientStat[id] = WAIT_FOR_FOOD ;
saveState(nFic, &sh->fSt);
```

Figura 5 - Código acrescentado à função *waitFood()*

Mais abaixo, entramos na zona crítica do semáforo do ***foodArrived*** para esperar que a comida seja entregue.

```
/* insert your code here */
if(semDown(semgid, sh->foodArrived) == -1){
    perror ("error on the down operation for semaphore access (CT)");
    exit (EXIT_FAILURE);
}
```

Figura 6 - Código acrescentado à função *waitFood()*

A seguir, mudamos o estado dos clientes a medida que a comida é servida pelo empregado de mesa para o estado ***EAT*** e guardámo-lo.

```
/* insert your code here */
sh->fSt.st.clientStat[id] = EAT;
saveState(nFic, &sh->fSt);
```

Figura 7 - Código acrescentado à função *waitFood()*

4.1.4. Função waitAndPay()

A função ***waitAndPay()*** é chamada quando um cliente acaba de comer, espera que os outros clientes acabem de comer e que o último cliente que chegou a mesa pague a conta. É inicializada uma variável do tipo booleano, ***last***.

A variável ***tableFinishedEat***, que serve para controlar o número de clientes que acabaram de comer, é incrementada cada vez que a função é chamada. A seguir, se o *id* com que a função foi chamada é o *id* do último cliente que chegou à mesa a variável ***last*** toma o valor de verdadeiro. Posteriormente, o estado do cliente é modificado para ***WAIT_FOR_OTHERS*** e guarda-se o novo estado. Caso a variável ***tableFinishedEat*** tiver o valor guardado da variável ***TABLESIZE***, com a ajuda de um ciclo ***for*** que vai iterar sobre o número de clientes na mesa. E em cada iteração, sai-se da zona crítica do semáforo do ***allFinished***.

```
/* insert your code here */

sh->fSt.tableFinishEat++;
if (sh->fSt.tableLast == id){
    last=true;}
sh->fSt.st.clientStat[id] = WAIT_FOR_OTHERS;
saveState(nFic, &sh->fSt);

if (sh->fSt.tableFinishEat == TABLESIZE){
    for(int i = 0; i < TABLESIZE; i++){
        if (semUp (semgid, sh->allFinished) == -1)                                  /* exit critical region */
        { perror ("error on the up operation for semaphore access (CT)");
            exit (EXIT_FAILURE);
        }
    }

}
```

Figura 8 - Código acrescentado à função *waitAndPay()*

Mais abaixo no código, entra-se na zona crítica do semáforo do ***allFinished***.

```
/* insert your code here */

if (semDown (semgid, sh->allFinished) == -1)                                    /* exit critical region */
    { perror ("error on the up operation for semaphore access (CT)");
    exit (EXIT_FAILURE);
}
```

Figura 9 - Código acrescentado à função *waitAndPay()*

Caso o *id* com que a função foi chamada for o *id* do último cliente que chegou à mesa, a *flag* ***paymentRequest*** fica com o valor 1 e o estado do cliente que é responsável por pagar a conta muda para ***WAIT_FOR_BILL*** e guarda-se o novo estado. A seguir, sai-se da zona crítica do semáforo do ***waiterRequest***.

```
/* insert your code here */
sh->fSt.paymentRequest=1;
sh->fSt.st.clientStat[sh->fSt.tableLast] = WAIT_FOR_BILL ;
saveState(nFic, &sh->fSt);
if(semUp(semgid, sh->waiterRequest) == -1){
    perror ("error on the down operation for semaphore access (CT)");
    exit (EXIT_FAILURE);
}
```

Figura 10 - Código acrescentado à função *waitAndPay()*

Mais abaixo no código, mas ainda dentro da condição acima referida, entra-se na zona crítica do semáforo do ***requestReceived***.

```
/* insert your code here */

if(semDown(semgid, sh->requestReceived) == -1){
    perror ("error on the down operation for semaphore access (CT)");
    exit (EXIT_FAILURE);
}
```

Figura 11 - Código acrescentado à função *waitAndPay()*

Por fim, o estado do cliente é modificado para ***FINISHED*** e é guardado.

```
sh->fSt.st.clientStat[id] = FINISHED  ;
saveState(nFic, &sh->fSt);
```

Figura 12 - Código acrescentado à função *waitAndPay()*

4.2. *semSharedMemWaiter.c*

4.2.1. Função *waitForClientOrChef()*

A função ***waitForClientOrChef()*** é responsável pela espera do empregado de mesa por um pedido do chefe ou de um cliente. Logo no início é inicializada uma variável do tipo inteiro, ***ret***, que posteriormente vai servir para diferenciar de quem provém o pedido.

Onde foi necessário acrescentar código, o estado do empregado é atualizado para ***WAIT_FOR_REQUEST*** e o novo estado é guardado.

```
/* insert your code here */
sh->fSt.st.waiterStat = WAIT_FOR_REQUEST ;
saveState(nFic, &sh->fSt);
```

Figura 13 - Código completado na função *waitForClientOrChef()*

A seguir, entramos na zona crítica do semáforo do ***waiterRequest***.

```
/* insert your code here */
if (semDown(semgid, sh->waiterRequest) == -1)
{ perror ("error on the up operation for semaphore access (WT)");
    exit (EXIT_FAILURE);
}
```

Figura 14 - Código completado na função *waitForClientOrChef()*

Depois com o recurso a um conjunto de ***"if"*** e ***"else if"*** vamos diferenciar de que provem o pedido de comida e para isso usamos as *flags* ***foodRequest***, ***foodReady*** e ***paymentRequest***. Caso ***foodRequest*** tenha o valor de 1, toma o valor de 0 e a variável ***ret*** fica com o valor guardado em ***FOODREQ***. Se ***foodReady*** seja igual a 1, fica com o valor 0 e ***ret*** fica igual a ***FOODREADY***. E por fim se foi o cliente que pediu a conta, ou seja, a *flag* ***paymentRequest*** seja igual a 1, reinicializa-se com o valor 0 e ***ret*** com o valo que esta em ***BILL*** e caso isto tudo não se verifica ***ret*** fica com valor igual a 0.

```
/* insert your code here */

if(sh->fSt.foodRequest==1){
    sh->fSt.foodRequest=0;
    ret=FOODREQ;
}
else if(sh->fSt.foodReady==1){
    sh->fSt.foodReady=0;
    ret=FOODREADY;
}
```

```
else if(sh->fSt.paymentRequest==1){
    sh->fSt.paymentRequest=0;
    ret=BILL;
}
else{
    ret=0;
}
```

Figura 14 - Código completado na função *waitForClientOrChef()*

4.2.2. Função *informChef()*

Esta função é responsável pelo empregado de mesa dar a informação ao chefe de cozinha acerca de um pedido de comida de um determinado cliente.

Na parte onde acrescentamos o que faltava, a *flag* ***foodOrder*** fica com um valor igual 1. A seguir, o estado do empregado muda para ***INFORM_CHEF*** e o mesmo é guardado.

```
/* insert your code here */

sh->fSt.foodOrder=1;
sh->fSt.st.waiterStat=INFORM_CHEF;
saveState(nFic, &sh->fSt);
```

Figura 15 - Código completado na função *informChef()*

Depois, saímos das zonas dos semáforos do ***waitOrder*** e do ***requestReceived***.

```
/* insert your code here */
if (semUp(semgid, sh->waitOrder) == -1) {                                    /* exit critical region */
    perror ("error on the down operation for semaphore access (WT)");
    exit (EXIT_FAILURE);
}
if (semUp (semgid, sh->requestReceived) == -1) {                                /* exit critical region */
    perror ("error on the down operation for semaphore access (WT)");
    exit (EXIT_FAILURE);
}
```

Figura 16 - Código completado na função *informChef()*

4.2.3. Função *takeFoodToTable()*

A função ***takeFoodToTable()*** simula a ação do empregado de mesa levar o pedido de comida de cada cliente para a mesa.

Começamos por atualizar o estado do empregado de mesa para ***TAKE_TO_TABLE*** e por guardar o novo estado. Logo a seguir, com recurso a um ciclo ***for***, que itera sobre todos os clientes que estão sentados na mesa, sai-se da zona crítica do semáforo do ***foodArrived***.

```
/* insert your code here */

sh->fSt.st.waiterStat=TAKE_TO_TABLE;
saveState(nFic, &sh->fSt);

for(int i=0;i<sh->fSt.tableClients;i++){
    if(semUp(semgid, sh->foodArrived) == -1){
    perror ("error on the down operation for semaphore access (CT)");
    exit (EXIT_FAILURE);
    }
}
```

Figura 17 - Código completado na função *takeFoodToTable()*

4.2.4. Função *receivePayment()*

Nesta função o empregado de mesa é responsável por receber o pagamento do último cliente que chegou ao restaurante.

Na secção onde é preciso acrescentar código, o estado do empregado muda-se para ***RECEIVE_PAYMENT*** e guarda-se o novo estado. Depois, saímos da zona crítica do semáforo do ***requestReceived***.

```
/* insert your code here */

sh->fSt.st.waiterStat=RECEIVE_PAYMENT;
saveState(nFic, &sh->fSt);

if(semUp(semgid, sh->requestReceived) == -1){
        perror ("error on the down operation for semaphore access (WT)");
        exit (EXIT_FAILURE);
}
```

Figura 18 - Código completado na função *receivePayment()*

4.3. *semSharedMemChef.c*

4.3.1. Função *waitForOrder()*

A função ***waitOrder()*** simula a ação de um chefe estar a espera que chegue um pedido à cozinha.

No código que foi necessário acrescentar de modo que funcione corretamente, entramos na zona crítica do semáforo ***waitOrder()***.

```
/* insert your code here */

if (semDown (semgid, sh->waitOrder) == -1) {                    /* enter critical region */
    perror ("error on the up operation for semaphore access (PT)");
    exit (EXIT_FAILURE);
}
```

Figura 19 - Código completado na função *waitForOrder()*

A seguir, a *flag* ***foodOrder*** toma o valor de 0 e o estado do chefe é atualizado para ***COOK*** e é guardado.

```
sh->fSt.foodOrder=0;
sh->fSt.st.chefStat = COOK;
saveState(nFic, &sh->fSt);
```

Figura 20 - Código completado na função *waitForOrder()*

4.3.2. Função *processOrder()*

Esta função é usada pelo chefe, num restaurante, e é invocada sempre que o chefe necessita de processar um pedido de comida trazido pelo empregado de mesa.

Na secção do código que foi necessário adicionar, colocamos a *flag*, ***foodReady***, com o valor 1. A seguir, atualizamos o estado do chefe para ***REST*** e é guardado.

```
/* insert your code here */
sh->fSt.foodReady = 1;
sh->fSt.st.chefStat = REST;
saveState(nFic, &sh->fSt);
```

Figura 21 - Código completado na função *processOrder()*

Mais abaixo no código, saímos da zona crítica do semáforo do ***waiterRequest***.

```
/* insert your code here */
if (semUp (semgid, sh->waiterRequest) == -1) {                                    /* exit critical region */
    perror ("error on the down operation for semaphore access (CT)");
    exit (EXIT_FAILURE);
}
```

Figura 22 - Código completado na função *processOrder()*

5. Resultados

Nesta secção do relatório vamos inserir os resultados que obtivemos ao correr a linha de comando ***"./run 4"***.

5.1. Run nº1

Run n.º 1

Restaurant - Description of the internal state

CH	WT	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	ATT	FIE	1st	las
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	-1
0	0	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	-1
0	0	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	2	0	1	-1
0	0	1	2	1	1	2	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	3	0	1	-1
0	0	1	2	1	1	2	1	1	2	2	1	1	1	1	1	1	1	1	1	1	1	4	0	1	-1
0	0	1	2	1	1	2	1	1	2	2	1	1	1	1	1	2	1	1	1	1	1	5	0	1	-1
0	0	1	2	1	2	2	1	1	2	2	1	1	1	1	1	2	1	1	1	1	1	6	0	1	-1
0	0	1	2	1	2	2	1	2	2	2	1	1	1	1	1	2	1	1	1	1	1	7	0	1	-1
0	0	1	2	2	2	2	1	2	2	2	1	1	1	1	1	2	1	1	1	1	1	8	0	1	-1
0	0	1	2	2	2	2	2	2	2	2	1	1	1	1	1	2	1	1	1	1	1	9	0	1	-1
0	0	1	2	2	2	2	2	2	2	2	1	1	1	2	1	2	1	1	1	1	1	10	0	1	-1
0	0	1	2	2	2	2	2	2	2	2	1	1	1	2	1	2	1	1	1	1	2	11	0	1	-1
0	0	1	2	2	2	2	2	2	2	2	1	1	1	2	1	2	2	1	1	1	2	12	0	1	-1
0	0	1	2	2	2	2	2	2	2	2	1	1	2	2	1	2	2	1	1	1	2	13	0	1	-1
0	0	1	2	2	2	2	2	2	2	2	1	1	2	2	1	2	2	1	1	2	2	14	0	1	-1
0	0	2	2	2	2	2	2	2	2	2	1	1	2	2	1	2	2	1	1	2	2	15	0	1	-1
0	0	2	2	2	2	2	2	2	2	2	1	2	2	2	1	2	2	1	1	2	2	16	0	1	-1
0	0	2	2	2	2	2	2	2	2	2	1	2	2	2	1	2	2	1	2	2	2	17	0	1	-1
0	0	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	1	2	2	2	18	0	1	-1
0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	19	0	1	-1
0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4	2	2	2	20	0	1	16
0	0	2	2	2	2	2	2	2	2	2	2	2	2	4	2	2	2	4	2	2	2	20	0	1	16
0	0	2	2	4	2	2	2	2	2	2	2	2	2	4	2	2	2	4	2	2	2	20	0	1	16
0	0	2	2	4	2	2	2	2	2	2	2	2	2	4	2	4	2	4	2	2	2	20	0	1	16
0	0	2	2	4	4	2	2	2	2	2	2	2	2	4	2	4	2	4	2	2	2	20	0	1	16
0	0	2	3	4	4	2	2	2	2	2	2	2	2	4	2	4	2	4	2	2	2	20	0	1	16
0	0	4	3	4	4	2	2	2	2	2	2	2	2	4	2	4	2	4	2	2	2	20	0	1	16
0	0	4	3	4	4	2	2	2	2	2	2	2	2	4	2	4	2	4	2	2	4	20	0	1	16
0	0	4	3	4	4	2	4	2	2	2	2	2	2	4	2	4	2	4	2	2	4	20	0	1	16
0	0	4	3	4	4	2	4	4	2	2	2	2	2	4	2	4	2	4	2	2	4	20	0	1	16
0	0	4	3	4	4	4	4	4	2	2	2	2	2	4	2	4	2	4	2	2	4	20	0	1	16
0	0	4	3	4	4	4	4	4	2	4	2	2	2	4	2	4	2	4	2	2	4	20	0	1	16
0	0	4	3	4	4	4	4	4	2	4	2	2	2	4	2	4	4	4	2	2	4	20	0	1	16
0	0	4	3	4	4	4	4	4	4	4	2	2	2	4	2	4	4	4	2	2	4	20	0	1	16
0	0	4	3	4	4	4	4	4	4	4	2	2	2	4	2	4	4	4	2	4	4	20	0	1	16
0	0	4	3	4	4	4	4	4	4	4	2	2	4	4	2	4	4	4	2	4	4	20	0	1	16
0	0	4	3	4	4	4	4	4	4	4	2	2	4	4	2	4	4	4	2	4	4	20	0	1	16
0	0	4	3	4	4	4	4	4	4	4	2	2	4	4	4	4	4	4	2	4	4	20	0	1	16
0	0	4	3	4	4	4	4	4	4	4	2	4	4	4	4	4	4	4	2	4	4	20	0	1	16
0	0	4	3	4	4	4	4	4	4	4	2	4	4	4	4	4	4	4	4	4	4	20	0	1	16
0	0	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	1	16
0	1	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	1	16
0	0	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	1	16
1	0	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	1	16
1	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	1	16
2	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	1	16
2	2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	1	16
2	2	4	4	4	4	4	4	5	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	1	16
2	2	4	4	4	4	4	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	1	16
2	2	4	4	4	5	4	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	1	16
2	2	4	4	4	5	4	5	5	4	4	4	4	4	5	4	4	4	4	4	4	4	20	0	1	16
2	2	5	4	4	5	4	5	5	4	4	4	4	4	5	4	4	4	4	4	4	4	20	0	1	16
2	2	5	4	4	5	5	5	5	4	4	4	4	4	5	4	4	4	4	4	4	4	20	0	1	16
2	2	5	4	4	5	5	5	5	4	4	4	4	4	5	4	5	4	4	4	4	4	20	0	1	16
2	2	5	4	4	5	5	5	5	4	5	4	4	4	5	4	5	4	4	4	4	4	20	0	1	16
2	2	5	4	5	5	5	5	5	4	5	4	4	4	5	4	5	4	4	4	4	4	20	0	1	16
2	2	5	4	5	5	5	5	5	4	5	4	4	5	5	4	5	4	4	4	4	4	20	0	1	16
2	2	5	4	5	5	5	5	5	5	5	4	4	5	5	4	5	4	4	4	4	4	20	0	1	16
2	2	5	4	5	5	5	5	5	5	5	4	4	5	5	4	5	4	5	4	4	4	20	0	1	16
2	2	5	5	5	5	5	5	5	5	5	4	4	5	5	4	5	4	5	4	4	4	20	0	1	16
2	2	5	5	5	5	5	5	5	5	5	4	4	5	5	5	5	4	5	4	4	4	20	0	1	16
2	2	5	5	5	5	5	5	5	5	5	4	4	5	5	5	5	4	5	4	4	5	20	0	1	16
2	2	5	5	5	5	5	5	5	5	5	4	4	5	5	5	5	4	5	4	5	5	20	0	1	16
2	2	5	5	5	5	5	5	5	5	5	4	4	5	5	5	5	4	5	5	5	5	20	0	1	16
2	2	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	4	5	5	5	5	20	0	1	16
2	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	20	0	1	16
2	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	20	0	1	16
2	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	20	0	1	16
2	0	5	5	5	5	6	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	20	1	1	16
2	0	5	5	5	5	6	5	5	5	5	6	5	5	5	5	5	5	5	5	5	5	20	2	1	16
2	0	5	5	5	6	6	5	5	5	5	6	5	5	5	5	5	5	5	5	5	5	20	3	1	16
2	0	5	5	5	6	6	5	5	5	6	6	5	5	5	5	5	5	5	5	5	5	20	4	1	16
2	0	5	5	6	6	6	5	5	5	6	6	5	5	5	5	5	5	5	5	5	5	20	5	1	16
2	0	5	5	6	6	6	5	5	5	6	6	5	5	5	5	5	5	5	5	5	6	20	6	1	16
2	0	5	5	6	6	6	5	5	5	6	6	5	5	5	6	5	5	5	5	5	6	20	7	1	16
2	0	5	6	6	6	6	5	5	5	6	6	5	5	5	6	5	5	5	5	5	6	20	8	1	16
2	0	6	6	6	6	6	5	5	5	6	6	5	5	5	6	5	5	5	5	5	6	20	9	1	16
2	0	6	6	6	6	6	6	5	5	6	6	5	5	5	6	5	5	5	5	5	6	20	10	1	16
2	0	6	6	6	6	6	6	5	5	6	6	5	5	5	6	5	5	6	5	5	6	20	11	1	16

Figura 23 - Run nº1 – 1ºParte

```
2  0    6  6  6  6  6  6  5  5  6  6  6  5  5  6  5  5  6  5  5  6  20  12  1  16
2  0    6  6  6  6  6  6  5  5  6  6  6  5  5  6  5  6  6  5  5  6  20  13  1  16
2  0    6  6  6  6  6  6  5  5  6  6  6  5  5  6  6  6  6  5  5  6  20  14  1  16
2  0    6  6  6  6  6  6  5  6  6  6  6  5  5  6  6  6  6  5  5  6  20  15  1  16
2  0    6  6  6  6  6  6  5  6  6  6  6  5  5  6  6  6  6  5  6  6  20  16  1  16
2  0    6  6  6  6  6  6  5  6  6  6  6  5  6  6  6  6  6  5  6  6  20  17  1  16
2  0    6  6  6  6  6  6  6  6  6  6  6  5  6  6  6  6  6  5  6  6  20  18  1  16
2  0    6  6  6  6  6  6  6  6  6  6  6  5  6  6  6  6  6  6  6  6  20  19  1  16
2  0    6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20  20  1  16
2  0    6  6  8  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20  20  1  16
2  0    8  6  8  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20  20  1  16
2  0    8  6  8  6  6  6  6  6  6  6  6  6  6  8  6  6  6  6  6  6  20  20  1  16
2  0    8  6  8  6  8  6  6  6  6  6  6  6  6  8  6  6  6  6  6  6  20  20  1  16
2  0    8  6  8  6  8  8  6  6  6  6  6  6  6  8  6  6  6  6  6  6  20  20  1  16
2  0    8  6  8  6  8  8  6  6  6  6  6  6  6  8  6  6  7  6  6  6  20  20  1  16
2  0    8  6  8  6  8  8  6  6  8  6  6  6  6  8  6  6  7  6  6  6  20  20  1  16
2  0    8  6  8  8  8  8  6  6  8  6  6  6  6  8  6  6  7  6  6  6  20  20  1  16
2  0    8  6  8  8  8  8  6  6  8  8  6  6  6  8  6  6  7  6  6  6  20  20  1  16
2  0    8  8  8  8  8  8  6  6  8  8  6  6  6  8  6  6  7  6  6  6  20  20  1  16
2  0    8  8  8  8  8  8  6  6  8  8  8  6  6  8  6  6  7  6  6  6  20  20  1  16
2  0    8  8  8  8  8  8  6  6  8  8  8  6  6  8  6  6  7  6  8  6  20  20  1  16
2  0    8  8  8  8  8  8  6  6  8  8  8  6  6  8  6  8  7  6  8  6  20  20  1  16
2  0    8  8  8  8  8  8  6  6  8  8  8  6  6  8  6  8  7  6  8  6  20  20  1  16
2  0    8  8  8  8  8  8  6  6  8  8  8  6  8  8  8  8  7  6  8  6  20  20  1  16
2  0    8  8  8  8  8  8  6  6  8  8  8  8  8  8  8  8  7  6  8  6  20  20  1  16
2  0    8  8  8  8  8  8  8  6  8  8  8  8  8  8  8  8  7  6  8  8  20  20  1  16
2  0    8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  7  6  8  8  20  20  1  16
2  0    8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  7  8  8  8  20  20  1  16
2  3    8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  7  8  8  8  20  20  1  16
2  3    8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  20  20  1  16
```

Figura 24 - Run nº1 – 2ºParte

5.2. Run nº2

Run n.º 2

Restaurant - Description of the internal state

CH	WT	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	ATT	FIE	1st	las
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	-1
0	0	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	6	-1
0	0	1	1	2	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	6	-1
0	0	1	1	2	1	1	1	2	1	1	1	1	1	1	1	1	1	2	1	1	1	3	0	6	-1
0	0	1	1	2	1	1	2	2	1	1	1	1	1	1	1	1	1	2	1	1	1	4	0	6	-1
0	0	1	1	2	1	1	2	2	1	1	1	1	1	1	1	1	1	2	1	1	2	5	0	6	-1
0	0	2	1	2	1	1	2	2	1	1	1	1	1	1	1	1	1	2	1	1	2	6	0	6	-1
0	0	2	1	2	1	1	2	2	2	1	1	1	1	1	1	1	1	2	1	1	2	7	0	6	-1
0	0	2	1	2	1	1	2	2	2	1	1	1	1	1	1	2	1	2	1	1	2	8	0	6	-1
0	0	2	1	2	2	1	2	2	2	1	1	1	1	1	1	2	1	2	1	1	2	9	0	6	-1
0	0	2	1	2	2	1	2	2	2	1	1	2	1	1	1	2	1	2	1	1	2	10	0	6	-1
0	0	2	1	2	2	1	2	2	2	1	1	2	1	1	1	2	1	2	2	1	2	11	0	6	-1
0	0	2	1	2	2	1	2	2	2	1	1	2	1	1	2	2	1	2	2	1	2	12	0	6	-1
0	0	2	1	2	2	1	2	2	2	1	2	2	1	1	2	2	1	2	2	1	2	13	0	6	-1
0	0	2	1	2	2	1	2	2	2	1	2	2	1	2	2	2	1	2	2	1	2	14	0	6	-1
0	0	2	2	2	2	1	2	2	2	1	2	2	1	2	2	2	1	2	2	1	2	15	0	6	-1
0	0	2	2	2	2	1	2	2	2	2	2	2	1	2	2	2	1	2	2	1	2	16	0	6	-1
0	0	2	2	2	2	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1	2	17	0	6	-1
0	0	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	1	2	18	0	6	-1
0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	19	0	6	-1
0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4	2	20	0	6	18
0	0	2	2	2	2	2	4	2	2	2	2	2	2	2	2	2	2	2	2	4	2	20	0	6	18
0	0	2	2	2	2	2	4	2	4	2	2	2	2	2	2	2	2	2	2	4	2	20	0	6	18
0	0	2	4	2	2	2	4	2	4	2	2	2	2	2	2	2	2	2	2	4	2	20	0	6	18
0	0	2	4	2	2	2	4	3	4	2	2	2	2	2	2	2	2	2	4	4	2	20	0	6	18
0	0	2	4	2	2	2	4	3	4	2	2	2	2	2	2	2	2	2	4	4	2	20	0	6	18
0	0	2	4	2	2	2	4	3	4	2	4	2	2	2	2	2	2	2	4	4	2	20	0	6	18
0	0	2	4	2	2	2	4	3	4	2	4	4	2	2	2	2	2	2	4	4	2	20	0	6	18
0	0	2	4	4	2	2	4	3	4	2	4	4	2	2	2	2	2	2	4	4	2	20	0	6	18
0	0	2	4	4	2	2	4	3	4	2	4	4	2	4	2	2	2	2	4	4	2	20	0	6	18
0	0	2	4	4	2	2	4	3	4	2	4	4	2	4	2	2	2	2	4	4	4	20	0	6	18
0	0	4	4	4	2	2	4	3	4	2	4	4	2	4	2	2	2	2	4	4	4	20	0	6	18
0	0	4	4	4	4	2	4	3	4	2	4	4	2	4	2	2	2	2	4	4	4	20	0	6	18
0	0	4	4	4	4	2	4	3	4	2	4	4	2	4	2	2	2	4	4	4	4	20	0	6	18
0	0	4	4	4	4	2	4	3	4	4	4	4	2	4	2	2	2	4	4	4	4	20	0	6	18
0	0	4	4	4	4	2	4	3	4	4	4	4	2	4	2	2	4	4	4	4	4	20	0	6	18
0	0	4	4	4	4	2	4	3	4	4	4	4	2	4	4	2	4	4	4	4	4	20	0	6	18
0	0	4	4	4	4	4	4	3	4	4	4	4	2	4	4	2	4	4	4	4	4	20	0	6	18
0	0	4	4	4	4	4	4	3	4	4	4	4	4	4	4	2	4	4	4	4	4	20	0	6	18
0	0	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	18
0	1	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	18
0	0	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	18
0	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	18
1	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	18
2	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	18
2	2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	18
2	2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	4	20	0	6	18
2	2	4	4	4	4	4	4	4	4	4	4	4	4	5	4	4	4	4	5	4	4	20	0	6	18
2	2	5	4	4	4	4	4	4	4	4	4	4	4	5	4	4	4	4	5	4	4	20	0	6	18
2	2	5	4	4	4	4	4	4	4	4	4	5	4	5	4	4	4	4	5	4	4	20	0	6	18
2	2	5	4	5	4	4	4	4	4	4	4	5	4	5	4	4	4	4	5	4	4	20	0	6	18
2	2	5	5	5	4	4	4	4	4	4	4	5	4	5	4	4	4	4	5	4	4	20	0	6	18
2	2	5	5	5	4	4	5	4	4	4	4	5	4	5	4	4	4	4	5	4	4	20	0	6	18
2	2	5	5	5	4	4	5	4	4	4	4	5	4	5	4	4	4	4	5	4	5	20	0	6	18
2	2	5	5	5	4	4	5	4	4	4	5	5	4	5	4	4	4	4	5	4	5	20	0	6	18
2	2	5	5	5	4	4	5	4	5	4	5	5	4	5	4	4	4	4	5	4	5	20	0	6	18
2	2	5	5	5	5	4	5	4	5	4	5	5	4	5	4	4	4	4	5	4	5	20	0	6	18
2	2	5	5	5	5	4	5	4	5	5	5	5	4	5	4	4	4	4	5	4	5	20	0	6	18
2	2	5	5	5	5	4	5	4	5	5	5	5	4	5	5	4	4	4	5	4	5	20	0	6	18
2	2	5	5	5	5	4	5	4	5	5	5	5	4	5	5	4	4	4	5	4	5	20	0	6	18
2	2	5	5	5	5	5	5	4	5	5	5	5	5	5	5	4	4	4	5	4	5	20	0	6	18
2	2	5	5	5	5	5	5	4	5	5	5	5	5	5	5	4	5	4	5	4	5	20	0	6	18
2	0	5	5	5	5	5	5	4	5	5	5	5	5	5	5	4	5	4	5	4	5	20	0	6	18
2	0	5	5	5	5	5	5	4	5	5	5	5	5	5	5	4	5	4	5	5	5	20	0	6	18
2	0	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5	5	4	5	5	5	20	0	6	18
2	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	20	0	6	18
2	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	20	0	6	18
2	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	6	5	5	5	5	20	1	6	18
2	0	5	5	5	5	5	5	5	5	5	6	5	5	5	5	5	6	5	5	5	5	20	2	6	18
2	0	5	5	5	5	5	5	5	5	6	6	5	5	5	5	5	6	5	5	5	5	20	3	6	18
2	0	5	5	6	5	5	5	5	5	6	6	5	5	5	5	5	6	5	5	5	5	20	4	6	18
2	0	5	5	6	5	5	5	5	5	6	6	5	5	5	6	5	6	5	5	5	5	20	5	6	18
2	0	5	5	6	5	5	5	5	6	6	6	5	5	5	6	5	6	5	5	5	5	20	6	6	18
2	0	5	6	6	5	5	5	5	6	6	6	5	5	5	6	5	6	5	5	5	5	20	7	6	18
2	0	5	6	6	5	5	5	5	6	6	6	5	5	5	6	5	6	5	5	6	5	20	8	6	18
2	0	5	6	6	5	5	5	5	6	6	6	5	5	5	6	5	6	5	6	6	5	20	9	6	18
2	0	5	6	6	5	5	5	5	6	6	6	5	6	5	6	5	6	5	6	6	5	20	10	6	18
2	0	5	6	6	5	5	6	5	6	6	6	5	6	5	6	5	6	5	6	6	5	20	11	6	18

Figura 25 - Run nº2 – 1ºParte

```
2  0   5  6  6  6  5  6  5  6  6  6  5  6  5  6  5  6  5  6  6  5  20 12  6  18
2  0   5  6  6  6  5  6  5  6  6  6  5  6  5  6  6  6  5  6  6  5  20 13  6  18
2  0   5  6  6  6  5  6  5  6  6  6  5  6  5  6  6  6  5  6  6  6  20 14  6  18
2  0   5  6  6  6  5  6  5  6  6  6  5  6  6  6  6  6  5  6  6  6  20 15  6  18
2  0   5  6  6  6  5  6  6  6  6  6  5  6  6  6  6  6  5  6  6  6  20 16  6  18
2  0   6  6  6  6  5  6  6  6  6  6  5  6  6  6  6  6  5  6  6  6  20 17  6  18
2  0   6  6  6  6  5  6  6  6  6  6  5  6  6  6  6  6  6  6  6  6  20 18  6  18
2  0   6  6  6  6  5  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20 19  6  18
2  0   6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20 20  6  18
2  0   6  6  8  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20 20  6  18
2  0   6  6  8  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  7  20 20  6  18
2  0   6  6  8  6  6  8  6  6  6  6  6  6  6  6  6  6  6  6  6  7  20 20  6  18
2  0   6  6  8  6  6  8  6  6  6  6  6  6  8  6  6  6  6  6  6  7  20 20  6  18
2  0   6  6  8  6  8  8  6  6  6  6  6  6  8  6  6  6  6  6  6  7  20 20  6  18
2  0   6  6  8  6  8  8  6  6  6  6  6  6  8  6  6  6  8  6  6  7  20 20  6  18
2  0   6  6  8  6  8  8  6  6  6  6  6  6  8  6  8  6  8  6  6  7  20 20  6  18
2  0   6  6  8  6  8  8  6  6  6  6  6  6  8  6  8  6  8  8  6  7  20 20  6  18
2  0   6  6  8  8  8  8  6  6  6  6  6  6  8  6  8  6  8  8  6  7  20 20  6  18
2  0   6  6  8  8  8  8  8  6  6  6  6  6  8  6  8  6  8  8  6  7  20 20  6  18
2  0   6  8  8  8  8  8  8  6  6  6  6  6  8  6  8  6  8  8  6  7  20 20  6  18
2  0   6  8  8  8  8  8  8  8  6  6  6  6  8  6  8  6  8  8  6  7  20 20  6  18
2  0   6  8  8  8  8  8  8  8  6  6  6  6  8  6  8  6  8  8  8  7  20 20  6  18
2  0   8  8  8  8  8  8  8  8  6  6  6  6  8  6  8  6  8  8  8  7  20 20  6  18
2  0   8  8  8  8  8  8  8  8  8  6  6  6  8  6  8  6  8  8  8  7  20 20  6  18
2  0   8  8  8  8  8  8  8  8  8  6  8  8  6  8  6  8  8  8  7  20 20  6  18
2  0   8  8  8  8  8  8  8  8  8  6  8  8  8  8  6  8  8  8  7  20 20  6  18
2  0   8  8  8  8  8  8  8  8  8  8  8  8  8  8  6  8  8  8  7  20 20  6  18
2  0   8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  7  20 20  6  18
2  0   8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  7  8  20 20  6  18
2  3   8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  7  8  20 20  6  18
2  3   8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  20 20  6  18
```

Figura 26 - Run nº2 – 2ºParte

5.3. Run nº3

Run n.º 3

Restaurant - Description of the internal state

CH	WT	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	ATT	FIE	1st	las
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	-1
0	0	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	6	-1
0	0	1	1	2	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	6	-1
0	0	1	1	2	1	1	1	2	1	1	1	1	1	1	1	1	1	2	1	1	1	3	0	6	-1
0	0	1	1	2	1	1	1	2	1	1	1	1	1	2	1	1	1	2	1	1	1	4	0	6	-1
0	0	1	2	2	1	1	1	2	1	1	1	1	1	2	1	1	1	2	1	1	1	5	0	6	-1
0	0	1	2	2	1	1	1	2	1	2	1	1	1	2	1	1	1	2	1	1	1	6	0	6	-1
0	0	1	2	2	1	1	1	2	1	2	1	1	1	2	1	1	2	2	1	1	1	7	0	6	-1
0	0	2	2	2	1	1	1	2	1	2	1	1	1	2	1	1	2	2	1	1	1	8	0	6	-1
0	0	2	2	2	1	1	1	2	1	2	1	2	1	2	1	1	2	2	1	1	1	9	0	6	-1
0	0	2	2	2	1	1	1	2	1	2	1	2	1	2	1	1	2	2	2	1	1	10	0	6	-1
0	0	2	2	2	1	1	1	2	1	2	1	2	1	2	2	1	2	2	2	1	1	11	0	6	-1
0	0	2	2	2	1	1	1	2	1	2	2	2	1	2	2	1	2	2	2	1	1	12	0	6	-1
0	0	2	2	2	1	1	2	2	1	2	2	2	1	2	2	1	2	2	2	1	1	13	0	6	-1
0	0	2	2	2	1	1	2	2	1	2	2	2	1	2	2	1	2	2	2	1	2	14	0	6	-1
0	0	2	2	2	1	2	2	2	1	2	2	2	1	2	2	1	2	2	2	1	2	15	0	6	-1
0	0	2	2	2	1	2	2	2	1	2	2	2	2	2	2	1	2	2	2	1	2	16	0	6	-1
0	0	2	2	2	1	2	2	2	1	2	2	2	2	2	2	1	2	2	2	2	2	17	0	6	-1
0	0	2	2	2	1	2	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	18	0	6	-1
0	0	2	2	2	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	19	0	6	-1
0	0	2	2	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	20	0	6	3
0	0	2	2	2	4	2	2	2	2	2	4	2	2	2	2	2	2	2	2	2	2	20	0	6	3
0	0	2	2	2	4	2	2	2	2	4	2	2	2	2	2	2	2	4	2	2	2	20	0	6	3
0	0	2	2	2	4	2	2	2	2	4	2	4	2	2	2	2	2	4	2	2	2	20	0	6	3
0	0	2	2	2	4	2	4	2	2	4	2	4	2	2	2	2	2	4	2	2	2	20	0	6	3
0	0	2	2	4	4	2	4	2	2	4	2	4	2	2	2	2	2	4	2	2	2	20	0	6	3
0	0	2	2	4	4	2	4	2	2	4	2	4	4	2	2	2	2	4	2	2	2	20	0	6	3
0	0	2	2	4	4	2	4	3	2	4	2	4	4	2	2	2	2	4	2	2	2	20	0	6	3
0	0	2	2	4	4	2	4	3	2	4	4	4	4	2	2	2	2	4	2	2	2	20	0	6	3
0	0	2	2	4	4	2	4	3	2	4	4	4	4	2	4	2	2	4	2	2	2	20	0	6	3
0	0	2	2	4	4	2	4	3	2	4	4	4	4	4	4	2	2	4	2	2	2	20	0	6	3
0	0	2	2	4	4	4	4	3	2	4	4	4	4	4	4	2	2	4	2	2	2	20	0	6	3
0	0	2	2	4	4	4	4	3	2	4	4	4	4	4	4	2	2	4	2	4	2	20	0	6	3
0	0	2	2	4	4	4	4	3	2	4	4	4	4	4	4	2	2	4	2	4	4	20	0	6	3
0	0	2	4	4	4	4	4	3	2	4	4	4	4	4	4	2	2	4	2	4	4	20	0	6	3
0	0	4	4	4	4	4	4	3	2	4	4	4	4	4	4	2	2	4	2	4	4	20	0	6	3
0	0	4	4	4	4	4	4	3	2	4	4	4	4	4	4	2	4	4	2	4	4	20	0	6	3
0	0	4	4	4	4	4	4	3	2	4	4	4	4	4	4	4	4	4	2	4	4	20	0	6	3
0	0	4	4	4	4	4	4	3	2	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	3
0	0	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	3
0	1	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	3
0	0	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	3
1	0	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	3
1	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	3
2	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	3
2	2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	3
2	2	4	4	4	4	4	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	6	3
2	2	4	4	4	4	4	5	4	4	4	4	4	4	4	4	4	4	5	4	4	4	20	0	6	3
2	2	4	4	4	4	5	4	5	4	4	4	4	4	4	4	4	4	5	4	4	4	20	0	6	3
2	2	4	4	4	5	5	5	4	4	4	4	4	4	4	4	4	4	5	4	4	4	20	0	6	3
2	2	4	4	4	5	5	5	4	4	4	4	4	5	4	4	4	4	5	4	4	4	20	0	6	3
2	2	4	4	4	5	5	5	4	4	5	4	4	5	4	4	4	4	5	4	4	4	20	0	6	3
2	2	4	4	4	5	5	5	4	4	5	5	4	5	4	4	4	4	5	4	4	4	20	0	6	3
2	2	4	4	4	5	5	5	4	4	5	5	4	5	5	4	4	4	5	4	4	4	20	0	6	3
2	2	4	5	4	5	5	5	4	4	5	5	4	5	5	5	4	4	5	4	4	4	20	0	6	3
2	2	5	5	4	5	5	5	4	4	5	5	4	5	5	5	4	4	5	4	4	4	20	0	6	3
2	2	5	5	4	5	5	5	4	4	5	5	4	5	5	5	4	4	5	4	4	5	20	0	6	3
2	2	5	5	4	5	5	5	4	4	5	5	5	5	5	5	4	4	5	4	4	5	20	0	6	3
2	2	5	5	4	5	5	5	4	4	5	5	5	5	5	5	4	5	5	4	4	5	20	0	6	3
2	2	5	5	5	5	5	5	4	4	5	5	5	5	5	5	4	5	5	4	4	5	20	0	6	3
2	0	5	5	5	5	5	5	4	4	5	5	5	5	5	5	4	5	5	4	4	5	20	0	6	3
2	0	5	5	5	5	5	5	4	4	5	5	5	5	5	5	5	5	5	4	4	5	20	0	6	3
2	0	5	5	5	5	5	5	4	4	5	5	5	5	5	5	5	5	5	4	5	5	20	0	6	3
2	0	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	4	5	5	20	0	6	3
2	0	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	20	0	6	3
2	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	20	0	6	3
2	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	6	5	5	5	20	1	6	3
2	0	5	5	5	5	5	5	5	5	5	5	6	5	5	5	5	5	6	5	5	5	20	2	6	3
2	0	5	5	5	5	6	5	5	5	5	5	6	5	5	5	5	5	6	5	5	5	20	3	6	3
2	0	5	5	5	5	6	5	5	5	5	6	6	5	5	5	5	5	6	5	5	5	20	4	6	3
2	0	5	5	5	5	6	5	5	5	6	6	6	5	5	5	5	5	6	5	5	5	20	5	6	3
2	0	5	5	5	5	6	5	5	5	6	6	6	5	5	5	5	5	6	5	5	6	20	6	6	3
2	0	5	5	5	5	6	5	5	5	6	6	6	5	5	6	5	5	6	5	5	6	20	7	6	3
2	0	5	5	5	5	6	5	5	6	6	6	6	5	5	6	5	5	6	5	5	6	20	8	6	3
2	0	5	6	5	5	6	5	5	6	6	6	6	5	5	6	5	5	6	5	5	6	20	9	6	3
2	0	5	6	5	5	6	5	5	6	6	6	6	5	5	6	5	5	6	5	6	6	20	10	6	3
2	0	6	6	5	5	6	5	5	6	6	6	6	5	5	6	5	5	6	5	6	6	20	11	6	3

Figura 27 - Run nº3 – 1ºParte

```
2  0   6  6  5  5  6  5  5  6  6  6  6  5  5  6  5  6  6  5  6  6  20 12  6  3
2  0   6  6  5  6  6  5  5  6  6  6  6  5  5  6  5  6  6  5  6  6  20 13  6  3
2  0   6  6  5  6  6  5  5  6  6  6  6  5  5  6  6  6  6  5  6  6  20 14  6  3
2  0   6  6  6  6  6  5  5  6  6  6  6  5  5  6  6  6  6  5  6  6  20 15  6  3
2  0   6  6  6  6  6  5  5  6  6  6  6  5  6  6  6  6  6  5  6  6  20 16  6  3
2  0   6  6  6  6  6  5  6  6  6  6  6  5  6  6  6  6  6  5  6  6  20 17  6  3
2  0   6  6  6  6  6  5  6  6  6  6  6  5  6  6  6  6  6  6  6  6  20 18  6  3
2  0   6  6  6  6  6  5  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20 19  6  3
2  0   6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20 20  6  3
2  0   6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  8  20 20  6  3
2  0   8  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  8  20 20  6  3
2  0   8  6  6  6  6  6  6  6  6  6  6  6  6  6  8  6  6  6  6  8  20 20  6  3
2  0   8  6  6  7  6  6  6  6  6  6  6  6  6  6  8  6  6  6  6  8  20 20  6  3
2  0   8  6  6  7  6  6  6  6  8  6  6  6  6  6  8  6  6  6  6  8  20 20  6  3
2  0   8  6  6  7  6  6  6  6  8  8  6  6  6  6  8  6  6  6  6  8  20 20  6  3
2  0   8  6  6  7  6  6  6  8  8  8  6  6  6  6  8  6  6  6  6  8  20 20  6  3
2  0   8  6  6  7  8  6  6  8  8  8  6  6  6  6  8  6  6  6  6  8  20 20  6  3
2  0   8  6  6  7  8  8  6  8  8  8  6  6  6  6  8  6  6  6  6  8  20 20  6  3
2  0   8  8  6  7  8  8  6  8  8  8  6  6  6  6  8  6  6  6  6  8  20 20  6  3
2  0   8  8  6  7  8  8  6  8  8  8  8  6  6  6  8  6  6  6  6  8  20 20  6  3
2  0   8  8  6  7  8  8  6  8  8  8  8  6  6  6  8  6  8  6  6  8  20 20  6  3
2  0   8  8  6  7  8  8  6  8  8  8  8  6  8  6  8  6  8  6  6  8  20 20  6  3
2  0   8  8  6  7  8  8  6  8  8  8  8  8  8  6  8  6  8  6  6  8  20 20  6  3
2  0   8  8  6  7  8  8  8  8  8  8  8  8  8  6  8  6  8  6  6  8  20 20  6  3
2  0   8  8  6  7  8  8  8  8  8  8  8  8  8  6  8  6  8  6  8  8  20 20  6  3
2  0   8  8  8  7  8  8  8  8  8  8  8  8  8  6  8  6  8  6  8  8  20 20  6  3
2  0   8  8  8  7  8  8  8  8  8  8  8  8  8  8  8  6  8  6  8  8  20 20  6  3
2  0   8  8  8  7  8  8  8  8  8  8  8  8  8  8  8  8  8  6  8  8  20 20  6  3
2  0   8  8  8  7  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  20 20  6  3
2  3   8  8  8  7  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  20 20  6  3
2  3   8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  20 20  6  3
```

Figura 28 - Run nº3 – 2ºParte

5.4. Run nº4

Run n.º 4

Restaurant - Description of the internal state

CH	WT	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	ATT	FIE	1st	las
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	-1
0	0	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	0	8	-1
0	0	1	1	1	1	2	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	2	0	8	-1
0	0	2	1	1	1	2	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	3	0	8	-1
0	0	2	1	1	1	2	1	1	1	2	1	1	1	1	1	2	1	1	1	1	1	4	0	8	-1
0	0	2	1	1	1	2	1	1	1	2	1	2	1	1	1	2	1	1	1	1	1	5	0	8	-1
0	0	2	1	2	1	2	1	1	1	2	1	2	1	1	1	2	1	1	1	1	1	6	0	8	-1
0	0	2	1	2	1	2	1	1	1	2	2	2	1	1	1	2	1	1	1	1	1	7	0	8	-1
0	0	2	1	2	1	2	2	1	1	2	2	2	1	1	1	2	1	1	1	1	1	8	0	8	-1
0	0	2	1	2	1	2	2	1	1	2	2	2	1	2	1	2	1	1	1	1	1	9	0	8	-1
0	0	2	1	2	1	2	2	1	1	2	2	2	1	2	1	2	1	1	1	1	2	10	0	8	-1
0	0	2	2	2	1	2	2	1	1	2	2	2	1	2	1	2	1	1	1	1	2	11	0	8	-1
0	0	2	2	2	1	2	2	1	1	2	2	2	1	2	1	2	2	1	1	1	2	12	0	8	-1
0	0	2	2	2	1	2	2	1	1	2	2	2	2	2	1	2	2	1	1	1	2	13	0	8	-1
0	0	2	2	2	1	2	2	1	1	2	2	2	2	2	1	2	2	1	1	2	2	14	0	8	-1
0	0	2	2	2	1	2	2	1	2	2	2	2	2	2	1	2	2	1	1	2	2	15	0	8	-1
0	0	2	2	2	2	2	2	1	2	2	2	2	2	2	1	2	2	1	1	2	2	16	0	8	-1
0	0	2	2	2	2	2	2	1	2	2	2	2	2	2	1	2	2	1	2	2	2	17	0	8	-1
0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	1	2	2	2	18	0	8	-1
0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	19	0	8	-1
0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4	2	2	2	20	0	8	16
0	0	2	2	2	2	2	4	2	2	2	2	2	2	2	2	2	2	4	2	2	2	20	0	8	16
0	0	2	2	2	2	2	4	2	2	2	2	2	2	2	2	4	2	4	2	2	2	20	0	8	16
0	0	2	2	2	2	2	4	2	2	2	2	2	2	2	2	4	2	4	2	2	2	20	0	8	16
0	0	2	2	2	2	2	4	2	4	2	2	2	2	2	2	4	2	4	2	2	2	20	0	8	16
0	0	2	2	2	2	2	4	2	4	2	2	2	2	2	4	4	2	4	2	2	2	20	0	8	16
0	0	4	2	2	2	2	4	2	4	2	2	2	2	2	4	4	2	4	2	2	2	20	0	8	16
0	0	4	2	2	4	2	4	2	4	2	2	2	2	2	4	4	2	4	2	2	2	20	0	8	16
0	0	4	2	2	4	2	4	2	4	2	2	2	4	2	4	4	2	4	2	2	2	20	0	8	16
0	0	4	4	2	4	2	4	2	4	2	2	2	4	2	4	4	2	4	2	2	2	20	0	8	16
0	0	4	4	2	4	2	4	4	4	2	2	2	4	2	4	4	2	4	2	2	2	20	0	8	16
0	0	4	4	4	4	2	4	4	4	2	2	2	4	2	4	4	2	4	2	2	2	20	0	8	16
0	0	4	4	4	4	2	4	4	4	2	2	2	4	2	4	4	2	4	4	2	2	20	0	8	16
0	0	4	4	4	4	2	4	4	4	2	2	2	4	4	4	4	2	4	4	2	2	20	0	8	16
0	0	4	4	4	4	2	4	4	4	2	2	4	4	4	4	4	2	4	4	2	2	20	0	8	16
0	0	4	4	4	4	2	4	4	4	3	2	4	4	4	4	4	2	4	4	2	2	20	0	8	16
0	0	4	4	4	4	2	4	4	4	3	4	4	4	4	4	4	2	4	4	2	2	20	0	8	16
0	0	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	2	4	4	2	2	20	0	8	16
0	0	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	2	4	4	4	2	20	0	8	16
0	0	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	2	4	4	4	4	20	0	8	16
0	0	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	20	0	8	16
0	1	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	20	0	8	16
0	0	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	20	0	8	16
1	0	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	20	0	8	16
1	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	8	16
2	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	8	16
2	2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	20	0	8	16
2	2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	4	4	4	20	0	8	16
2	2	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	4	4	4	20	0	8	16
2	2	5	4	4	4	4	4	4	5	4	4	4	4	4	4	4	4	5	4	4	4	20	0	8	16
2	2	5	4	4	4	4	4	4	5	4	4	4	4	4	5	4	4	5	4	4	4	20	0	8	16
2	2	5	4	4	5	4	4	4	5	4	4	4	4	4	5	4	4	5	4	4	4	20	0	8	16
2	2	5	5	4	5	4	4	4	5	4	4	4	4	4	5	4	4	5	4	4	4	20	0	8	16
2	2	5	5	4	5	4	4	4	5	4	4	4	5	4	5	4	4	5	4	4	4	20	0	8	16
2	2	5	5	4	5	4	4	4	5	4	4	4	5	4	5	5	4	5	4	4	4	20	0	8	16
2	2	5	5	4	5	4	5	4	5	4	4	4	5	4	5	5	4	5	4	4	4	20	0	8	16
2	2	5	5	4	5	4	5	4	5	4	4	4	5	4	5	5	4	5	5	4	4	20	0	8	16
2	2	5	5	5	5	4	5	4	5	4	4	4	5	4	5	5	4	5	5	4	4	20	0	8	16
2	2	5	5	5	5	4	5	4	5	4	5	4	5	4	5	5	4	5	5	4	4	20	0	8	16
2	2	5	5	5	5	4	5	4	5	4	5	4	5	5	5	5	4	5	5	4	4	20	0	8	16
2	2	5	5	5	5	4	5	5	5	4	5	4	5	5	5	5	4	5	5	4	4	20	0	8	16
2	2	5	5	5	5	4	5	5	5	4	5	5	5	5	5	5	4	5	5	4	4	20	0	8	16
2	2	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	4	5	5	4	4	20	0	8	16
2	0	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	4	5	5	4	4	20	0	8	16
2	0	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	4	5	5	5	4	20	0	8	16
2	0	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	4	5	5	5	5	20	0	8	16
2	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	20	0	8	16
2	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	20	0	8	16
2	0	5	5	5	5	6	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	20	1	8	16
2	0	5	5	5	5	6	5	5	5	5	6	5	5	5	5	5	5	5	5	5	5	20	2	8	16
2	0	5	5	5	5	6	5	5	5	5	6	5	5	5	5	6	5	5	5	5	5	20	3	8	16
2	0	5	5	6	5	6	5	5	5	5	6	5	5	5	5	6	5	5	5	5	5	20	4	8	16
2	0	5	5	6	5	6	5	5	5	5	6	5	5	5	5	6	5	5	5	5	6	20	5	8	16
2	0	5	5	6	5	6	5	5	5	5	6	5	5	5	6	6	5	5	5	5	6	20	6	8	16
2	0	5	5	6	5	6	5	5	5	5	6	5	5	5	6	6	5	5	5	6	6	20	7	8	16
2	0	5	5	6	5	6	5	5	5	5	6	5	5	6	6	6	5	5	5	6	6	20	8	8	16
2	0	5	5	6	5	6	5	6	5	5	6	5	5	6	6	6	5	5	5	6	6	20	9	8	16
2	0	5	5	6	5	6	5	6	5	5	6	5	5	6	6	6	5	5	6	6	6	20	10	8	16
2	0	5	5	6	5	6	5	6	5	5	6	5	5	6	6	6	5	6	6	6	6	20	11	8	16

Figura 29 - Run nº4 – 1ºParte

```
2  0   5  5  6  5  6  5  6  5  5  6  6  5  6  6  6  5  6  6  6  6  20 12  8  16
2  0   5  5  6  5  6  5  6  5  5  6  6  5  6  6  6  6  6  6  6  6  20 13  8  16
2  0   5  5  6  6  6  5  6  5  5  6  6  5  6  6  6  6  6  6  6  6  20 14  8  16
2  0   5  5  6  6  6  5  6  5  6  6  6  5  6  6  6  6  6  6  6  6  20 15  8  16
2  0   5  5  6  6  6  5  6  6  6  6  6  5  6  6  6  6  6  6  6  6  20 16  8  16
2  0   5  6  6  6  6  5  6  6  6  6  6  5  6  6  6  6  6  6  6  6  20 17  8  16
2  0   6  6  6  6  6  5  6  6  6  6  6  5  6  6  6  6  6  6  6  6  20 18  8  16
2  0   6  6  6  6  6  5  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20 19  8  16
2  0   6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20 20  8  16
2  0   6  6  6  6  8  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  20 20  8  16
2  0   6  6  6  6  8  6  8  6  6  6  6  6  6  6  6  6  6  6  6  6  20 20  8  16
2  0   6  6  6  6  8  6  8  6  6  6  6  8  6  6  6  6  6  6  6  6  20 20  8  16
2  0   6  6  6  6  8  6  8  6  8  6  6  6  8  6  6  6  6  6  6  6  20 20  8  16
2  0   6  6  6  6  8  6  8  6  8  8  6  6  8  6  6  6  6  6  6  6  20 20  8  16
2  0   6  6  6  6  8  8  8  6  8  8  6  6  8  6  6  6  6  6  6  6  20 20  8  16
2  0   6  6  6  6  8  8  8  6  8  8  6  6  8  6  6  6  6  8  6  6  20 20  8  16
2  0   6  8  6  6  8  8  8  6  8  8  6  6  8  6  6  6  6  8  6  6  20 20  8  16
2  0   6  8  6  6  8  8  8  6  8  8  6  6  8  6  6  6  6  8  8  6  20 20  8  16
2  0   6  8  6  8  8  8  8  6  8  8  6  6  8  6  6  6  6  8  8  6  20 20  8  16
2  0   6  8  6  8  8  8  8  6  8  8  6  6  8  6  6  6  6  8  8  8  20 20  8  16
2  0   6  8  6  8  8  8  8  6  8  8  6  6  8  8  6  6  6  8  8  8  20 20  8  16
2  0   6  8  8  8  8  8  8  6  8  8  6  6  8  8  6  6  6  8  8  8  20 20  8  16
2  0   6  8  8  8  8  8  8  6  8  8  6  6  8  8  6  8  6  8  8  8  20 20  8  16
2  0   6  8  8  8  8  8  8  8  8  8  6  6  8  8  6  8  6  8  8  8  20 20  8  16
2  0   6  8  8  8  8  8  8  8  8  8  6  8  8  8  6  8  6  8  8  8  20 20  8  16
2  0   8  8  8  8  8  8  8  8  8  8  6  8  8  8  6  8  6  8  8  8  20 20  8  16
2  0   8  8  8  8  8  8  8  8  8  8  8  8  8  8  6  8  6  8  8  8  20 20  8  16
2  0   8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  7  8  8  8  20 20  8  16
2  0   8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  7  8  8  8  20 20  8  16
2  3   8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  7  8  8  8  20 20  8  16
2  3   8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  20 20  8  16
```

Figura 30 - Run nº4 – 2ºParte

Como se pode observar, através destas quatro tabelas, quando o programa começa os clientes tem todos o número ***1*** que corresponde ao estado ***INIT***, onde os clientes estão se a dirigir ao restaurante. A seguir, começam a aparecer alguns números ***2*** (estado correspondente ao ***WAIT_FOR_FRIENDS***) na tabela, que significa que esse cliente chegou a mesa e esta a espera que que os outros clientes também cheguem. Uma vez que o último cliente chegue, o primeiro cliente que chegou começa a pedir a comida (o estado dele muda para ***FOOD_REQUEST*** e o seu número para ***3***) e o número dos outros clientes muda para ***4*** (estado correspondente ao ***WAIT_FOR_FOOD***).

Quando o cliente que pediu a comida acabe o pedido o estado dele muda para ***WAIT_FOR_FOOD*** (número ***4***) e o empregado de mesa também muda o seu estado de ***WAIT_FOR_REQUEST*** (número ***0***) para ***INFORM_CHEF*** (número ***1***). Uma que o chefe recebeu o pedido de comida muda o seu estado de ***WAIT_FOR_ORDER*** (número ***0***) para ***COOK*** (número ***1***) e o empregado de mesa volta ao seu estado inicial. A seguir, quando a comida estiver pronta, o chefe muda para o estado ***REST*** (número ***2***), o empregado de mesa para ***TAKE_TO_TABLE*** (número ***2***) e os clientes que vão ser servidos vão mudando o seu estado para ***EAT*** (número ***5***) e a medida que forem acabando de comer mudam o seu estado para ***WAIT_FOR_ORTHERS***.

A seguir, quando todos acabem de comer o último cliente que chegou à mesa pede a conta logo o seu estado muda para ***WAIT_FOR_BILL*** (número ***7***) e o estado do empregado de mesa muda para ***RECEIVE_PAYMENT*** (número ***3***). E por fim o estado de todos os clientes muda para ***FINISHED*** (número ***8***).

6. Conclusão

A este ponto do trabalho/relatório podemos afirmar que o desenvolvimento e implementação do script apresentado na introdução deste trabalho foi realizado com sucesso, uma vez que conseguimos alcançar todos os objetivos propostos e consecutivamente os resultados esperados.

No entanto, ao longo da realização do projeto, deparamo-nos com alguns entraves, cujos foram superados com uma pesquisa intensa sobre a linguagem em questão e sobre mecanismos associados à execução e sincronização de processos e ***threads***, apoiada sobretudo nos materiais disponibilizados nas aulas da disciplina, e com uma persistência dos membros envolvidos.

Concluímos ainda que este trabalho proporcionou um melhor e maior conhecimento sobre a linguagem em questão e ajudou para uma melhor perceção sobre o funcionamento dos processos que em envolvem semáforos.

7. Bibliografia

Materiais disponibilizados pelo Docente na página do e-learning da unidade curricular em questão.

Site consultado no decorrer do trabalho:

- https://stackoverflow.com/