



Ingeniería Informática + ADE

Universidad de Granada (UGR)

Autor: Ismael Sallami Moreno

Asignatura: Sistemas Concurrentes y Distribuidos



Índice

1. Enunciado	3
2. Código	3
3. Explicación detallada del Código	8
3.1. Introducción	8
3.2. Estructuras Utilizadas	8
3.3. Uso de Mutex	8
3.4. Variable de Fin del Juego	8
3.5. Hebra Jugador	8
3.6. Hebra NPC	9
3.7. Asignación de Puntos	9
3.8. Conclusión	9
4. Ejemplo de salida	9
5. Comando de compilación	12
6. Material	12



1 Enunciado

En este examen, se debe implementar una solución para un minijuego de Mario Party. El juego debe permitir entre 2 y 8 jugadores. A continuación, se describen las reglas y la lógica del juego:

- **Número de jugadores:** El juego debe permitir entre 2 y 8 jugadores. Cada jugador tendrá un turno para participar en el juego.
- **Objetivo del juego:** Los jugadores deben pulsar una dirección (arriba, abajo, izquierda, derecha). Si la dirección pulsada coincide con la dirección aleatoria en la que se ha colocado el corazón, el jugador que primero llegue al corazón obtiene 5 puntos, el segundo 3 puntos y los demás 2 puntos. En este caso la dirección del jugador se debe de generar de manera aleatoria.
- **Dirección aleatoria:** Al inicio de cada ronda, se generará una dirección aleatoria en la que se colocará el corazón. Esta dirección puede ser una de las siguientes: arriba, abajo, izquierda o derecha.
- **Turnos de los jugadores:** Cada jugador tendrá un turno para pulsar una dirección. El orden de los turnos puede ser determinado aleatoriamente o en un orden predefinido.
- **Puntuación:**
 - El primer jugador que pulse la dirección correcta obtiene 5 puntos.
 - El segundo jugador que pulse la dirección correcta obtiene 3 puntos.
 - Los demás jugadores que pulsen la dirección correcta obtienen 2 puntos.
- Se deben de entregar tres ficheros:
 - sol_tusapellidoynombre_P1yP2.cpp: código de la solución.
 - sol_tusapellidoynombre_P1yP2.txt: instrucción de compilación.
 - sol_tusapellidoynombre_P1yP2.pdf: fichero en que se incluye capturas del código y la explicación detallada del mismo.

2 Código

```
1 #include <iostream>
2 #include <cassert>
3 #include <thread>
4 #include <mutex>
5 #include <random>
6 #include <chrono>
7 #include "scd.h"
8
9 using namespace std;
10 using namespace scd;
```

```

11
12 const int num_rondas = 10;    // Número de rondas
13 const int num_jugadores = 4; // Número de jugadores
14
15 int puntos[num_jugadores] = {0}; // Puntos de cada jugador
16 int puntosParaGanar[3] = {5, 3, 2}; // El primer jugador gana 5, el segundo
    gana 3 y el resto ganan 2 (si miran a la dirección en la que está el
    corazón)
17 int puntos_ronda = 0; // Puntos que se ganan en la ronda actual
18 bool fin = false;    // Booleano que ayuda a determinar si se ha terminado
    el juego
19
20 mutex mtx_cout;      // Mutex para proteger la salida por pantalla
21 mutex mtx_operacion; // Mutex para cambiar el valor de variables
22
23 int direccion_corazon; // Dirección del corazón
24 int puestoJugador = 0; // Cuando el jugador mira al corazón en comparación
    con los demás jugadores. La usaremos para acceder al vector de puntos
    que puede ganar.
25 int jugadores_actuales = 0; // variable que refleja los jugadores que ya
    han actuado en la ronda actual
26 Semaphore corazon_disponible(0); // Semáforo para saber si el corazón
    está disponible. 0 equivale a no disponible, 1 equivale a disponible
27 Semaphore puedo_generar(1); // Semáforo para saber si se puede
    generar un corazón. 1=Si, 0=No
28 Semaphore fin_ronda(0); // Semáforo inicializado en 0, tiene la función de
    indicar que la ronda ha finalizado
29
30 /**
31  * @brief Función que muestra los puntos de cada jugador
32  */
33 void mostrar_puntos() {
34     cout << "Puntos: ";
35     for (int i = 0; i < num_jugadores; ++i)
36         cout << puntos[i] << " ";
37     cout << endl << endl << endl << flush;
38 }
39 /**
40  * @brief Función que dependiendo de el numero que le pase me dice si es
    mario, luigi, peach o bowser (mario = 0, luigi = 1, peach = 2, bowser
    = 3)
41  * @param i identificador del jugador
42  */
43 string nombre_jugador(int i) {
44     switch (i) {
45         case 0:
46             return "Mario";
47         case 1:
48             return "Luigi";
49         case 2:
50             return "Peach";
51         case 3:
52             return "Bowser";
53         default:
54             return "Jugador desconocido";
55     }

```

```

56 }
57
58 /**
59  * @brief Función que ejecuta la hebra del NPC, trata de poner un corazón
    en una determinada dirección, además lleva la cuenta de las rondas. Si
    se ha terminado el juego, se sale del bucle.
60  */
61 void funcion_hebra_NPC() {
62     for (int i = 0; i < num_rondas; ++i) {
63         jugadores_actuales = 0; // Reseteamos el contador al inicio de cada
            ronda
64         puestoJugador = 0;      // Reseteamos el puesto al inicio de cada
            ronda
65
66         sem_wait(puedo_generar); // Esperamos a que se pueda generar un nuevo
            corazón
67
68         direccion_corazon = aleatorio<0, 3>(); // Generamos dirección
            aleatoria en la que poner el corazón
69
70         cout << "NPC: Corazón visible en la dirección " << direccion_corazon
            << " ." << endl << flush;
71
72         for (int i = 0; i < num_jugadores; ++i) {
73             //compruebo si el semáforo corazon_disponible tiene procesos
                esperando
74             int sval;
75             sem_signal(corazon_disponible); // Despertamos a todos los
                jugadores
76             //sem_signal(corazon_disponible); // Despertamos a todos los
                jugadores
77         }
78
79         // Esperamos a que todos los jugadores terminen la ronda
80         sem_wait(fin_ronda);
81
82         // Fin de ronda
83         cout << "++++++-----+++++-----++++ FIN DE LA RONDA " << i+1 << "
            ++++++-----+++++-----++++" << endl << flush;
84         mostrar_puntos();
85     }
86
87     // Señalizamos el fin del juego
88     fin = true;
89     for (int i = 0; i < num_jugadores; ++i) {
90         sem_signal(corazon_disponible);
91     }
92     //Imprimimos por pantalla el fin del juego
93     cout << endl << "++++++-----+++++-----++++ FIN DEL JUEGO DE MARIO PARTY
        CON 10 RONDAS" << " ++++++-----+++++-----++++" << endl << flush;
94     mostrar_puntos();
95 }
96 }
97
98 /**

```



```

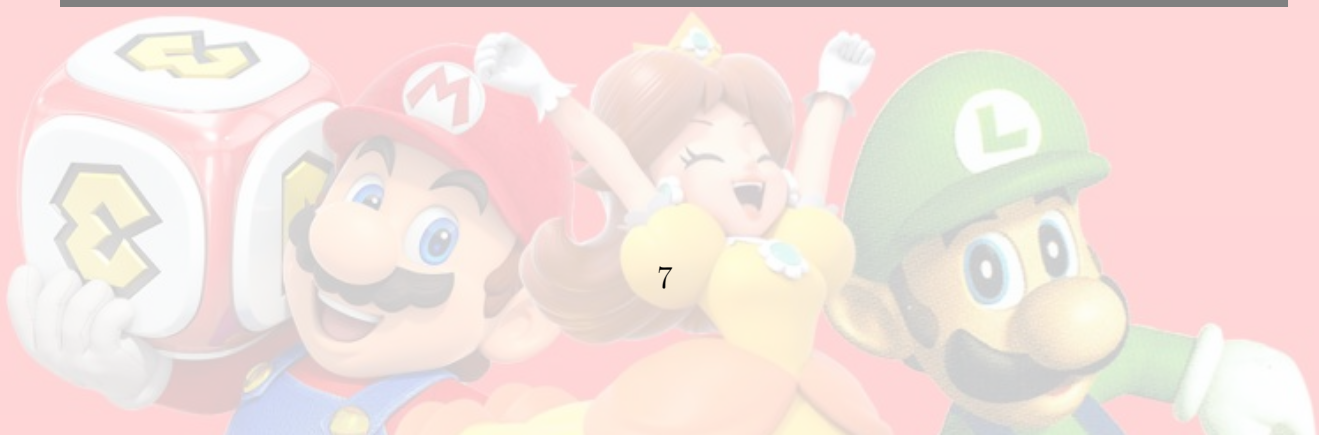
99  * @brief Función que ejecuta la hebra de un jugador, trata de mirar hacia
    una dirección aleatoria y si coincide con la del corazón, se le asigna
    una cantidad de puntos en función de el orden de llegada. Si es el ú
    ltimo jugador en actuar, se libera el semáforo de fin de ronda.
100  *
101  * @param num_jugador Número del jugador
102  */
103 void funcion_hebra_jugador(int num_jugador) {
104     while (!fin) {
105         sem_wait(corazon_disponible); // Como hemos despertado anteriormente
            a todos los jugadores, cuando este adquiere el semáforo, los demás
            pueden entrar de igual manera
106         if (!fin) { // Necesario comprobarlo. Por si se ha puesto fin=true
            mientras estaba bloqueado
107             string nombre = nombre_jugador(num_jugador);
108             mtx_operacion.lock();
109             jugadores_actuales++; // Incrementamos el número de jugadores que
                han actuado, al ser una variable compartida, la protegemos con
                un mutex
110             mtx_operacion.unlock();
111
112             int direccion_jugador = aleatorio<0, 3>(); // Generamos
                aleatoriamente la dirección hacia donde va a mirar el jugador
113             bool direccion_correcta = direccion_corazon == direccion_jugador;
                // Si la dirección es la correcta podemos sumar la cantidad de
                puntos que le corresponda
114
115             if (direccion_correcta) {
116                 mtx_cout.lock();
117                 cout << nombre << ": mira hacia la dirección del corazón." <<
                    endl << flush; // Imprimimos que efectivamente mira hacia la
                    dirección del corazón
118                 mtx_cout.unlock();
119
120                 mtx_operacion.lock();
121                 puntos_ronda = puntosParaGanar[puestoJugador]; // Hacer que
                    gane los puntos correspondientes
122                 if ( puestoJugador < 2) { // Si es 2 es el máximo de puntos que
                    puede ganar según el array , luego al inicio de cada ronda
                    se resetea ( puestoJugador = 0)
123                     puestoJugador++;
124                 }
125                 mtx_operacion.unlock();
126             } else { //si no mira hacia la dirección en la que se ha puesto el
                corazón, se le suma 0 puntos
127                 mtx_cout.lock();
128                 cout << nombre << ": mira hacia la dirección incorrecta." <<
                    endl << flush; // No ha mirado hacia la misma dirección
129                 mtx_cout.unlock();
130
131                 mtx_operacion.lock();
132                 puntos_ronda = 0;
133                 mtx_operacion.unlock();
134             }
135
136             mtx_cout.lock();

```

```

137     cout << nombre << " ha ganado: " << puntos_ronda << " puntos." <<
138         endl << flush;
139     mtx_cout.unlock();
140
141     // Sumo los puntos
142     // No hace falta candado, cada jugador accede a su posición en el
143     // array sin bloqueos ni nada por el estilo
144     puntos[num_jugador] += puntos_ronda;
145
146     // Si es el último jugador que actúa, liberamos el semáforo de fin
147     // de ronda
148     if (jugadores_actuales == num_jugadores) {
149         sem_signal(fin_ronda); // indicamos que la ronda ha finalizado
150         sem_signal(puedo_generar); // indicamos que ya puede generar el
151         // corazón
152     }
153 }
154
155 int main() {
156     // Lanzamos hebra NPC y jugadores
157     thread hebra_NPC(funcion_hebra_NPC);
158     thread hebra_jugador[num_jugadores];
159     for (int i = 0; i < num_jugadores; ++i)
160         hebra_jugador[i] = thread(funcion_hebra_jugador, i);
161
162     // Esperamos a que terminen
163     hebra_NPC.join();
164     for (int i = 0; i < num_jugadores; ++i)
165         hebra_jugador[i].join();
166
167     // Mostramos los puntos finales
168     mtx_cout.lock();
169     cout << endl << "++++++-----+++++-----+++++ FIN DEL JUEGO
170         ++++++-----+++++-----+++++ " << endl << flush;
171     mostrar_puntos();
172     mtx_cout.unlock();
173
174     cout << R"(
175 M   M   A   RRRR   III   000       PPPP   A   RRRR   TTTT   Y   Y
176 MM MM   A A R   R   I   O   O       P   P A A R   R   T       Y Y
177 M M M   AAAAA RRRR   I   O   O       PPPP   AAAAA RRRR   T       Y
178 M   M A   A R   R   I   O   O       P   A   A R   R   T       Y
179 M   M A   A R   R   III   000       P   A   A R   R   T       Y
180 )" << endl;

```



3 Explicación detallada del Código

3.1. Introducción

En este documento se presenta la implementación de una solución para un minijuego de Mario Party. En este juego, los jugadores deben pulsar una dirección y, si coincide con la dirección aleatoria en la que se ha colocado el corazón, el primer jugador que llegue al corazón obtiene 5 puntos, el segundo 3 puntos y los demás 2 puntos. A continuación, se detalla la lógica y las estructuras utilizadas para implementar esta solución.

3.2. Estructuras Utilizadas

Para la implementación se han utilizado las siguientes estructuras abstractas de semáforos de la clase `scd.cpp`:

- **corazón_disponible**: Indica cuándo el corazón está disponible.
- **puede_generar**: Indica cuándo se puede generar un nuevo corazón.
- **fin_ronda**: Asegura que las impresiones de mensajes y operaciones no se vean afectadas por condiciones de carrera.

3.3. Uso de Mutex

Se han utilizado numerosos candados/mutex para asegurar la exclusión mutua en las operaciones críticas. Aunque se han usado más de los necesarios, se ha hecho con el propósito de dejar claro que cada operación utiliza un mutex distinto.

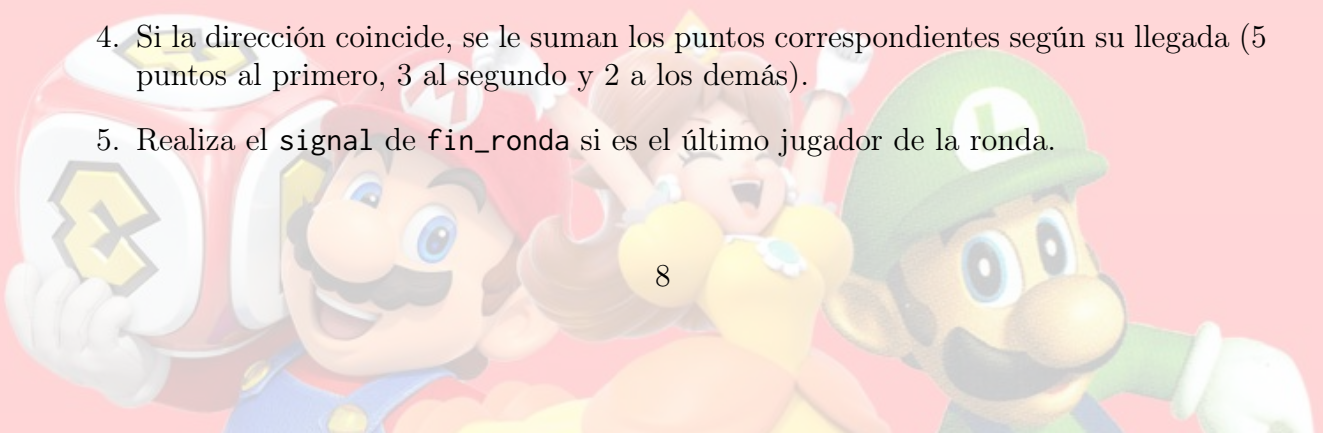
3.4. Variable de Fin del Juego

Se ha utilizado una variable booleana `fin` para indicar el fin del juego. En este caso, se realizan las operaciones correspondientes como la espera de los jugadores y otras acciones necesarias.

3.5. Hebra Jugador

La hebra del jugador sigue la siguiente lógica:

1. Espera a que el corazón esté disponible mediante el semáforo `corazon_disponible`.
2. Comprueba si el juego no ha terminado.
3. Calcula una dirección aleatoria hacia donde mira el jugador.
4. Si la dirección coincide, se le suman los puntos correspondientes según su llegada (5 puntos al primero, 3 al segundo y 2 a los demás).
5. Realiza el `signal` de `fin_ronda` si es el último jugador de la ronda.



3.6. Hebra NPC

La hebra NPC se encarga de:

1. Recorrer las rondas, reasignando el valor 0 al inicio de cada ronda a las variables `jugadores_actuales` y `puestoJugador`.
2. Realizar el `signal` a todos los jugadores cada vez que se genera un corazón.
3. Esperar al semáforo de `fin_ronda`.
4. Indicar a cada jugador que el juego ha terminado cuando la variable booleana `fin` sea verdadera.

3.7. Asignación de Puntos

Para la asignación de puntos, se ha establecido un array con los valores 5, 3 y 2. La variable `puestoJugador` determina el puesto del jugador y se le asignan los puntos correspondientes:

- Si `puestoJugador` es 0, se asignan 5 puntos.
- Si `puestoJugador` es 1, se asignan 3 puntos.
- En otros casos, se asignan 2 puntos.

3.8. Conclusión

El código está detallado línea por línea y se han implementado funciones auxiliares para mejorar la coherencia y complejidad del mismo. A continuación, se presentan capturas de la ejecución del código compilado mediante la instrucción proporcionada en el fichero `sol_SallamiMorenoIsmael_SCD_P1yP2.txt`.

4 Ejemplo de salida

```
1 NPC: Corazón visible en la dirección 2 .
2 Mario: mira hacia la dirección incorrecta.
3 Mario ha ganado: 0 puntos.
4 Peach: mira hacia la dirección incorrecta.
5 Peach ha ganado: 0 puntos.
6 Luigi: mira hacia la dirección incorrecta.
7 Luigi ha ganado: 0 puntos.
8 Bowser: mira hacia la dirección incorrecta.
9 Bowser ha ganado: 0 puntos.
10 ++++++-----+++++-----++++ FIN DE LA RONDA 1 ++++++-----+++++-----++++
11 Puntos: 0 0 0 0
12
13
14 NPC: Corazón visible en la dirección 2 .
15 Mario: mira hacia la dirección incorrecta.
16 Mario ha ganado: 0 puntos.
```

17 Peach: mira hacia la dirección incorrecta.
 18 Peach ha ganado: 0 puntos.
 19 Luigi: mira hacia la dirección incorrecta.
 20 Luigi ha ganado: 0 puntos.
 21 Bowser: mira hacia la dirección incorrecta.
 22 Bowser ha ganado: 0 puntos.
 23 ++++++-----+++++-----++++ FIN DE LA RONDA 2 ++++++-----+++++-----++++
 24 Puntos: 0 0 0 0
 25
 26
 27 NPC: Corazón visible en la dirección 1 .
 28 Mario: mira hacia la dirección incorrecta.
 29 Mario ha ganado: 0 puntos.
 30 Peach: mira hacia la dirección incorrecta.
 31 Peach ha ganado: 0 puntos.
 32 Luigi: mira hacia la dirección incorrecta.
 33 Luigi ha ganado: 0 puntos.
 34 Bowser: mira hacia la dirección incorrecta.
 35 Bowser ha ganado: 0 puntos.
 36 ++++++-----+++++-----++++ FIN DE LA RONDA 3 ++++++-----+++++-----++++
 37 Puntos: 0 0 0 0
 38
 39
 40 NPC: Corazón visible en la dirección 1 .
 41 Mario: mira hacia la dirección incorrecta.
 42 Mario ha ganado: 0 puntos.
 43 Peach: mira hacia la dirección del corazón.
 44 Peach ha ganado: 5 puntos.
 45 Luigi: mira hacia la dirección incorrecta.
 46 Luigi ha ganado: 0 puntos.
 47 ++++++-----+++++-----++++ FIN DE LA RONDA 4 ++++++-----+++++-----++++
 48 Puntos: 0 0 5 0
 49
 50
 51 NPC: Corazón visible en la dirección 1 .
 52 Bowser: mira hacia la dirección incorrecta.
 53 Bowser ha ganado: 0 puntos.
 54 Mario: mira hacia la dirección incorrecta.
 55 Mario ha ganado: 0 puntos.
 56 Peach: mira hacia la dirección incorrecta.
 57 Peach ha ganado: 0 puntos.
 58 Luigi: mira hacia la dirección incorrecta.
 59 Luigi ha ganado: 0 puntos.
 60 ++++++-----+++++-----++++ FIN DE LA RONDA 5 ++++++-----+++++-----++++
 61 Puntos: 0 0 5 0
 62
 63
 64 NPC: Corazón visible en la dirección 2 .
 65 Bowser: mira hacia la dirección incorrecta.
 66 Bowser ha ganado: 0 puntos.
 67 Mario: mira hacia la dirección del corazón.
 68 Mario ha ganado: 5 puntos.
 69 Peach: mira hacia la dirección del corazón.
 70 Peach ha ganado: 3 puntos.
 71 Luigi: mira hacia la dirección incorrecta.
 72 Luigi ha ganado: 0 puntos.



```

73 Bowser: mira hacia la dirección incorrecta.
74 Bowser ha ganado: 0 puntos.
75 ++++++-----+++++-----+++++ FIN DE LA RONDA 6 ++++++-----+++++-----+++++
76 Puntos: 5 0 8 0
77
78
79 NPC: Corazón visible en la dirección 3 .
80 Mario: mira hacia la dirección del corazón.
81 Mario ha ganado: 5 puntos.
82 Peach: mira hacia la dirección incorrecta.
83 Peach ha ganado: 0 puntos.
84 Luigi: mira hacia la dirección incorrecta.
85 Luigi ha ganado: 0 puntos.
86 Bowser: mira hacia la dirección incorrecta.
87 Bowser ha ganado: 0 puntos.
88 ++++++-----+++++-----+++++ FIN DE LA RONDA 7 ++++++-----+++++-----+++++
89 Puntos: 10 0 8 0
90
91
92 NPC: Corazón visible en la dirección 0 .
93 Mario: mira hacia la dirección del corazón.
94 Mario ha ganado: 5 puntos.
95 Peach: mira hacia la dirección incorrecta.
96 Peach ha ganado: 0 puntos.
97 Luigi: mira hacia la dirección incorrecta.
98 Luigi ha ganado: 0 puntos.
99 Bowser: mira hacia la dirección incorrecta.
100 Bowser ha ganado: 0 puntos.
101 ++++++-----+++++-----+++++ FIN DE LA RONDA 8 ++++++-----+++++-----+++++
102 Puntos: 15 0 8 0
103
104
105 NPC: Corazón visible en la dirección 2 .
106 Mario: mira hacia la dirección incorrecta.
107 Mario ha ganado: 0 puntos.
108 Peach: mira hacia la dirección incorrecta.
109 Peach ha ganado: 0 puntos.
110 Luigi: mira hacia la dirección incorrecta.
111 Luigi ha ganado: 0 puntos.
112 Bowser: mira hacia la dirección incorrecta.
113 Bowser ha ganado: 0 puntos.
114 ++++++-----+++++-----+++++ FIN DE LA RONDA 9 ++++++-----+++++-----+++++
115 Puntos: 15 0 8 0
116
117
118 NPC: Corazón visible en la dirección 2 .
119 Mario: mira hacia la dirección incorrecta.
120 Mario ha ganado: 0 puntos.
121 Peach: mira hacia la dirección del corazón.
122 Peach ha ganado: 5 puntos.
123 Luigi: mira hacia la dirección incorrecta.
124 Luigi ha ganado: 0 puntos.
125 Bowser: mira hacia la dirección incorrecta.
126 Bowser ha ganado: 0 puntos.
127 ++++++-----+++++-----+++++ FIN DE LA RONDA 10 ++++++-----+++++-----+++++
128 Puntos: 15 0 13 0

```



```

129
130
131
132 ++++++-----+++++-----+++++ FIN DEL JUEGO DE MARIO PARTY CON 10 RONDAS
    ++++++-----+++++-----+++++
133 Puntos: 15 0 13 0
134
135
136
137 ++++++-----+++++-----+++++ FIN DEL JUEGO ++++++-----+++++-----+++++
138 Puntos: 15 0 13 0
139
140
141
142 M M A RRRR III 000 PPPP A RRRR TTTT Y Y
143 MM MM A A R R I O O P P A A R R T Y Y
144 M M M AAAAA RRRR I O O PPPP AAAAA RRRR T Y
145 M M A A R R I O O P A A R R T Y
146 M M A A R R III 000 P A A R R T Y

```

5 Comando de compilación

```

1 g++ -std=c++11 -pthread sol_SallamiMorenoIsmael_SCD_P1yP2.cpp scd.cpp -o
  sol_SallamiMorenoIsmael_SCD_P1yP2

```

6 Material

Material

