Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 – Estructuras de Datos y Algoritmos

2025 - 2

Tarea 0

Fecha de entrega código: 24 de Agosto, 23:59 Chile continental

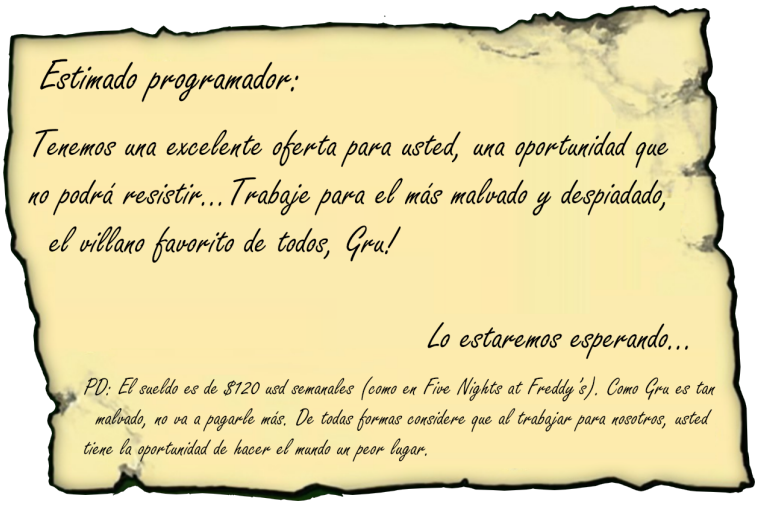
Link a generador de repos: CLICK AQUÍ

Objetivos

- Diseñar algoritmos simples en C.
- Comprender la diferencia entre Arrays y Listas Ligadas.
- Familiarizarse con el uso de punteros y manejo de memoria.

Introducción

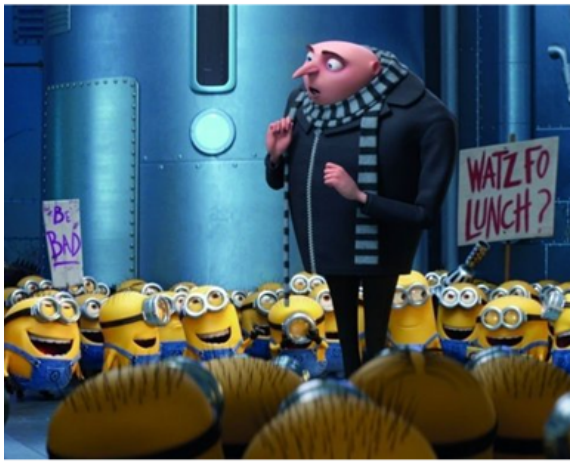
Mientras vuelves a tu casa después de un largo día de trabajo, encuentras a una sospechosa sombra merodeando por la entrada de tu domicilio. Antes de que puedas descifrar qué es lo que estás viendo, la criatura se escapa. Sin embargo, antes de escabullirse, la pequeña entidad de color amarillo deja una malvada carta a los pies de tu puerta. Con curiosidad, abres la carta, y lees lo siguiente:



Cansado de ser explotado en tu trabajo de medio tiempo, decides que definitivamente vale la pena una cosa por otra. Al día siguiente ya tienes listas tus maletas y partes con entusiasmo a trabajar para el más malvado villano que ha visto jamás la historia.

¡Ayuda a Gru a organizar sus minions!

La casa de Gru es gigante, tan grande como es necesario para llevar a cabo sus malévolos planes. Por lo tanto, cada vez es más difícil organizar y llevar la cuenta de cuantos minions hay en cada lugar. Es por esto que Gru decidió contactarte a ti, para automatizar el ingreso, salida, bienestar y cuenta total de todos los minions.



Problema

Debes implementar diversas funciones que permitan reflejar el estado actual de la casa de Gru y de sus minions. Para lo anterior, la casa será dividida en un número fijo de sectores. En cada sector puede ingresar, salir o cambiar una cantidad de minions arbitraria.

El bienestar de los minions también es muy importante... ¡Los minions tienen amigos! En algunas ocasiones deberás asegurarte de que los mejores amigos puedan reunirse. También tendrás que modelar su comportamiento entusiasta cuando vean ciertos objetos... (como una banana).

Entidades

Minion: Cada minion cuenta con un `ID` único, una profesión `JOB` de hasta 14 caracteres, y `BF_ID` que indica la `ID` de su mejor amigo. Además, posee el atributo `evil`, que toma valor 0 si se trata de un minion bueno y valor 1 si es un minion malvado.

Sector: Cada sector tiene un `ID` único que va desde 0 a $S-1$ y almacena un número arbitrario de minions. Por defecto, el primer minion en entrar se ubica al inicio del sector y el último al final, aunque ciertos eventos pueden modificar este orden.

Flujo

Primero, recibirás una línea con el entero S, que representa la cantidad de sectores. Luego, recibirás una línea con el entero E que indica la cantidad de eventos a recibir. A continuación, se te entregarán los E eventos, donde cada uno puede contener más de una línea de input que deberás procesar.

Parte 1 - Eventos

`ENTER {sector_id} {minion_id} {job} {bf_id} {evil}`

Este evento ingresa al sector[1] con ID `sector_id` un nuevo minion con ID `minion_id`, profesión `job`, ID de mejor amigo `bf_id` y maldad `evil`.

El resultado esperado debe mostrar el siguiente output:

output

```
Minion {minion_id} ingreso al sector {sector_id}.
```

[1] El sector siempre estará activo, es decir, no explotado por el evento `EXPLOTION`.

LEAVE {sector_id} {minion_id}

Este evento indica que el minion `minion_id` ha abandonado el sector `sector_id`. Existe la posibilidad de que el minion indicado no se encuentre en el sector, en cuyo caso se entregará un output distinto. Finalmente, se puede asumir que si un minion que se marcha, este no volverá a ser mencionado en ningún evento posterior.

El resultado esperado debe mostrar el siguiente output:

- Caso en que el minion indicado se encuentra en el sector.

output

```
Minion {minion_id} abandono el sector {sector_id}.
```

- Caso en que el minion indicado **NO** se encuentra en el sector.

output

```
No se encontro al minion {minion_id} en el sector {sector_id}.
```

SECTOR-INFO {sector_id}

Este evento muestra para un sector activo en particular su número total de minions. Luego, muestra para cada minion su `ID` y si es bueno o malvado según el atributo `evil`, siguiendo el orden actual en que se encuentran en el sector. [2]

El resultado esperado debe mostrar el siguiente output:

output

```
SECTOR-INFO:
    Sector {sector_id} - {total_minions} minions:
        Minion bueno con id {minion_id}
        Minion malvado con id {minion_id}
        ...
        Minion bueno con id {minion_id}
```

STATUS

Este evento muestra las estadísticas de todos los sectores. Para cada sector activo[3] se muestra su `ID` y número total de minions.

El resultado esperado debe mostrar el siguiente output:

output

```
STATUS:
    Sector {0}: {total_minions} minions
    ...
    Sector {S-1}: {total_minions} minions
```

[2] Por defecto, están por orden de llegada, pero recuerda que esto puede ser modificado por otros eventos.
[3] En el caso de un sector explotado, no se muestra información.

EXPLOSION {sector_id}

El sector con id `sector_id` ha explotado, debido a una torpe falla técnica... Todos los minions de este sector abandonan el lugar y el sector no será utilizado en el futuro.

Cabe destacar que una vez explotado, un sector no muestra su información en el evento `STATUS` y no será consultado directamente por otros eventos vía argumentos. El resultado esperado debe mostrar el siguiente output:

output

```
1 El sector {sector_id} ha explotado.
```

CONTAMINATE {sector_id}

¡El sector con id `sector_id` ha sido contaminado! Ahora, todos los minions de este sector han mutado y se han vuelto malvados. Debes cambiar a 1 el atributo `evil` de todos los minions buenos en el sector.

El resultado esperado debe mostrar el siguiente output:

output

```
1 El sector {sector_id} ha sido contaminado.
```

Parte 2 - Eventos

NEW-BEST-FRIEND {sector_id} {minion_id} {new_bf_id}

Este evento indica que el minion con ID `minion_id` del sector con ID `sector_id` ha peleado con su mejor amigo, porque este se comió su banana... Ahora, el minion tiene un nuevo mejor amigo de ID `new_bf_id`, quien esperamos no se coma su banana también.

El resultado esperado debe mostrar el siguiente output:

output

```
1 Minions {minion_id} y {new_bf_id} ahora son mejores amigos.
```

SWITCH {minion_1_id} {minion_2_id}

Para este evento, los minions de IDs `minion_1_id` y `minion_2_id` deben intercambiarse de lugar. Además, debes considerar que no necesariamente se encuentran en el mismo sector.

El resultado esperado debe mostrar el siguiente output:

output

```
1 Minions {minion_1_id} y {minion_2_id} cambiaron de lugar.
```

BANANA {sector_id}

Una banana es encontrada en el sector con ID `sector_id`. Debido a su amor por las bananas, los minions de los sectores adyacentes activos corren amontonados (pero en orden) hacia el sector con la banana. Para encontrar los sectores adyacentes activos, debes buscar desde el sector con ID `sector_id` un sector con menor ID y otro con mayor ID que no hayan sido explotados por el evento `EXPLOSION`.

Para efectos de la tarea, los minions del sector con menor ID ingresan antes que el sector de mayor ID. Además, si algún sector no se encuentra disponible, se ignora y se sigue buscando uno que lo esté.[4]

[4] Siempre encontrarás un sector disponible con menor ID y otro con mayor ID. Además, el sector encontrado podría no tener minions.

El resultado esperado debe mostrar el siguiente output[5]:

```
output
1 Una banana ha aparecido en el sector {sector_id}:
2     Minions de los sectores {A} y {B} fueron a comerse la banana.
```

`INVERSE {sector_id}`

Este evento invierte el orden de los minions, es decir, ahora el primer minion será el último, y el último será el primero. Así, todos los minions invierten su orden, como si hubieran ingresado en el orden reverso. Este evento no tiene output.

`MOVE-BEST-FRIEND {sector_id} {minion_id}`

El minion con ID `minion_id` del sector `sector_id` extraña mucho a su mejor amigo, por lo que decide buscarlo dentro del mismo sector. Si es que lo encuentra, el minion se mueve justo detrás de él. En otro caso, el minion se mueve al final del sector.

El resultado esperado debe mostrar el siguiente output:

- Caso en que el mejor amigo se encuentra en el sector.

```
output
1 Minion {minion_id} se sento junto a su mejor amigo.
```

- Caso en que el mejor amigo **NO** se encuentra en el sector

```
output
1 Minion {minion_id} se sento en la ultima posicion.
```

`LEAVE-JOB {sector_id} {job}`

Todos los minions del sector con ID `sector_id` que tengan un trabajo igual a `job` deben marcharse a trabajar inmediatamente. Para este evento se espera que todos los minions con dicho trabajo abandonen el sector.

El resultado esperado debe mostrar el siguiente output:

```
output
1 Minions con el trabajo {job} abandonaron el sector {sector_id}.
```

Consideraciones Generales

- El mejor amigo de un minion no siempre se encuentra en el mismo sector y podría no estar en ninguno.
- Una vez que un sector explota, este no será recibido como input en eventos futuros.
- Los minions que abandonaron por `LEAVE`, `LEAVE-JOB` o `EXPLOSION` no serán recibidos como input en eventos futuros.
- Toda ID es un entero no negativo único.[6]
- Las profesiones de minions que aparecen en los tests públicos no necesariamente serán las mismas que se utilicen en los tests privados. Es importante que consideres esto al modelar tu solución.
- Se encuentra estrictamente prohibido el uso de la función `realloc`.

[5] A y B siendo las ids de los sectores adyacentes.
[6] Podría coincidir la ID de un sector con la de un minion, pero no entre dos minions o dos sectores.

Ejecución

Tu programa se debe compilar con el comando `make` y debe generar un ejecutable de nombre `minions` que se ejecuta con el siguiente comando:

```
./minions input output
```

donde `input` será un archivo con los eventos a simular y `output` el archivo donde se guardarán los resultados.

En caso de querer correr el programa para ver los leaks de memoria utilizando valgrind deberás utilizar el siguiente comando:

```
valgrind ./minions input output
```

Tu tarea será ejecutada con archivos de creciente dificultad, asignando puntaje a cada una de estas ejecuciones que tenga un output igual al esperado. A continuación detallaremos los archivos de input y output.

Input

El archivo de input comenzará con el número `S` que indica la cantidad máxima de sectores, posterior a ello se entrega un número `E` que corresponde al número de eventos a recibir, y las `E` líneas correspondientes a cada uno.

Como ejemplo tenemos el siguiente input:

input

```
4 # Cantidad de sectores
9 # Cantidad de eventos
ENTER 3 1 Scientist 2 0
ENTER 3 2 Engineer 3 0
ENTER 1 3 Spy 2 1
ENTER 2 4 Scientist 8 1
CONTAMINATE 3
EXPLOSION 1
LEAVE 2 4
STATUS
SECTOR-INFO 3
```

Output

output

```
Minion 1 ingreso al sector 3.
Minion 2 ingreso al sector 3.
Minion 3 ingreso al sector 1.
Minion 4 ingreso al sector 2.
El sector 3 ha sido contaminado.
El sector 1 ha explotado.
Minion 4 abandono el sector 2.
STATUS:
    Sector 0: 0 minions
    Sector 2: 0 minions
    Sector 3: 2 minions
SECTOR-INFO:
    Sector 3 - 2 minions:
        Minion malvado con id: 1
        Minion malvado con id: 2
```

Evaluación

La nota de tu tarea es calculada a partir de testcases de input/output que se evaluarán cada uno en forma binaria, es decir, 1 punto por output correcto y 0 puntos por output incorrecto. La ponderación se descompone de la siguiente forma:

Nota entre partes (90 %)	**Manejo de memoria (10 %)**
30 % Tests Parte 1	5 % Sin leaks de memoria
60 % Tests Parte 2	5 % Sin errores de memoria

Para cada test de evaluación, tu programa deberá entregar el output correcto en **menos de 5 segundos** y utilizar menos de 1 GB de RAM[7], de lo contrario recibirás 0 puntos en ese test. Por último, está **estrictamente prohibido** el uso de la función `realloc`, cuyo uso implicará que seas evaluado con la nota mínima.

Recomendación de tus ayudantes

Estas tareas generalmente requieren de mucha dedicación, por lo que desde ya te recomendamos distribuir tu tiempo considerando los plazos definidos (Recuerda que la entrega es el 24 de Agosto). Asímismo, te recomendamos fuertemente que antes de empezar a programar tu tarea, **leas el enunciado detenidamente y con anticipación para que te dediques a entender de manera profunda lo que te pedimos**. Una vez hayas comprendido el enunciado, dedica el tiempo que sea necesario en planificar y modelar los problemas, para posteriormente poder programar de manera más eficiente. No olvides compilar el código como se indica en la tarea y de preguntar cualquier duda o por problemas que te surjan en Discussions. Estos son consejos de tus ayudantes que te pueden ayudar a pasar el ramo.

Entrega

Código: GIT - Rama principal del repositorio asignado, que debes generar con el link dado al comienzo de este enunciado. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

Atraso: Esta tarea **NO** considera la política de atraso y cupones, es decir, no se considerarán tareas entregadas fuera de plazo.

Integridad académica

Este curso se adscribe al Código de Honor establecido por la Escuela de Ingeniería. Todo trabajo evaluado en este curso debe ser hecho **individualmente** por el alumno y **sin apoyo de terceros**. Se espera que los alumnos mantengan altos estándares de honestidad académica, acorde al Código de Honor de la Universidad. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Pregrado de la Escuela de Ingeniería.

Uso de IA

Se prohíbe el uso de herramientas de inteligencia artificial para la realización de esta tarea. Esto incluye, pero no se limita al uso de modelos de lenguaje como ChatGPT, Copilot, u otras tecnologías similares para la generación automática de código, soluciones, o cualquier parte del trabajo requerido. Nos reservamos el derecho de revisar todas las tareas entregadas con el fin de detectar el uso de inteligencia artificial en la creación de las soluciones. Las tareas en las que se determine que se ha utilizado IA serán consideradas como una infracción a la política de honestidad académica y serán tratadas como casos de copia.

[7] Puedes revisarlo con el comando `htop` u ocupando `valgrind --tool=massif`