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

IIC2133 — Estructuras de Datos y Algoritmos
2021 - 2

Tarea 0

Fecha de entrega código e informe: Por definir

Objetivos

- Familiarizar al estudiante con el lenguaje de programación `C`.
- Entender e implementar estructuras de datos que combinan arreglos, nodos y listas ligadas.

Introducción

Con el avance del plan de vacunación mundial, el DCCwarts está volviendo a tener clases presenciales, lo que genera un gran desafío para los aprendices de magia alrededor del mundo. Peter Potter es el presidente del centro de estudiantes de DCCwarts y tiene como trabajo organizar y simular los traslados del alumnado para poder mandar el tren mágico a buscarlos. Sin embargo, debido a la cantidad de estudiantes, Peter no pudo simular los viajes con su magia y decidió buscar ayudas computacionales, por lo que llegó a ti, insigne aprendiz de DCCwarts. Tu trabajo es ayudarlo en simular este proceso.

Estación $9\frac{3}{4}$

Problema

El problema consiste en transportar a todos los magos que desean viajar. Para esto debes construir las estaciones de trenes, andenes y las filas de pasajeros. Luego debes simular la llegada de pasajeros y viajes de tren.

El mundo con el que trabajarás contará con E estaciones de trenes y cada estación tendrá una cantidad de andenes A_e. Para identificar cada estación estas irán numeradas de 0 a $E-1$ y en cada estación los andenes irán numerados entre 0 y $A_e - 1$.

Los Andenes se componen por una fila de pasajeros y el espacio para la llegada de un tren. Además, por el caos en la red de trenes, se ha implementado dos tipos de pasajes, que pueden ser Premium (0) o Normal (1). Estos pasajes definen el orden en la fila de pasajeros y orden de subida al tren, que se explica en detalle en la sección de Eventos.

Los trenes se componen por una cantidad variable de vagones y cada vagón tiene una cantidad c_V y que no puede cambiar de asientos. El identificador de cada tren consiste en un número que parte en 0 para el primer tren y va creciendo en la medida que se crea un nuevo tren.

Cada vez que se sube un pasajero al tren, se ubicará siempre en el primer asiento disponible. Este es en el primer vagón que tenga un asiento desocupado y en el primer asiento sin pasajeros.

Eventos

A continuación se describen los distintos eventos que debes simular:

`NUEVO-TREN estación andén vagones capacidad`$_1$ `...` `capacidad`$_v$

Indica que se crea un tren en la estación con id `estación` y andén con id `andén`. El tren tiene la cantidad de vagones indicada con `vagones` y cada vagón i tiene capacidad para `capacidad`$_i$ personas. El tren inicialmente se encuentra vacío y la capacidad de los vagones no cambia.

`PASAJERO estación andén destino categoría`

Llega un nuevo pasajero a la estación y andén mencionado que se dirige a la estación `destino`. La `categoría` puede ser Premium (0) o Normal (1). En caso de que el pasajero sea Premium, haya un tren en el andén y hay asientos disponibles, el pasajero se sube de inmediato al tren y se ubica en el primer asiento disponible. En otro caso el pasajero se ubica al final de la fila de su categoría. Para identificar a los pasajeros, estos tendrán un id que consiste en un entero que parte en 0 y crece a medida que llega un nuevo pasajero.

Diagrama que muestra la llegada de un pasajero sin un tren en el andén

Diagrama que muestra la llegada de pasajeros con tren en el andén

`REMOVER estación andén vagón asiento`

Dado que DCCwarts no acepta estudiantes comercialggles, hay comercialggles que consiguen los pasajes de manera ilegal abordando el tren. Afortunadamente, los trenes poseen magias para detectar si hay comercialggles en algún asiento, una vez detectado un comercialggle, va a ser removido de su asiento y del tren.
Es por eso que debes eliminar al pasajero que se encuentra en el vagón y asiento mencionado del tren que se encuentra en la `estación` y `andén` dados. Luego debes imprimir REMOVER id_pasajero destino

`SALIR estación-inicial andén-inicial`

Indica el viaje del tren que se encuentra en la `estación-inicial` y `andén-inicial`. Antes de partir, se deben subir todos los pasajeros que se encuentran en la fila mientras queden asientos disponibles. Si hay asientos disponibles solo deberían subir pasajeros con pasaje Normal (1).

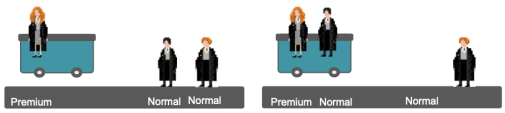

Diagrama que muestra la subida de pasajeros una vez que sale el tren

Este evento siempre es seguido de alguno de los tres eventos siguientes:

`LLEGAR estación-destino andén-destino`

El mismo tren que salió recientemente llega a la estación y andén de destino. Primero debes imprimir `LLEGAR` seguido del estado del tren (descrito en el evento `ESTADO TREN`).
Al llegar se deben bajar todos los pasajeros cuyo destino sea la estación actual, los pasajeros que no se bajan no se deben mover de sus asientos. Luego, si hay pasajeros Premium esperando y hay asientos disponibles estos se deben subir al tren de inmediato.

`DESAPARECER`

Debido a hechizos maliciosos ocurridos dentro del tren, el tren se perdió en un universo paralelo y no puede ser encontrado. En caso de que este evento suceda debes imprimir `DESAPARECER` seguido del estado del tren.

`SEPARAR estación-destino-1 andén-destino-1 estación-destino-2 andén-destino-2`

Durante el trayecto de los trenes se encontraron intrusos que no eran estudiantes de DCCWarts sino de EAAWarts.
Por petición de Peter Potter, se debe separar el tren en dos nuevos trenes. Todos los vagones de posición par (partiendo por el 0) van al destino 1 y todos los vagones de posición impar van al destino 2. Al llegar se cumplen las mismas reglas que en el evento `LLEGAR` primero con el tren que se dirige al destino-1 y luego con el tren que se dirige al destino-2. Por último, para mantener consistencia con los identificadores, el tren que se dirige al destino-1 mantiene el id y el que se dirige al destino-2 crea un nuevo id.

Tren original con id = 5

A la izquierda el primer tren con id = 5 y a la derecha el segundo tren con id = 9

`ESTADO TREN`

Esta instrucción no se pedirá directamente, pero es parte de `LLEGADA`, `DESAPARECER` y `ESTACIÓN`. Se usará para describir el estado de un tren en específico.
Debes imprimir `T` seguido del id del tren, luego debes imprimir cada asiento. Si hay un pasajero se imprime su id junto con su lugar de destino, en cambio, si el asiento está vacío se imprime `X`. Para separar cada vagón imprimes `|`.

Por ejemplo, la representación de un tren (id 13) con dos vagones con dos asientos cada uno, donde solo en el primer asiento del primer vagón hay un pasajero cuyo id es 7 y su destino es la estación 2 es:

```
T13 | 7-2 X | X X |
```

`ESTACIÓN estación`

Debes reportar el estado de la estación cada vez que se indique la instrucción. Primero debes imprimir `ESTACION id`, Luego, para cada andén debes imprimir `A` junto con el id del andén seguido de la fila de pasajeros.

La fila de pasajeros se incluye primero todos los pasajeros Premium y luego los Normal. Hay que indicar `id-destino-pasaje`.

Por último, de haber un tren en el andén, se imprime como se mencionó recientemente.

Ejemplo para una estación (id 2) con dos andenes, el primer andén tiene 3 personas con ids 0, 1, 2; destino 2, 3, 3; los tres con prioridad Normal y el mismo tren anterior.
El segundo andén tiene una persona Premium y una normal representadas por `4-0-0` y `3-0-1`.
El resultado es el siguiente:

```
ESTACION 2
A0 0-2-1 1-3-1 2-3-1
T13 | 7-2 X | X X |
A1 4-0-0 3-0-1
```

Recomendación de tus ayudantes

Las tareas requieren de mucha dedicación de tiempo generalmente, por lo que desde ya te recomendamos distribuir tu tiempo considerando los plazos definidos. Así mismo, te recomendamos fuertemente que antes de empezar a programar tu tarea, leas el enunciado y te dediques a entender de manera profunda lo que te pedimos. Una vez que hayas comprendido el enunciado, dedica el tiempo que sea necesario para la planificación y modelación de tu solución, para posteriormente poder programar de manera eficiente. Estos son consejos de tus ayudantes que te puede ayudar a pasar el ramo :)

Archivos iniciales

Para ayudarte a empezar a programar y que te centres en la creación de estructuras y sus funciones, los ayudantes han creado un código base que incluye la lectura de archivos. Este se encuentra en tu repositorio cuyo nombre es `T0-2021-2-{usuario-github}`. Recuerda aceptar la invitación cuando te llegue ya que esta vence luego de 7 días. Si tienes algún problema con tu repositorio escribe una issue o pregunta por Discord.

Ejecución

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

```
./magictrains <input> <output>
```

Donde `input` será un archivo con los eventos a simular y `output` el archivo a guardar los resultados.

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

Input

La información del archivo de `input` viene entregada en el siguiente formato:

- La primera linea tiene un número N que indica la cantidad de estaciones.
- Luego vienen N lineas donde la iésima linea indica la cantidad A_e de andenes que tiene la Estación e.
- Luego cada línea contiene un evento, como fueron detallados en la sección Eventos.
- Finalmente, una linea conteniendo la palabra `END` que indica el fin del archivo.

Puedes asumir que los eventos detallados en el archivo de input siempre serán válidos y ejecutables para el estado que debería tener el programa. Específicamente, todos los trenes tienen al menos 1 vagón y para el evento `SEPARAR` el tren tendrá al menos 2 vagones; siempre existirá un tren para el evento de `SALIR`; el andén de destino siempre se encontrará vacío.

Output

Tu archivo de output debe incluir los estados de estación o trenes cada vez que se piden. También, al finalizar el programa (con la línea de input `END`) debes imprimir el estado de cada estación como se describe en `ESTACIÓN`.

Recomendamos ver los ejemplos subidos para que tus archivos de `output` finales sean los iguales a los esperados.

Uso de memoria

Parte de los objetivos de esta tarea es que trabajen solicitando y liberando memoria manualmente. Para evaluar esto, usaremos *valgrind*. Se recomienda fuertemente ver los videos de este repositorio.

Para asegurarte que no tengas errores de memoria debes correr tu programa con:

`valgrind ./magictrains <input> <output>`

y el output debe contener

`"All heap blocks were freed -- no leaks are possible"` y
`"ERROR SUMMARY: 0 errors from 0 contexts"`

Análisis

Deberás escribir un informe de análisis[1] donde menciones los siguientes puntos:

- Describe y justifica la estructura de datos usada para cada elemento (persona, andén, tren etc.).
- Calcula y justifica la complejidad en notación $\mathcal{O}$ para cada uno de los eventos del programa en función de la cantidad de andenes, vagones y personas, trenes y asientos.

Evaluación

La nota de tu tarea se descompone como se detalla a continuación:

- 70% a la nota de tu código, dividido en:
  - 80% que el output de tu programa sea correcto y eficiente. Para esto el puntaje se divide en partes iguales para los tests Easy, Medium y Hard.
  - 10% a que *valgrind* indique que tu programa no tiene errores de memoria.
  - 10% a que *valgrind* indique que tu programa no tiene *memory leaks*.

  Para cada test de evaluación, tu programa deberá entregar el output correcto en menos de 10 segundos. De lo contrario, recibirás 0 puntos en ese test.
- 30% a la nota del informe, dividido en:
  - 30% por usar la estructura correcta para cada elemento.
  - 70% por calcular las complejidades teóricas y justificarlas para cada uno de los eventos.

Entrega

Código: GIT - Rama master del repositorio asignado. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

Informe: SIDING - En el cuestionario correspondiente, en formato PDF. Sigue las instrucciones del cuestionario. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

Atraso: A lo largo del semestre tendrás 4 días de gracia, los cuales podrás utilizar en caso de que no alcances a entregar alguna tarea en el tiempo indicado. Para esto, puedes usar un máximo de 2 días en una

[1] en un máximo de 5 planas

tarea, sin importar si tienes más días disponibles, y tendrás que avisar si quieres que se revise un commit posterior a la fecha de entrega en el formulario que se mandará el día siguiente. Cabe destacar que si entregas a las 00:01 hrs perderás un día en caso de llenar el formulario, y no será revisado ese commit en caso de que decidas no contestarlo. Por otro lado, si se te acaban los 4 días y entregas una tarea atrasada, entonces tendrás la calificación mínima **sin derecho a reclamo.**

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.