Guía 1 — fecha límite: domingo 10 de mayo

(Actualización 01/05/2020: se hicieron ajustes en el calendario de clases, no impactan en este bloque)

Algoritmos y Estructuras de Datos II, DC, UBA.

Primer cuatrimestre de 2020

Índice

1. Modalidad de trabajo

Como todes sabemos, este cuatrimestre las clases se dictarán a distancia con motivo del Aislamiento Social Preventivo y Obligatorio dispuesto por el Decreto 297/2020. Si el aislamiento terminara antes del final del cuatrimestre y la Facultad volviera a su funcionamiento normal, informaremos cómo continuar con la cursada. Por el momento no podemos asegurar que la cursada vuelva a ser presencial, ni que siga siendo a distancia hasta el final del cuatrimestre.

1.1. Cronograma de ejercicios para esta guía

Este es un cronograma *sugerido* para que vayan trabajando. La idea no es que sea fijo, sino brindarles lo que quienes hicimos la guía pensamos que sería un ritmo realizable.

Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
20 de abril Quiz 2.1.1 Quiz 2.1.2	21 de abril Empezar 2.2	22 de abril	23 de abril	24 de abril Corrección 2.2 Empezar 2.3	25 de abril	26 de abril
27 de abril	28 de abril	29 de abril	30 de abril	1 de mayo Corrección 2.3	2 de mayo	3 de mayo
4 de mayo	5 de mayo	6 de mayo	7 de mayo	8 de mayo	9 de mayo	10 de mayo **límite entrega**

1.2. Clases

La materia se divide en clases teóricas, clases prácticas y clases de laboratorio. Como regla general, el material expositivo de las clases será grabado en diferido (no en vivo) y estará disponible en video en la sección **Clases** del sitio de la materia en el Campus[1]. Cualquier excepción a esta regla se informará con anticipación. Dado que es la primera vez que la mayoría de les docentes dicta cursos a distancia, solicitamos ser comprensives con todo tipo de falencias que las clases puedan llegar a presentar. Aceptamos todo tipo de crítica constructiva o sugerencia.

1.3. Turnos

Desde el punto de vista administrativo, docentes y alumnes estarán en un único turno.

1.4. Vías de comunicación y espacios de consulta

Toda la información importante sobre la materia se comunicará a través de la lista de correo `algo2-alu@dc.uba.ar` que pueden leer y escribir todes les docentes y alumnes de la materia. Las consultas a docentes acerca de cuestiones administrativas o de índole personal se pueden hacer a través de la lista `algo2-doc@dc.uba.ar` que pueden leer todes les docentes de la materia, y a la que pueden escribir todes les alumnes.
Las consultas acerca de temas de la materia se responderán a través de los siguientes medios:

1. Para comunicación *asincrónica* (estilo foro) en la sección **Foro de Consultas** del sitio de la materia en el Campus se pueden dejar por escrito consultas que les docentes podrán leer y responder. Aconsejamos revisar el foro antes de formular una pregunta para ver si ya fue respondida con anterioridad.

2. Para comunicación *sincrónica* (estilo chat/videoconferencia), pondremos disponibles canales de comunicación (a través de algún software que indicaremos y podrá depender de diversos factores, ej. Telegram o Zoom). El cronograma de consultas sincrónicas se ajustará a la demanda de consultas y la disponibilidad docente bajo consideración de las circunstancias especiales actuales. Buscaremos respetar que los horarios de consulta sincrónica estén dentro de los horarios de la cursada presencial con el fin de respetar las disponibilidades establecidas anteriormente. En particular, *trataremos* de que haya docentes disponibles durante las siguientes franjas horarias:

    - **Laboratorio:** miércoles de 11:00 a 14:00 (turno mañana) y de 17:00 a 20:00 (turno noche).

[1] `https://campus.exactas.uba.ar/course/view.php?id=1846`

- **Práctica:** viernes de 11:00 a 14:00 (turno mañana) y de 17:00 a 20:00 (turno noche).

1.5. Correlativas

Para poder cursar la materia se requiere tener aprobada la cursada (no así el final) de Algoritmos y Estructuras de Datos I antes del comienzo del cuatrimestre.

1.6. Guías de ejercitación y criterios de aprobación

La materia se dividirá en bloques. De manera tentativa, habría **cuatro bloques**, cada uno de **tres semanas** de duración (pero esto puede llegar a cambiar). Cada bloque tendrá una **guía** como esta, que se encontrará disponible en la sección **Guías de ejercitación** del sitio de la materia en el Campus. Cada guía incluye:

1. **Ejercicios seleccionados (no se entregan).** Preguntas teóricas y ejercicios prácticos destinados a validar que hayan entendido el material presentado en las clases teóricas. Estas preguntas discutirán en el foro y eventualmente se publicarán soluciones de algunas o varias de ellas.
2. **Ejercicios obligatorios de la práctica**. Los ejercicios son individuales y se deben entregar al finalizar el bloque. Los ejercicios serán corregidos y devueltos. Cada entrega podrá ser aprobada o devuelta para incorporar correcciones. Se detallan en la Sección 3 (margen azul).
3. **Ejercicios obligatorios del laboratorio (TPs y talleres)**. Los TPs son grupales (en grupos de 4 integrantes). Los talleres son individuales. Cada instancia se debe entregar antes de la fecha de finalización del bloque, y podrá ser aprobada o devuelta para incorporar correcciones. Se detallan en la Sección 4 (margen verde).

La forma de entrega de los ejercicios individuales y trabajos prácticos será informada oportunamente. Para aprobar los prácticos de la materia será necesario aprobar **todas** las instancias de evaluación (tras las instancias de recuperación correspondientes).

En caso de que finalice la cuarentena y la Facultad vuelva a funcionar con normalidad, se planifica también un **Examen integrador** presencial para el día viernes 17/07 (con su respectivo recuperatorio el viernes 31/07). Tener en cuenta que, dado el contexto excepcional de este cuatrimestre, el mecanismo de evaluación podrá sufrir modificaciones.

1.7. Cronograma tentativo de la materia

Se incluye un cronograma tentativo de la materia. Se publica con fines orientativos, pero está sujeto a todo tipo de cambios. Los colores corresponden a teórica (T), laboratorio (L) y práctica (P). Las clases están numeradas: por ejemplo el lunes 13/04 está programado que se publiquen las primeras dos clases teóricas (T01 y T02).

1. **Semana 1**
   - Lun 13/04 — T01, T02: Especificación
   - Mié 15/04 — L01: Uso de clases
   - Vie 17/04 — Consultas
2. **Semana 2**
   - Lun 20/04 — P01: TADs y recursión (básico) ..... **Publicación de la guía 1**
   - Mié 22/04 — L02: Clases en C++ (básico), testing
   - Vie 24/04 — P02: TADs y recursión (avanzado)
3. **Semana 3**
   - Lun 27/04 — P03: TAD en dos niveles, T03: Complejidad
   - Mié 29/04 — L03: Clases en C++
   - Vie 01/05 — **Feriado: Día del Trabajador**
4. **Semana 4**

- Lun 04/05 — T04: Diseño, T03: Consultas 11:00 y 17:00 hs., Consultas 12:00 y 18:00 hs.
- Mié 06/05 — L04: Memoria dinámica
- Vie 08/05 — T04: Consultas 11:00 y 17:00 hs.
- Dom 10/05 ..... **Entrega de la guía 1**

5. **Semana 5**
   - Lun 11/05 — P04: Notación "O", complejidad ..... **Publicación de la guía 2**
   - Mié 13/05 — L05: Listas enlazadas
   - Vie 15/05 — T05: Diccionarios sobre ABBs y AVLs
6. **Semana 6**
   - Lun 18/05 — P05: Invariante de representación y función de abstracción
   - Mié 20/05 — L06: Templates y algoritmos genéricos
   - Vie 22/05 — P06: Repaso
7. **Semana 7**
   - Lun 25/05 — **Feriado: Día de la Revolución de Mayo**
   - Mié 27/05 — Consultas
   - Vie 29/05 — T06: AVL 2, Tries
   - Dom 31/05 ..... **Entrega de la guía 2**
8. **Semana 8**
   - Lun 01/06 — T07: Hashing 1 y 2 ..... **Publicación de la guía 3**
   - Mié 03/06 — L07: ABBs
   - Vie 05/06 — P07: Interfaces, iteradores, elección de estructuras
9. **Semana 9**
   - Lun 08/06 — T08: Colas de prioridad
   - Mié 10/06 — Consultas
   - Vie 12/06 — P08: Elección de estructuras 2
10. **Semana 10**
    - Lun 15/06 — **Feriado: Paso a la Inmortalidad del General Martín Miguel de Güemes**
    - Mié 10/06 — L08: Tries
    - Vie 12/06 — P09: Elección de estructuras avanzadas
    - Dom 14/06 ..... **Entrega de la guía 3**
11. **Semana 11**
    - Lun 22/06 — T09: Sorting básico ..... **Publicación de la guía 4**
    - Mié 24/06 — L09: Heaps
    - Vie 26/06 — P10: Sorting
12. **Semana 12**
    - Lun 29/06 — T10: Divide and conquer
    - Mié 01/07 — Consultas
    - Vie 03/07 — P11: Divide and conquer, recurrencias y teorema maestro
13. **Semana 13**
    - Lun 06/07 — P12: D&C avanzado

- Mié 08/07 — L10: Desarrollo de iteradores
- Vie 10/07 — **Feriado puente**
- Dom 12/07 ..... **Entrega de la guía 4**

14. **Semana 14**
    - Lun 13/07 — T11: Sorting y diccionarios en memoria externa
    - Mié 15/07 — Consultas
    - Vie 17/07 — Consultas
15. **Semana 15**
    - Lun 20/07 — T12: Splay trees y skip lists — Consultas ..... **Presentación del recuperatorio**
    - Mié 22/07 — L12: Sorting (ex taller de sorting)[2]
    - Vie 24/07 — Consultas
16. **Semana 16**
    - Lun 27/07 — Consultas ..... **Entrega del recuperatorio**
    - Mié 29/07 — Consultas

[2] Esta actividad solamente está programada de manera presencial, en caso de que se levante la cuarentena y la Facultad vuelva a su funcionamiento normal.

2. Ejercicios seleccionados

Los **ejercicios seleccionados** son un conjunto de ejercicios *mínimos* destinados a que cada estudiante pueda realizar una autoevaluación sobre su progreso en el dominio de los contenidos que se presentan en la materia, tanto desde el aspecto conceptual (entendimiento de los temas) como en el aspecto procedimental (capacidad de aplicar los conocimientos para resolver problemas prácticos). Deberían servir como disparadores para repasar partes de las clases que no hayan quedado claras, referirse a bibliografía complementaria y formular consultas.

Corrección: se brindarán resoluciones de los ejercicios prácticos, y se comentará sobre estas resoluciones en las clases de consulta. La idea es que no acudan a la resolución brindada por la cátedra sin haber intentado resolverlos por su cuenta.

Alentamos que los piensen individualmente, los discutan con sus compañeres y los consulten con les docentes. **Advertencia:** la resolución de los ejercicios seleccionados en esta guía no sustituye la resolución de prácticas (guías de ejercicios) publicadas en el sitio de la materia en el Campus.

2.1. Quiz sobre las clases teóricas

2.1.1. Clase T01: Introducción a TADs

- ¿Por qué especificamos con TADs?
- ¿Qué partes componen un TAD?
- ¿En qué se diferencian los TADs Secuencia y Arreglo Dimensionable? Para contestar esta pregunta sugerimos descargar el documento de TADs básicos.
- ¿Cuáles son los observadores del TAD Multiconjunto? ¿Se te ocurre algún contexto de uso para este TAD?

2.1.2. Clase T02: Especificación

- ¿Cuál es la diferencia entre un observador básico y otra operación?
- ¿A qué nos referimos con *romper congruencia*?
- Enumere 3 consideraciones a tener en cuenta para elegir generadores y observadores.
- ¿Por qué cree que el TAD *diccionario* no necesita un generador *borrar*?

2.2. Axiomatización con funciones recursivas

Antes de comenzar especificar utilizando TADs es importante practicar un poco de axiomatización. La axiomatización indica que tiene que devolver un observador u otra operación a partir de instancias concretas de nuestros TADs.

Ejercicio 1: se pide extender el tipo Secuencia(α) definiendo nuevas operaciones. Entre ellas, *Duplicar*, que dada una secuencia la devuelve duplicada elemento a elemento. Por ejemplo:

$$\text{Duplicar}(a \bullet b \bullet c \bullet <>) \equiv a \bullet a \bullet b \bullet b \bullet c \bullet c \bullet <>$$

¿Por dónde empezamos? Un primer paso podría ser escribir escribir la operación que queremos definir junto a cada generador.

Duplicar(<>) ≡ ...
Duplicar(e • s) ≡ ...

Ahora bien, ¿qué tendría que devolver *duplicar* de una secuencia vacía? En principio una secuencia vacía. ¿Y qué tendría que devolver *duplicar* de una secuencia que contiene el elemento *e* por delante? Una secuencia que tenga *e* duplicado y además los demás elementos duplicados. Esto se puede especificar mediante un axioma recursivo que recorra toda la secuencia logrando como resultado final el comportamiento deseado.

Duplicar(<>) ≡ <>
Duplicar(e • s) ≡ e • e • Duplicar(s)

¿Te animás a hacer los siguientes?

- *Reverso*, que dada una secuencia devuelve su reverso (la secuencia dada vuelta).
- *EsPrefijo?*, que chequea si una secuencia es prefijo de otra.
- *Buscar*, que busca una secuencia dentro de otra. Si la secuencia buscada está una o más veces, la función devuelve la posición de la primera aparición; si no está, la función se indefine.
- *EstáOrdenada?*, que verifica si una secuencia está ordenada de menor a mayor.
- *InsertarOrdenada*, que dados una secuencia *so* (que debe estar ordenada) y un elemento e (de género α) inserta e en *so* de manera ordenada.
- *CantidadApariciones*, que dados una secuencia y un α devuelve la cantidad de apariciones del elemento en la secuencia.
- *EsPermutación?*, que chequea si dos secuencias dadas son permutación una de otra. Pista: utilizar *CantidadApariciones*.

Ejercicio 2: Ejercicios con árboles binarios.
Definir las siguientes funciones sobre árboles binarios:

- *#Hojas*, que cuenta la cantidad de hojas del árbol.
- *DegeneradoAIzquierda*, que chequea si todo nodo interno (no hoja) tiene sólo subárbol izquierdo.
- *ZigZag*, que chequea si todo el árbol es degenerado con direcciones alternadas.
- *ÚltimoNivelCompleto*, que devuelve el número del último nivel que está completo (es decir, aquél que tiene todos los nodos posibles).

Ejercicio 3: A continuación se presenta el tipo ROSETREE:

TAD ROSETREE(α)

géneros rosetree(α)

igualdad observacional

$$(\forall r_1, r_2 : \text{rosetree})\ (r_1 =_{\text{obs}} r_2 \iff (\text{raíz}(r_1) =_{\text{obs}} \text{raíz}(r_2) \wedge \text{hijos}(r_1) =_{\text{obs}} \text{hijos}(r_2)))$$

observadores básicos

raíz : rosetree(α) $\longrightarrow$ α
hijos : rosetree(α) $\longrightarrow$ secu(rosetree(α))

generadores

rose : $\alpha \times$ secu(rosetree(α)) $\longrightarrow$ rosetree(α)

axiomas $\forall s$: secu(rosetree(α)) $\forall a$: α

raíz(rose(a, s)) $\equiv$ a
hijos(rose(a, s)) $\equiv$ s

Fin TAD

Ejemplos de instancias:
rose(1, <>), rose(true, rose(false, <>) • <>), rose(true, rose(false, <>) • rose(true, rose(true, <>) • <>) • rose(false, <>) • <>)

Defina operaciones auxiliares para:

- Conocer la altura de un rosetree.

- Calcular la cantidad de hojas de un rosetree.
- Podar un rosetree, es decir, eliminar todas las hojas del rosetree.
- Obtener todas las ramas de un rosetree cuya longitud sea menor o igual a un valor n dado. Una rama es una secuencia de nodos del árbol que va desde la raíz hasta una de sus hojas.
- Dado un número natural n, devolver una secuencia con los elementos del nivel n enumerados de izquierda a derecha. Los elementos de nivel n son aquellos nodos que se encuentran a distancia n de la raíz del rosetree.
- Calcular el conjunto de las ramas más largas de un rosetree que contienen al menos un elemento repetido.

2.3. Especificación (modelado)

Ejercicio 1: Con el objetivo de mejorar su servicio, un banco decidió analizar el comportamiento de sus largas filas de clientes. Se encargó la tarea al señor Felipe N. Sativo, empleado de la casa central del banco en cuestión, quien inmediatamente reconoció la necesidad de pedir nuestro asesoramiento.
Proponer una solución para cada ítem del ejercicio, discutirlo con sus compañeros y luego consultarlo con un docente.
El señor Sativo nos presentó un informe en el que detallaba el comportamiento esperado de una fila de clientes. Después de leerlo, releerlo y extraer de allí la información importante, comprendimos que las acciones que esperaba registrar eran la apertura de la ventanilla, la llegada de un nuevo cliente a la fila, y la atención del cliente que estuviera en primer lugar (con su consecuente egreso de la fila). También quedó claro que el banco deseaba poder verificar si la fila estaba vacía, conocer la longitud (cantidad de clientes) de la fila, si un cliente determinado estaba o no esperando su atención en dicha fila y, en caso de estarlo, cuál era su posición en ella (siendo la posición 1 la del próximo cliente que sería atendido, es decir, el que haya llegado primero entre los clientes presentes).
Especificar (distinguiendo generadores, observadores básicos y otras operaciones, y escribiendo los axiomas) el tipo FILA, cuyas funciones a exportar son las siguientes:

AbrirVentanilla	: $\longrightarrow$ fila	
Llegar	: persona p $\times$ fila f $\longrightarrow$ fila	$\{\neg\text{Esperando}(p, f)\}$
Atender	: fila f $\longrightarrow$ fila	$\{\neg\text{Vacía}(f)\}$
Esperando	: persona $\times$ fila $\longrightarrow$ bool	
Vacía	: fila $\longrightarrow$ bool	
Posición	: persona p $\times$ fila f $\longrightarrow$ nat	$\{\text{Esperando}(p, f)\}$
Longitud	: fila $\longrightarrow$ nat	

El género es "fila". Se permite agregar funciones auxiliares en caso de ser necesario.

Durante el primer día de observación, Felipe N. Sativo descubrió que la realidad de las filas del banco era más compleja de lo que él había previsto. Notó que algunos clientes, desalentados por la longitud y lentitud de la fila, se retiraban sin haber sido atendidos. También detectó algunos clientes que no respetaban el orden de la fila, introduciéndose en ella delante de otros clientes que habían llegado antes (en términos más simples: colándose). Felipe decidió que era importante registrar estos sucesos, así como también conocer la identidad de los irrespetuosos alteradores del orden.
Modificar la especificación anterior para incluir las siguientes funciones:

Retirarse	: persona p $\times$ fila f $\longrightarrow$ fila	$\{\text{Esperando}(p, f)\}$
ColarseAdelanteDe	: persona p $\times$ persona q $\times$ fila f $\longrightarrow$ fila	$\{\neg\text{Esperando}(p, f) \wedge \text{Esperando}(q, f)\}$
SeColó?	: persona p $\times$ fila f $\longrightarrow$ bool	$\{\text{Esperando}(p, f)\}$

El banco felicitó al Sr. Sativo por su gran desempeño, y le entregó un último encargo, necesario para comprobar la verdadera eficiencia del banco con respecto a la atención de sus clientes. El encargo consistía en averiguar si un cliente dado ingresó alguna vez a la fila (ya se encuentre esperando en ella, haya sido atendido o se haya retirado), y si una persona determinada había sido atendida durante el día (nótese que el tipo FILA sólo registra la información de un día, desde el momento en que se abre la ventanilla).

Modificar nuevamente la especificación para agregar las funciones faltantes:

Entro?	:	persona × fila	⟶	bool
FueAtendido?	:	persona × fila	⟶	bool

Notar que la nueva especificación puede no cumplir los puntos anteriores.

Ejercicio 2: Técnicos a domicilio

Técnicos a Domicilio (o simplemente "TaD"), es una empresa que provee servicio técnico para problemas de electricidad en hogares y empresas. TaD cuenta con un grupo de técnicos altamente capacitados para atender la demanda de sus clientes y tiene una estrategia de trabajo algo particular. Cuando alguien solicita un técnico, la central de TaD verifica si alguno de sus técnicos se encuentra en la empresa y de ser así envía inmediatamente un técnico al domicilio de la persona. En caso de no haber técnicos disponibles en ese momento (i.e., todos se encuentran atendiendo algún pedido), el pedido queda *pendiente de asignación* a la espera de que algún técnico se desocupe.
Por otro lado, cuando un técnico termina de resolver un problema, y antes de retirarse de ese domicilio, el técnico avisa por radio a la central que quedó disponible para otro trabajo. Si existiesen en ese momento pedidos *pendientes de asignación*, la central le asigna al técnico el más cercano al domicilio en el que éste se encuentra y el técnico se dirige automáticamente hacia allí (si hay más de un pendiente a la misma distancia mínima, se asignará al pedido entre éstos que lleve más tiempo esperando). Por el contrario, de no haber trabajos pendientes, el técnico regresa a la central y queda disponible para futuros trabajos.
Modelar con un TAD la empresa *Técnicos a Domicilio* descripta teniendo en cuenta además que interesa saber, dada una dirección, quiénes fueron los técnicos que la visitaron la mayor cantidad de veces (aun si todavía no resolvieron el inconveniente técnico).

Observación: Se puede asumir como dado el TAD Dirección que exporta el género *dirección* y la operación *dist(d, d′)* que devuelve un *nat* que representa la distancia entre las direcciones d y d'.

3. Ejercicios obligatorios de la práctica

IMPORTANTE: Los ejercicios de esta sección son de carácter individual y serán calificados. Cada ejercicio podrá estar aprobado, o será devuelto para incorporar correcciones. Las consultas sobre estos ejercicios deben limitarse a expresar dudas de interpretación sobre el enunciado. Además, como el objetivo de estos ejercicios es evaluar el desempeño individual, no se permite trabajar grupalmente ni compartir soluciones. La fecha límite de entrega es la fecha de finalización del bloque.

3.1. Especificación utilizando TADs

Ejercicio 1: Axiomatización

Consideremos el TAD Rosetree presentado en la guía de ejercicios.

TAD Rosetree(α)

géneros rosetree(α)

igualdad observacional

$(\forall r_1, r_2 : \text{rosetree})\ (r_1 =_{\text{obs}} r_2 \iff (\text{raíz}(r_1) =_{\text{obs}} \text{raíz}(r_2) \wedge \text{hijos}(r_1) =_{\text{obs}} \text{hijos}(r_2)))$

observadores básicos

raíz : rosetree(α) $\longrightarrow$ α

hijos : rosetree(α) $\longrightarrow$ secu(rosetree(α))

generadores

rose : $\alpha \times$ secu(rosetree(α)) $\longrightarrow$ rosetree(α)

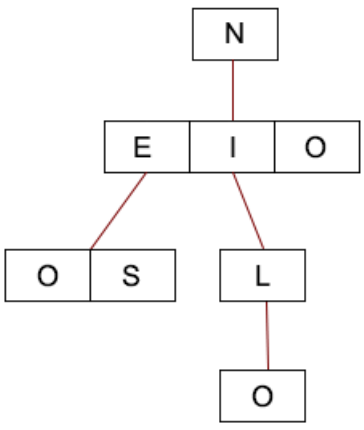
axiomas $\forall s$: secu(rosetree(α)) $\forall a$: α

raíz(rose(a, s)) $\equiv$ a

hijos(rose(a, s)) $\equiv$ s

Fin TAD

Se pide extender el tipo de datos agregando una función que dada una instancia de un rosetree de caracteres, y una secuencia de palabras, devuelva las palabras de la secuencia que se puede formar recorriendo el árbol desde la raíz hasta una hoja. Por ejemplo, tomando la secuencia `["Nilo", "No", "Si"]` y el siguiente rosetree, el resultado debería ser `["Nilo", "No"]`.



Ejercicio 2: Sistema de Administración de Repatriados

El estado argentino está repatriando gente del exterior en vuelos que provienen de diversas partes del mundo. Para poder recibir a la gente, se dispusieron unos centros de hospedaje que sirven para albergar a los pasajeros que vayan llegando con el fin de que hagan la cuarentena allí. Cuando aterriza un vuelo, se le asigna alguno de los centros para que las personas a bordo se alojen allí durante 15 días. El algoritmo para seleccionar el centro se definirá en la etapa de diseño. Una vez transcurrido el período de cuarentena los pasajeros pueden irse a sus casas. Por último, en cualquier momento, las autoridades pueden decidir mover una persona de un centro a otro.

En el relevamiento nos informaron que se desea saber en todo momento qué personas se alojan en cada centro y cuántas personas fueron dadas de alta. Podemos asumir que cada persona se identifica con un nombre único.

Especificar con un TAD el Sistema de Administración de Repatriados (SAR) descripto, teniendo en cuenta que además se desea saber por cuál centro hubo más tránsito, es decir, si una persona pasa dos veces por el mismo centro se cuenta doble (si hay más de un centro para elegir, devolver alguno sin una preferencia establecida). La especificación debe estar completa, incluyendo tanto el modelo (observadores, generadores y otras operaciones con sus restricciones) como la axiomatización.

4. Ejercicios obligatorios del laboratorio

IMPORTANTE: Los ejercicios de esta sección se dividen en TPs (grupales, grupos de 4 integrantes) y talleres (individuales), y serán calificados. Cada instancia podrá estar aprobada, o será devuelta para incorporar correcciones. Para la resolución de los TPs se espera que trabajen grupalmente y consulten todas sus dudas. No está permitido compartir soluciones detalladas entre grupos de trabajo distintos. En el caso de los talleres, si bien se permite que discutan ideas grupalmente, cada entrega debe ser individual y todo el código entregado debe ser de producción propia. La fecha límite de entrega es la fecha de finalización del bloque.

4.1. TP 1.1

Este trabajo práctico plantea modelar una lógica simplificada del juego SimCity de 1989.[3]
El SimCity es un juego de simulador de la vida y crecimiento de las ciudades. En el recorte para el TP, solo vamos a manejar el crecimiento de casas y comercios.

- En nuestra versión del juego, las partidas comienzan con un mapa ya definido. Un mapa puede considerarse una grilla infinita en cual corren ríos. Los ríos corren en líneas infinitas horizontales o verticales. En la partida, en cada turno pueden agregarse casas o comercios. Cada uno ocupa una única casilla, que no puede estar en el mismo lugar que un río ni encima de otra construcción.
- Los comercios y casas además tienen un nivel de avance. En las casas, el nivel se define como la cantidad de turnos que sucedieron desde que fue agregada. En los comercios, el nivel se define de igual manera, con la excepción de que si una o más casas a 3 casilleros de distancia manhattan[4] tiene un nivel más alto, el comercio adopta el máximo de ellos.
- El juego se sucede por turnos, donde en cada uno se debe agregar al menos una construcción, pudiéndose agregar más de una en el mismo turno.

Entrega

La entrega consistirá de un único documento digital con el modelado en TADs del enunciado presentado. Se recomienda el uso de los paquetes de LaTeX de la cátedra para lograr una mejor visualización del informe.
El medio de entrega será informado más adelante.

[3] `https://www.youtube.com/watch?v=A54blk-ojA4`.
[4] $manhattan(x_1, y_1, x_2, y_2) = |x_2 - x_1| + |y_2 - y_1|$

4.2. Taller: Clases

En este taller se debe implementar una agenda. El enunciado descripto a continuación replica el Ejercicio 14 de la guía de ejercicios de laboratorio de la clase de definición de clases. Esta guía contiene sugerencias para la implementación de las partes que conforman la implementación a entregar.
La agenda debe iniciarse con la fecha del día actual. La fecha es referente al año corriente, por lo que debe definir día y mes. En una agenda deben poder agregarse recordatorios. Los recordatorios son asignados a un cierto horario en una fecha del año y contienen un mensaje describiendo que ha de recordarse. Una agenda debe poder presentar los recordatorios para el día actual. Finalmente, se debe poder incrementar el día en el sistema. Se debe cumplir la siguiente interfaz:

```
class Agenda {
  public:
    Agenda(Fecha fecha_inicial);
    void agregar_recordatorio(Recordatorio rec);
    void incrementar_dia();
    list<Recordatorio> recordatorios_de_hoy();
    Fecha hoy();
};
```

Para conocer los recordatorios del día se espera que se sobreescriba el operador << y se impriman los recordatorios del día con el siguiente formato:

```
Agenda a(Fecha(5, 10));
a.agregar_recordatorio(Recordatorio(Fecha(5, 10), Horario(9, 0), "Clase Algo2"));
a.agregar_recordatorio(Recordatorio(Fecha(5, 10), Horario(11, 0), "Labo Algo2"));
cout << a << endl;
// 10/5
// =====
// Clase Algo2 @ 10/5 9:0
// Labo Algo2 @ 10/5 11:0
```

Notar que se imprime primero la fecha actual, luego un separador compuesto por el símbolo igual repetido cinco veces (“=====”) y finalmente un recordatorio por línea. Los recordatorios del día deben imprimirse ordenados en orden cronológico (creciente por horario).

Entrega

La entrega deberá consistir de un único archivo comprimido (`.zip`) con los archivos fuente (`.h`, `.cpp`) necesarios para la agenda.
El medio de entrega será informado más adelante.