PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 Programación Avanzada (II/2017)

Tarea 4

1. Entrega

- `Entregable`
  - **Fecha/hora**: 11 de octubre del 2017, 23:59 horas.
  - **Lugar:** Cuestionario en el Siding
- `Tarea`
  - **Fecha/hora:** 22 de octubre del 2017, 23:59 horas.
  - **Lugar:** GitHub – Carpeta: `Tareas/T04/`
- `README.md`
  - **Fecha/hora:** 23 de octubre del 2017, 23:59 horas.
  - **Lugar:** GitHub – `Carpeta: Tareas/T04/`

2. Objetivos

- Aplicar conocimientos de simulación por eventos discretos
- OOP:
  - Uso de *properties*
  - *Overriding* de algunos métodos mágicos
  - Herencia
  - Métodos de clases
  - Atributos de clases
- Documentación de código

3. Introducción

Han pasado ~~siglos~~ días desde que lograste traducir el lenguaje genoma-humánico recopilado por ainav-I y fer_and.ino, logrando recuperar toda la raza humana. Poco a poco, vuelves a la rutina de viajar a San Joaquín que habías dejado para traducir el genoma, pero hay algo distinto en la PUC. El ~~benevolente~~ magnate *Devil Sawak*, enfurecido por notar que las *nebcoins* no estaban siendo aceptadas como medio de pago en los locales

de comida dentro del campus, decidió adueñarse de todos ellos. Gracias a su ~~mafia~~ amplio conocimiento y habilidad en el mercado de las *criptomonedas*, logró instaurar un monopolio bajo el nombre de *Quik Devil*. Dado que dentro del campus no existe competencia alguna, la fijación de precios en todos los productos es excesivamente alta, muy lejos del alcance de la mayoría de los alumnos y funcionarios. Ante el problema inminente de la hambruna de los miembros de la PUC, empezaron a aparecer puestos de vendedores ~~legales~~ afuera del campus quienes, a diferencia de *Devil*, ofrecen productos con precios asequibles para alumnos y funcionarios.

Tú, como alumno, estás en contra del monopolio impuesto por *Devil* y eres un fiel partidario de los puestos ambulantes. Entonces, quieres ser capaz de minimizar las ganancias de *Quik Devil*, pero no sabes cómo hacerlo en la vida real. De repente, te acuerdas que en el curso de *Programación Avanzada* viste algo que se llamaba **DES**[1] por lo que decides aplicarlo a esta situación. En particular, quieres averiguar cosas como: el número de personas que almuerza a una determinada hora, cantidad de alimentos promedio vendido por día, cantidad promedio de vendedores que se queda sin *stock* en un día, entre otros. La información obtenida de la simulación puede ayudar a los vendedores a que se organicen y tomen decisiones como las siguientes: cuántos vendedores trabajarán por día, cuánto *stock* deben tener en un día normal, entre otras decisiones.

4. Descripción del problema

Para replicar lo que está pasando en *San Joaquín*, decides simular el horario de almuerzo[2] durante **un semestre académico**[3] contabilizando sólo los días hábiles. Para ello, debes modelar el problema incluyendo las distintas entidades que interactúan en este entorno. Existen cuatro entidades principales: alumnos, funcionarios (ambos conocidos como **MiembrosUC**), vendedores y carabineros (los dos conocidos como **Externos**), quienes cumplen roles distintos. Los **MiembrosUC** tienen el rol de comprar comida, los vendedores tienen el rol de vender comida y los carabineros tienen el rol de fiscalizar a las los vendedores.

Tanto **Alumnos** como **Funcionarios** tienen la posibilidad de elegir dónde almorzar, ya sea en alguno de los puestos de sus vendedores favoritos o en el ~~querido~~ *Quik Devil*. Para los **Alumnos**, la elección dependerá de varios factores: preferencia inicial, cuánto dinero tienen disponible, calidad de la comida, cuanto tiempo tiene que esperar en la cola de un puesto, etc. En cambio, los **Funcionarios** son más simples: la elección la realizan al azar dentro de sus preferencias iniciales.

Por otra parte, los **Vendedores** pueden ofrecer almuerzos o *snacks*. Cada **Vendedor** tiene un puesto asignado dentro de la feria, pueden atender a una persona a la vez y cada uno tiene una velocidad de atención. El precio y calidad de los productos son un factor importante en las preferencias de los alumnos. El precio está determinado por diversos factores: quedarse sin *stock* o pocas ventas, mientras que la calidad de los productos se determina según el nivel de putrefacción, precio y calorías.

Para que la simulación sea más *real*, tienes que incluir ciertos eventos que pueden ser tanto programados como no programados. Dentro de los eventos programados, están los eventos de llegar al campus, almorzar, ir a comprar comida, instalar puestos, entre otros. Por otra parte, en los eventos no programados se encuentran las temperaturas extremas, llegada de policías, concha estéreo, etcétera. Todos estos eventos, influirán de alguna forma en el evento principal del almuerzo ya sea porque clausuran puestos, la comida se descompone, etcétera.

Además de incluir los eventos, debes utilizar datos reales recopilados por tus queridos amigos Aravena y Mr. Olea de muchos estudiantes que ~~por coincidencia~~ están cursando *Programación Avanzada*. Los datos recopilados incluyen: la cantidad de dinero de los estudiantes y funcionarios, datos estadísticos (como la moda de llegada al campus, tiempo de traslado al campus, etcétera), velocidad de atención de los vendedores,

[1] *Discrete Event Simulation*
[2] 11am - 3pm
[3] 4 meses

la distribución de los horarios de almuerzo, entre otras cosas. Estos datos estarán guardados en un archivo llamado `parametros_iniciales.csv`.

Finalmente, el gran objetivo de simular esta realidad es poder visualizar mediante estadísticas y comparación de escenarios qué eventos influyen en los resultados. De esta manera, podrás realizar un estudio exhaustivo para lograr que —eventualmente— *Quik Devil* quiebre.

5. Entidades [35 %]

En la simulación, es posible identificar 3 tipos de entidades distintas: las **Personas**, **Productos** y ***Quik Devil***.

5.1. Personas

Todas las personas tienen un nombre, apellido, edad y género.

5.1.1. Miembros UC

Los **MiembrosUC** se diferencian según el *saldo disponible* para gastar y las *prioridades* que tienen al momento de comprar comida. Dentro de los **MiembrosUC** existen los **Alumnos** y los **Funcionarios**.

- `Alumnos`:
  - Disponen una cantidad de pesos diarios (no acumulables) para gastar en comida dadas por una *mesada*[4]. En otras palabras, disponen de $\lfloor$ `mesada_del_mes / 20` $\rfloor$ pesos diarios no acumulables.
  - Al comprar comida en un puesto ingresan a una cola para esperar a que los atiendan.
  - Tienen una lista de preferencias ordenadas según el puesto más preferido al menos preferido para definir a qué puesto ir a comprar. Existe la posibilidad de que la comida de un puesto haya estado mala y los enferme; si eso llega a ocurrir, ese puesto desaparece de su lista de preferencias. Las características, para que una persona se enferme por la comida, serán especificadas en la sección de *Productos*.
  - Existen tres formas de modificar la cola de preferencias: el puesto se quedó sin *stock*, el alumno no tiene dinero suficiente para comprar, o que el tiempo de atención del puesto haya superado el límite de paciencia del alumno. Si se cumple al menos uno de los puntos descritos anteriormente, el alumno sale de la cola y se va al siguiente puesto dentro de su lista de preferencias. En caso de no comprar en ninguno de los puestos de su lista de preferencias, y si tienen el saldo suficiente, se resignan y van al *Quik Devil* y sino, ese día no almuerzan. A continuación, se describirán las formas de modificar la cola de preferencias:
    - **El puesto se quedó sin *stock***. En este caso, si en la cola del puesto hay "X" personas, el *stock* debe ser al menos "X + 1" para que el alumno decida ir a la cola.
    - **Dinero insuficiente**. Si el puesto solo tiene productos cuyo precio supera el dinero que tiene el alumno, este debe elegir el siguiente vendedor de su lista.
    - **Límite de paciencia**. En este caso, existe un factor que representa la cantidad de minutos que un alumno puede esperar en una cola sin aburrirse. El límite de paciencia parte siendo cada día un `randint` entre $\alpha_{paciencia}$ y $\beta_{paciencia}$ minutos; y por cada puesto que rechaza el alumno, el factor se reduce en 5 minutos, hasta llegar a 0. Cuando el alumno decide comprar a un vendedor, debe estimar según la rapidez de atención del vendedor y la cantidad de personas en la cola, cuánto tiempo esperará. Si su límite de paciencia es igual o mayor al tiempo que se demorará el vendedor, entonces decide ir a ese puesto y unirse a la cola. En caso contrario, se decide ir al siguiente puesto.

[4] Ver sección de *Eventos programados*.

- `Funcionarios`:
  - Disponen de $dinero\_funcionarios$ pesos diarios no acumulables para gastar.
  - Al llegar a un puesto, ellos tienen preferencia por sobre los **Alumnos**. Si un funcionario llega a un puesto y no hay más funcionarios en cola, él será atendido inmediatamente después de la persona que está siendo atendida en ese momento, independiente de la cantidad de alumnos que esté en cola. Si hay otros funcionarios en cola, el nuevo funcionario entrará a la cola ocupando el puesto inmediatamente posterior al último funcionario en cola. En consecuencia, cada vez que un funcionario llegue a un puesto, todos los alumnos en cola se atrasarán un tiempo equivalente a lo que se demore en ser atendido el funcionario.
  - Se les asegura el mejor producto. Esto significa que dentro de los productos que tiene el vendedor, a éste se le entrega el de mejor calidad disponible. La calidad será especificada en la sección de *Productos*.
  - Al igual que los alumnos, tienen una lista de puestos donde comprar. La elección de un puesto se hace de forma aleatoria, pero al siguiente día no pueden ir al mismo (les gusta variar). En caso de que la comida de un puesto enferme a un funcionario, el puesto desaparecerá de la lista. Por otra parte, si el puesto elegido se queda sin *stock*, el funcionario elige otro puesto (hasta un máximo de 3). En caso de que ninguno tenga *stock*, irá al *Quik Devil*.

5.1.2. Externos

Estas personas son aquellas externas a la universidad que hacen posible que la feria ambulante de SJ exista.

Dentro de esta categoría, podemos diferenciar dos tipos de personas: los **Vendedores** (gracias a ellos no morimos de hambre de lunes a viernes) y los **Carabineros**. Estos últimos están encargados de hacer caer (y cumplir) la ley sobre este creciente mercado.

- `Vendedores`
  - Cada vendedor tiene un puesto en la feria donde pueden vender **Almuerzos** de algún tipo (comida china o comida mexicana) o ***Snacks***.
  - Los vendedores son fieles a su estilo y cada uno venderá siempre el mismo tipo de comida. Cada vendedor vende todos los productos existentes del tipo de comida en el que se especializa.
  - Cada vendedor tiene una cola en la que los **Miembros UC** esperan por su turno siguiendo las reglas explicadas en la sección anterior.
  - Cada vendedor tiene una velocidad de atención en minutos de `randint(`$\alpha_{rapidez}$`,` $\beta_{rapidez}$`)`. Esta velocidad se define una sola vez al inicio de la simulación para cada vendedor.
  - Cada vendedor tiene una cantidad de productos disponibles para vender en un día. Esta cantidad le llaman *stock*. El *stock* se define diariamente por una *distribución uniforme discreta* de parámetros α_{stock} y β_{stock}. Una vez que el *stock* se agota, los alumnos que están en la cola de dicho puesto se van a su siguiente alternativa. Si al final del horario de almuerzo queda *stock* disponible, este se pierde. Como simplificación, en la simulación no se maneja un stock independiente por cada producto, sino un stock general para cada vendedor. En otras palabras, o bien el vendedor tiene todos sus productos disponibles, o bien no tiene ninguno.
  - Una parte esencial en los negocios son los precios; Inicialmente, contarás con los precios iniciales universales para cada producto en el archivo `productos.csv`, pero a lo largo de la simulación, los vendedores cambiarán sus precios cada inicio de mes. Cuando suceda, deberás informar a través de la consola los precios que cambiaron y en que porcentaje. El sistema de cambio de precios seguirá el siguiente criterio:

- Si un vendedor se quedó sin *stock* N veces en un mes, entonces el próximo mes subirá el precio de todos los productos que vende en un $6N\,\%$. Por ejemplo, *Si el precio inicial era* P*, el siguiente mes será* $P + P * \frac{6N}{100}$.
- Si un vendedor no vendió ninguna unidad del producto N veces en un mes, entonces el próximo mes se reducirá en $5N\,\%$. Por ejemplo, si el precio inicial era P, el mes siguiente será $P - P * \frac{5N}{100}$. El precio mínimo al que se puede vender un producto es el $1\,\%$ del precio a inicio de la simulación.
- En caso que no suceda ninguna de estas opciones, el precio del producto se mantendrá igual.

Estas opciones pueden ocurrir en un mismo mes. En este caso el precio nuevo será $P - P * \frac{5N}{100} + P * \frac{6N}{100}$

- Si un vendedor lleva a lo largo de la simulación 20 días seguidos sin ventas, entonces se declarará en bancarrota y no volverá a vender más productos.
- Debido a temas legales, cada vendedor debería tener un permiso de venta. Es por esto que ante la llegada de **Carabineros**, los vendedores tendrán que mostrar el permiso para que su puesto no sea clausurado. La probabilidad de que un vendedor tenga un permiso es $p_{permiso}$ y debe quedar definido al inicio de la simulación. Las consecuencias de la atención de clientes por culpa de **Carabineros** se define en el siguiente ítem.

▪ `Carabineros`

- Cuando esta entidad llega, el **caos** cunde en la feria... ellos revisarán ciertos puestos de vendedores y los que no tengan el permiso serán clausurados y se les confiscará el *stock*. Los vendedores *ilegales* quedarán tan asustados que volverán a la feria después de *dias_susto* días. En cambio, los puestos que sí cuenten con el permiso, se les inspeccionarán los productos (la inspección se detalla en los siguientes párrafos). Si al menos un producto fue encontrado en mal estado, entonces toda la comida del vendedor será confiscada.
- Cada inspección que realiza **Carabineros** está dada por un solo agente, el cual puede tener dos personalidades con igual probabilidad: *Dr. Jekyll* y *Mr. Hyde* (definidas en el cuadro 1). Él/ella fiscalizará todos los puestos entre las 13:00 hasta las 13:40. Un **Carabinero** decide aleatoriamente el puesto a fiscalizar. Cuando llega a revisar un puesto, el vendedor detiene sus ventas hasta que la revisión haya terminado. Si estaba atendiendo a alguien, éste vuelve al primer lugar de la cola para que al finalizar la revisión. Si todo sale bien, esa persona es la primera en ser atendida. La revisión de cada puesto demorará $\frac{40}{N}$ minutos donde N es el número total de vendedores.
- Cuando un **Carabinero** llega a revisar un puesto, pide primero el permiso. En caso de que se descubra que no tenga el permiso, el vendedor escapa instantáneamente, ausentándose a la feria *dias_susto* días. Existe la posibilidad de que el agente sea engañado y se convenza de que el puesto sí tiene permisos. En caso de que sea engañado o el vendedor sí tenga el permiso, se deja de atender a los clientes y se procede a revisar la comida. Si a lo menos un producto es encontrado en mal estado, entonces se confisca todo el *stock* de comida de este vendedor. Si se confisca la comida, el vendedor se ve obligado a irse de la feria y volver al día siguiente de manera normal[5].

- **Tasa_ de_productos_a_revisar**: es el porcentaje total de productos que revisará en busca de comida en mal estado.
- **Prob_engaño**: es la probabilidad que tienen de ser engañados por parte de los vendedores a la hora de pedir el permiso.

[5]No se confiscan los permisos

Personalidad	Tasa productos a revisar	Prob engaño
Dr. Jekyll	$Tasa\_productos\_jekyll$	$Probabilidad\_engaño\_jekyll$
Mr. Hyde	$Tasa\_productos\_hyde$	$Probabilidad\_engaño\_hyde$

Cuadro 1: Tabla personalidades Carabineros (Los valores de las variables se obtienen de `parametros_iniciales.csv`)

5.2. *Quik Devil*

Empresa fundada por el millonario que encabeza el *ranking* ~~virtual~~ de la revista *Forbes*, *Devil Sawak*. *Quik Devil* es una revolucionaria empresa que vende productos baratos a precios caros. Misteriosamente, a pesar de ello, los alumnos están satisfechos con sus productos. Sumado a esto, su fundador le entregó un enfoque *neoliberal* a la empresa, es decir venden todo tipo de producto.

La principal diferencia de este local comercial con los vendedores es que contiene robots con un avanzado algoritmo, que les permiten vender de forma instantánea y sin que se produzcan filas. Si un estudiante decide comprar en este lugar, no tendrá tiempo de espera. Además, *Quik Devil* se encuentra asociada de forma directa con el banco *VVBA*, por lo que esta empresa no puede caer en banca rota y no le importa variar sus precios (los precios se mantienen constantes).

5.3. Productos

Los productos pueden ser clasificados como **Almuerzos** o **Snacks**. Ambos tipos de producto comparten los mismos atributos, pero se diferencian en la hora del día en que son comprados. A continuación, se explicarán los atributos que comparten los **Productos**:

- **Atributos**
  - **Putrefacción**: Corresponde a un valor entre 0 y 1 que representa el porcentaje de putrefacción y descomposición de los alimentos. Este se define a partir de la tasa de putrefacción según la formula:

    $$putrefaccion(t) = 1 - e^{-\frac{t}{\lambda}}$$

    Donde t es el tiempo en minutos transcurridas desde las 8:00 am del mismo día hasta que se compra el producto.
  - **Precio**: Es el precio en *DCCoins* de una unidad del producto.
  - **Calorías**: Corresponde al aporte *calórico* de una unidad del producto.
  - **Calidad**: La calidad de un producto se calcula según la fórmula

    $$calidad(t) = \frac{calorias*(1-putrefaccion(t))^4}{precio^{4/5}}$$

    En el Cuadro 2 se muestran algunos ejemplos de productos y sus características.
- **Snacks**: Los miembros de la universidad decidirán comprar un *snacks* en un día cualquiera con probabilidad 0,5. Si deciden comprar un *snack* el tiempo en el cual deciden comprarlo(compran solo un snack por día) se determina según una distribución $uniforme(hora\_llegada, 15:00)$. Los *snacks* se encuentran en la misma base de datos que el resto de los productos.(*considere que no se demoran tiempo en trasladarse al momento de comprar un snack*)

Si un producto tiene una calidad menor a 0.2 entonces este tiene un 35 % de enfermar a su consumidor.

Producto	Tipo	Precio	Calorías	Tasa_de_putrefacción	Vendido en
Carne mongoliana	Almuerzo	40 DCCoins	1000	3	Puesto de comida china
Tacos al pastor	Almuerzo	20 DCCoins	400	3	Puesto de comida mexicana
Chumbeque	*Snack*	15 DCCoins	150	1	Puesto de *snacks*

Cuadro 2: Ejemplo Tabla productos

6. Eventos programados [10 %]

Hay eventos o acciones que ocurren todos los días y solo cambian, quizás, los horarios o las cantidades en que ocurren. Algunos de ellos son:

- `MiembrosUC`
  - **Almorzar**: Todos los usuarios, tanto estudiantes como funcionarios, almuerzan todos los días. El X % de los alumnos y funcionarios almuerza de 13:00 a 13:59, el Y % almuerza de 14:00 a 15:00, mientras que el $(100 - X - Y)$ % lo hace de 12:00 a 12:59. Con respecto a los vendedores, en este mundo no comen. Los datos de X e Y se entregarán al inicio de la simulación.
  - **Llegar al campus**: Los miembros de la PUC llegan al campus desde las 11:00, siguiendo una distribución triangular con un mínimo de 0 minutos, máximo de 240 minutos y moda $c_{llegada}$ minutos, de esta manera la llegada será 11:00+$distribucion\_triangular(0, 240, c_{llegada})$, ya que todos los alumnos llegan en metro, entonces todos los alumnos pasan cerca de los puestos de comercio externos.
  - **Decidir ir a comer**: Para efectos de la simulación, todas las personas deciden ir a comer mas o menos a la misma hora, esto significa que la hora en la cual los miembros de la UC se dirigen a comer distribuye normal con parametros μ = X:10 horas con $\sigma = 10$ minutos, donde X puede ser 12, 13 o 14 dependiendo si la persona almuerza entre 12 y 13 horas, 13 y 14 horas, 14 y 15 horas respectivamente.(Considere que la distribución Normal no va tomar valores menores a X-1 hora y mayores a X+2 horas, es decir si almuerza entre 13 a 14, sólo tomará valores entre la 12:00 y 15:00 horas)
  - **Ir a comprar comida**: Ya que no todos los miembros de la PUC se ubican cerca de los puestos de comida, el tiempo T que representa el tiempo que demorará la persona en llegar distribuye *exponencial* de parámetro $\lambda_{traslado}$(considere que un alumno nunca se demorará más de $3\lambda_{traslado}$ y que el tiempo para cambiarse de cola es nulo)
- `Alumnos`:
  - **Mesada**: Ya que casi ningún alumno trabaja, la fuente de ingresos se reduce a una mesada mensual variable. En otras palabras, cada 30 días se recalcula la mesada para cada alumno utilizando la siguiente fórmula:

$$mesada = base\_mesada * (1 + \texttt{random()}^{\texttt{random()}}) * 20$$

- `Vendedores:`
  - **Instalar puestos** Todos los puestos abren alrededor de las 11 am, según un proceso de distribución *normal* de parámetros μ = 11:00, $\sigma = 30$ minutos.

7. Eventos no programados [10 %]

La vida del estudiante no es tan simple y siempre surgen situaciones inesperadas que hacen cambiar sus decisiones. A continuación, se detallarán aquellas situaciones que suceden de forma aleatoria y que pueden afectar al mercado y a los estudiantes:

- **Temperaturas extremas** *(el cambio climático tiene sus efectos éh)*: existe una probabilidad de que un día tenga temperaturas extremas, es decir, puede hacer demasiado calor o bien demasiado frío. Lo anterior terminará afectando la vida útil de los productos o su calidad. Las temperaturas extremas tienen una *distribución uniforme discreta* que va desde 2 hasta 20 días. Los días de temperatura calurosa o de frío tienen la misma probabilidad y tienen las siguientes repercusiones:
  - Frío intenso: La calidad de todos los productos disminuye a la mitad.
  - Calor intenso: La putrefacción de todos los productos aumenta al doble o hasta el máximo valor.

- **Concha ~~acústica~~ estéreo**: Los estudiantes organizarán una gran tocata musical en la universidad que atraerá a gente de otras casas de estudio. Esto motivará a todos los alumnos, inclusive a los más flojos. La probabilidad de que se realice este evento un día viernes es de p_{concha}. Sin embargo, si pasan más de cuatro semanas sin que se realice el evento, éste se llevará a cabo la quinta semana. La asistencia a la universidad en aquel día aumentará y, como los vendedores tienen estudios de *Microeconomía*, éstos subirán sus precios en ese día en un 25 % debido al aumento en la demanda.

- **Llegada de policías:** La empresa *Quik Devil* usualmente se siente perjudicada por la presencia de vendedores ambulantes, por lo que su dueño *Devil* llamará algún día anónimamente a los policías locales —sobornados en base a *nebcoins*— para que fiscalicen la situación que existe afuera de la universidad. Las llamadas de *Devil* distribuyen `Exponencial` con parámetro $\lambda t_{llamada}$, y una vez que se realiza la llamada, los policías llegarán inmediatamente.

- **Lluvia de hamburguesas:** Al más puro estilo *hollywoodense*, comenzará una lluvia de hamburguesas a lo largo de todo el campus que logrará satisfacer la hambruna de todos en un día y hacerlos felices a casi-todos (*lamentablemente a los comerciantes no*). Los estudiantes y funcionarios no consumirán productos a lo largo de todo el día y el día siguiente aumentarán su probabilidad de enfermarse frente a un producto de mala calidad al doble(de 35 % a 70 %), ya que el *smog* transforma las hamburguesas en hamburguejas al vapor, las cuales dejan al estomago muy sensible.

  La probabilidad de que en un día haya una lluvia de hamburguesas distribuye exponencial de parámetro $\lambda t = (\frac{1}{21-n})$ donde n son los días transcurridos desde la última temperatura extrema (*considere inicialmente $n = 0$*). Como lo enuncia *Pieressa* en la cuarta ley de la termodinámica *"Las tormentas de hamburguesas son sensibles al clima extremo, por ende no podrá haber una lluvia de hamburguesas el mismo día que haya una temperatura extrema"*.

8. Estadísticas [20 %]

El *output* o salida de la simulación son conocidos como medidas de desempeño, ya que posteriormente permiten comparar resultados de la simulación para loa distintos *inputs*. En esta sección deberán desplegar en consola[6] estadísticas de las medidas de desempeño de la simulación. Específicamente, tendrán que desplegar:

1. Cantidad promedio de dinero confiscado a los vendedores debido a llegada de carabineros.
2. Cantidad mínima, máxima y promedio por productos vendidos durante un día.

[6] Sean ordenados cuando muestren los resultados. Aprovechen los saltos de línea, tabs, etcétera.

3. Cantidad de confiscaciones que realizó cada tipo de **Carabinero**.
4. Número de llamadas que realizó *Quick Devil* a los **Carabineros**.
5. Número de veces que se realizó la *Concha Estéreo*.
6. Número de veces donde hubo *Temperaturas extremas*.
7. Número de veces donde hubo una *Lluvia de hamburguesas*.
8. Cantidad **promedio** de personas que almorzó entre las 12:00-12:59, 13:00-13:59 y 14:00-15:00.
9. Cantidad de alumnos que no almorzaron por mes.
10. Calidad promedio de productos de todos los vendedores por escenario.
11. Cantidad de **MiembrosUC** que se intoxicaron por vendedor.
12. Cantidad de **Productos** que se descompone.
13. Cantidad promedio de miembros de la PUC que abandonaron una cola de espera por día.
14. Cantidad promedio de vendedores por día que se quedaron sin *stock*.
15. Cantidad de veces que se engaño a los carabineros de personalidad *Dr. Jekyll* y *Mr. Hyde*.

9. Parámetros

Toda simulación recibe un *input*, que corresponde a los datos de entrada. Estos valores serán entregados en el archivo `parametros_iniciales.csv`. El *output* de la simulación corresponde a los datos de salida de esta y son conocidos como medidas de desempeño.

Respecto al *input* de la simulación, que son los parámetros que se encuentran en `parametros_iniciales.csv`, estos serán los siguientes:

- `dinero_estudiantes`: Corresponde al dinero base de la mesada de los estudiantes.
- `limite_paciencia`: Corresponde a los valores de $\alpha_{paciencia}$ y $\beta_{paciencia}$. Son entregados de la forma $\alpha_{paciencia}$; $\beta_{paciencia}$.
- `dinero_funcionarios`: Corresponde al dinero diario que tienen los funcionarios para gastar.
- `rapidez_vendedores`: Corresponde a los valores de los parámetros $\alpha_{rapidez}$ y $\beta_{rapidez}$ que se utilizan en la función de distribución de la rapidez de los vendedores. Son entregados de la forma $\alpha_{rapidez}$; $\beta_{rapidez}$.
- `probabilidad_permiso`: Corresponde al valor de $p_{permiso}$.
- `stock_vendedores`: Corresponde a los valores de los parámetros α_{stock} y β_{stock} que se utilizan en la función de distribución del *stock* de vendedores. Son entregados de la forma α_{stock}; β_{stock}.
- `dias_susto`: Corresponde al número de días en que los vendedores no vuelven a la feria, luego de que carabineros los clausure.
- `personalidad_jekyll`: Corresponde a los valores de la personalidad del policía `Dr. Jekyll`. Serán entregados de la forma *Tasa_productos_jekyll*; *Probabilidad_engaño_jekyll*.
- `personalidad_hyde`: Corresponde a los valores de la personalidad del policía `Mr. Hyde`. Serán entregados de la forma *Tasa_productos_hyde*; *Probabilidad_engaño_hyde*.

- `distribución_almuerzo`: Corresponde a los porcentajes X e Y, que indican la cantidad de personas que almuerzan de 13:00 a 13:59 y de 14:00 a 15:00 respectivamente. Serán entregados de la forma X; Y.
- `moda_llegada_campus`: Corresponde al valor $c_{llegada}$ que indica la moda de la distribución de la llegada de los miembros de la UC.
- `traslado_campus`: Corresponde al valor de $\lambda_{traslado}$, que es el parámetro de la distribución del tiempo que se demoran un miembro de la UC en llegar a los puestos de comida.
- `concha_estéreo`: Corresponde al valor p_{concha} que se encuentra en el evento Concha Estéreo.
- `llamado_policial`: Corresponde al valor del parámetro $\lambda t_{llamada}$ que se encuentra en el evento Llegada de policías.

9.1. Escenarios [25 %]

La simulación es una herramienta que se utiliza en la toma de decisiones. A raíz de esto, es de suma importancia analizar qué ocurre cuando se cambian las condiciones del problema que se está simulando. De esta forma, se pueden comparar las estadísticas obtenidas en estos nuevos escenarios.

En el contexto de simulación, un **escenario** es una simulación con todos los valores de *input* definidos. Luego, si se cambia alguno de estos valores, se considera como otro escenario. Una **réplica** es una ejecución de la simulación de un escenario en particular. Es importante notar que dos réplicas, aunque sean del mismo escenario, en la mayoría de los casos, no tendrán los mismos valores de *output*, puesto que estos dependen del valor que hayan tomado las variables aleatorias en la ejecución de la simulación.

No basta comparar los resultados obtenidos de una réplica de cada escenario para evaluar cuál es mejor, ya que, debido a la aleatoriedad del sistema, no sería una comparación representativa. Es por esto que para comparar distintos escenarios se deben realizar k réplicas para cada uno.

Finalmente, para comparar distintos escenarios, especificaremos qué medida de desempeño nos interesa analizar, donde el procedimiento será el siguiente:
- Consideraremos que se estará comparando escenarios bajo la medida de desempeño j.

1. Se realizan k réplicas para cada escenario.
2. Para cada escenario se calcula:

$$\forall \text{ i in escenarios, } \bar{Z}_i = \sum_{k \text{ in réplicas}} medida\_desempeño_{j,k}$$

3. Un $escenario_a$ será mejor que un $escenario_b$ para la medida de desempeño j si:

$$(\bar{Z}_a - \bar{Z}_b) > 0$$

4. Se obtiene el mejor de los escenarios como aquel que es mejor que el resto

En tu código deberás implementar una opción que permita comparar escenarios para una medida de desempeño en particular. En el archivo `escenarios.csv` podrás encontrar los parámetros necesarios para cada escenario. Los parámetros que se entregan para cada escenario son los mismos que en archivo `parametros_iniciales.csv`. Se deben desplegar en consola las distintas medidas de desempeño que se pueden escoger, las cuales corresponden a las estadísticas que entrega la simulación. Además, se debe permitir ingresar una, y luego especificar un número de réplicas para cada escenario. Cuando una réplica se ejecute

bajo este modo **NO** debe desplegar su *output* en consola.

Finalmente, debes mostrar en consola cuáles son los tres mejores escenarios evaluando bajo esa medida de desempeño.

9.2. Base de datos

Para esta tarea, contarás con dos archivos `.csv` con la información respectiva de los productos y personas. El separador de todos estos archivos será ";" (punto y coma) y están codificados en "UTF-8".

- `personas.csv`: Este archivo contiene la información de todas las personas de la simulación, esto incluye alumnos, funcionarios, vendedores y carabineros. Las columnas de este archivo son

  1. **Nombre**
  2. **Apellido**
  3. **Edad**
  4. **Vendedores de preferencia**: Secuencia de nombre y apellido de vendedores. Cada vendedor está separado por "-" (guión)
  5. **Entidad**: "Alumno", "Funcionario", "Vendedor" o "Carabinero"
  6. **Tipo Comida**: "China", "Mexicana" o "Snack"
  7. **Personalidad**: "Dr. Jekyll" o "Mr. Hyde".

  Si algún dato no corresponde al tipo de persona, en esa columna encontrarán un "-" (guión).

- `productos.csv`: Este archivo contiene información de cada producto existente. Las columnas son **Producto**, **Tipo**, **Precio**, **Calorías**, **Tasa de putrefacción** y **Vendido en**

9.3. Requisitos

Se evaluará la calidad de su código. En particular, su código debe contener lo siguiente:

- Uso de *properties*.
- *Sobrescribir* métodos cuando sea necesario.
- No usar *referencia circular* en la modelación de clases. Revisar este *issue* para mayor detalle o revisar este otro *issue*. Si tienes dudas, pueden crear un *issue* para mejor orientación sobre la modelación.
- **Todas** las variables de probabilidades y constantes que utilicen en su programa, que no estén en los archivos `.csv`, **deben** estar en un archivo llamado `variables.py`. De esta forma, en el resto de los módulos importan las variables como se ve a continuación:

```
"""
En el archivo variables.py
"""
PROB_TEMP = random()
CANT_SIMULACIONES = 1


"""
En sus archivos.py
"""
from variables import PROB_TEMP, CANT_SIMULACIONES

if __name__ == "__main__":
        while CANT_SIMULACIONES > 0:
            #run_simulacion
```

- *Documentar* métodos y funciones: todos sus módulos deben estar documentados, en donde se explica cada método y función e indican el tipo de dato de los argumentos recibidos y retornados, junto con una explicación de qué hace ese método o función. Sobrescribir un método de la clase `object` `__init__` `__str__`, `__add__` no requiere documentación. Si un método o función tiene nombre y argumentos descriptivos, no es necesario documentar; sólo basta con agregar un `# Descriptivo`.

```
class Auto:
    def __init__(self, patente):
        self.patente = patente

    def __str__(self):
        return self.patente

class Estacionamiento:
    def __init__(self, encargado, autos_dentro):
        self.encargado = encargado
        self.autos_dentro = autos_dentro

    def agregar_auto(self, auto): # Descriptivo
        self.autos_dentro.append(auto)

    def retirar_auto(self, patente):
        """
        Busca un auto según su patente y lo retira del estacionamiento
        patente Es un String
        Retorna un objeto clase Auto o None
        """
        index = 0
        while index < len(self.autos_dentro):
            if self.autos_dentro[index].patente == patente:
                return self.autos_dentro.pop(index)
            index += 1
        return None
```

10. Entregable

El entegrable de esta evaluación será una *Lista de Eventos*. En esta lista debes describir y todos los posibles eventos que pueden ocurrir durante la simulación de la forma:

`evento`$_1$ `| cuándo ocurre | qué pasa cuando ocurre el evento`$_1$ `(cambios, actualizaciones, etc)`
`evento`$_2$ `| cuándo ocurre | qué pasa cuando ocurre el evento`$_2$ `(cambios, actualizaciones, etc)`

.
.
.

`evento`$_n$ `| cuándo ocurre | qué pasa cuando ocurre el evento`$_n$ `(cambios, actualizaciones, etc)`

11. Restricciones y alcances

- Tu programa debe ser desarrollado en Python v3.6.
- Esta tarea es estrictamente individual, y está regida por el Código de Honor de la Escuela: Clickear para Leer.
- Tu código debe seguir la guía de estilos descrita en el `PEP8`.
- Si no se encuentra especificado en el enunciado, asume que el uso de cualquier librería Python está prohibida. Pregunta en el foro si es que es posible utilizar alguna librería en particular.

- Si no subes el entregable, tendrás **5 décimas menos** en la nota final de la tarea.
- El ayudante puede castigar el puntaje[7] de tu tarea, si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación algoritmos.
- Debes adjuntar un archivo `README.md` donde comentes sus alcances y el funcionamiento del sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*. **Tendrás hasta 24 horas después de la fecha de entrega** de la tarea para subir el `README.md` a tu repositorio.
- Crea un módulo para cada conjunto de clases. Divídelas por las relaciones y los tipos que poseen en común. **Se descontará hasta un punto si se entrega la tarea en un solo módulo[8].**
- No cumplir con alguno de los ítems de la sección **Requisitos** vendrá con un descuento asociado.
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro.

Tareas que no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).

[7] Hasta −5 décimas.
[8] No agarres tu código de un solo módulo para dividirlo en dos; separa su código de forma lógica