Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Tecnologías de la Información

Programación Orientada a Objetos I (Primavera 2018)

Proyecto FINAL

Objetivos del proyecto:

1. Aplicar los conocimientos obtenidos en el curso, específicamente los temas de Modelado UML, Programación Orientada a Objeto y Recursión.
2. Aplicar tu creatividad, autoaprendizaje, trabajo en equipo, honestidad, responsabilidad individual y colaborativa.

Colaborativamente:

- Trabajen colaborativamente en **equipos de 4 integrantes, no más - no menos.**
- Asignen entre ustedes un líder del proyecto. Que sea el líder no quiere decir que haga el trabajo de todos, sino que coordina a los demás para que trabajen de la mejor manera.
- Entre todo el equipo seleccionen uno de los proyectos propuestos más adelante. Analice el problema a resolver y diseñe juntos una posible solución (diagrama de clases). Una vez que tengan esto podrán repartirse tareas como convenga a sus intereses.
- Agenden fechas de reunión de su equipo para presentar entre ustedes sus avances, para resolver sus dudas, para integración y pruebas del programa completo, etc.

Criterios para el éxito:

En la fecha indicada más adelante, deberán tener implementado y corriendo totalmente su proyecto. **Cada integrante** del equipo deberá ser capaz de presentar el proyecto, explicar este y de responder a las preguntas que se le hagan al momento de la entrega.

Responsabilidad Individual:

Cualquier integrante del equipo podrá ser seleccionad@ **al azar** el día de la entrega del proyecto para la presentación de su trabajo. **Su calificación será la del equipo.**

Descripción de los Proyectos

Todo proyecto de incluir lo siguiente:

1. Diagrama UML
2. Programación Orientada a Objetos en C++
3. Recursividad o pilas o colas
4. Archivos

1. Exploración de un laberinto

Un problema muy común y relevante para el mundo de la robótica es ¿Cómo encontrar la salida de un laberinto? El problema que queremos resolver es ayudar a nuestro W*ally* a encontrar la salida de un laberinto virtual. En nuestro problema asumiremos que Wally es dejado caer en alguna parte en medio del laberinto y debe encontrar la salida.

En la figura siguiente se da una idea de cómo podría verse el laberinto.

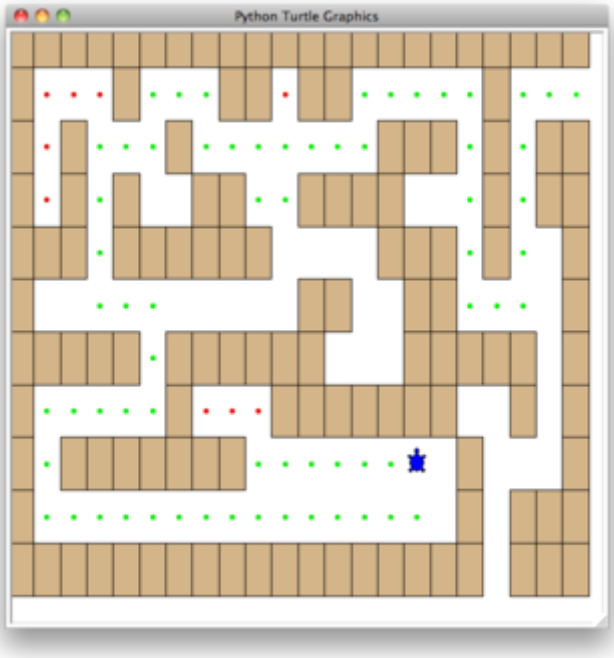


Figura 2: El programa de búsqueda en un laberinto terminado

EL laberinto, podemos implementarlo como un arreglo de celdas, donde cada celda puede estar abierta o cerrada (pared). Wally sólo puede pasar a través de los cuadrados abiertos del laberinto. Si se topa con una pared, debe intentar continuar en una dirección diferente. Se requiere de un procedimiento sistemático para encontrar la salida del laberinto, este procedimiento es recursivo.

- Desde nuestra posición de partida, primero intentaremos avanzar un cuadrado hacia el Norte y luego recursivamente probaremos nuestro procedimiento desde allí.
- Si no tenemos éxito al intentar un camino hacia el Norte como primer paso, daremos un paso hacia el Sur y repetiremos recursivamente nuestro procedimiento.
- Si ir hacia el Sur no funciona, entonces intentaremos dar un paso hacia el Oeste como nuestro primer paso y aplicaremos nuestro procedimiento recursivamente.
- Si el Norte, el Sur y el Oeste no han tenido éxito, aplicaremos el procedimiento recursivamente desde una posición un paso hacia nuestro Este.
- Si ninguna de estas direcciones funciona entonces no hay manera de salir del laberinto y hemos fracasado.
- En caso contrario podemos encontrar la salida y seremos libres.

El procedimiento anterior es muy general y se requiere que consideren varios casos para implementarlo adecuadamente. Entre estos, los casos base.

Recuerden que el problema del laberinto tiene raíces tan profundas como el mito griego sobre Teseo que fue enviado a un laberinto para matar al minotauro. Teseo usó una madeja de hilo para ayudarse a encontrar su camino de regreso una vez que hubiera eliminado a la bestia.

Por lo tanto Wally también requiere que contemos con una estrategia para recordar dónde hemos estado. En este caso vamos a suponer que tenemos una bolsa de migas de pan que podemos dejar caer a lo largo de nuestro camino. Si damos un paso en una dirección determinada y encontramos que ya hay una miga de pan en ese cuadrado, sabemos que debemos retroceder inmediatamente y probar la siguiente dirección en nuestro procedimiento.

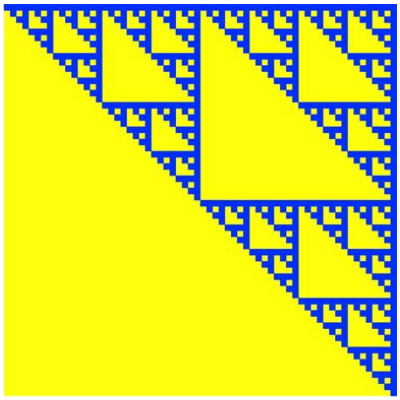
Consideraciones:

- Se debe modelar el problema utilizando adecuadamente el paradigma Orientado a Objetos. Esto es, identificar clases, objetos y relaciones
- La matriz de ser de objetos celdas (no enteros). Por default debe ser de 50 X 50, pero se pueden manejar diferentes niveles de complejidad, relacionados con el tamaño de la matriz.
- Inicialmente se deberá incluir en un menú las opciones para que la estructura del laberinto pueda inicializarse de cualquiera de las siguientes opciones: ser leído desde un archivo, de forma aleatoria o manual (introducido por el usuario)
- Cada movimiento de wally a través del laberinto deberá ser desplegado en pantalla. La “animación” será a consola (Opcionalmente puedes utilizar alguna GUI)
- Se debe considerar el caso de un overflow, o que no se llegue a la salida. Pensar en una estrategia para reconocer esto y que no “se cuelgue”.
- Para la opción en la cual se inicializa la matriz de un archivo, el formato en el que recibes la información queda a tu criterio, de la forma que más fácil se te haga.

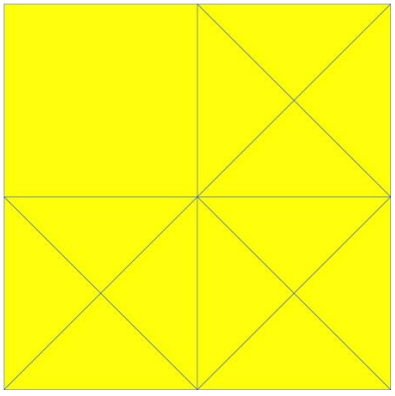
2. Recursion in Art

Muchas de las formas que se encuentran en la naturaleza exhiben alguna forma de auto-similitud; la forma más grande parece contener copias más pequeñas de sí mismo a diferentes escalas. Abundan los ejemplos en el mundo de las plantas; vemos también en montañas, nubes, la estructura ramificada de ríos y vasos sanguíneos, patrones en pieles de animales, etc. La naturaleza impone restricciones a las reglas de crecimiento, pero eso no significa que el artista lo necesite. Primero imitando las formas en la naturaleza, el artista se encuentra cambiando una forma, una escala o un color para producir una imagen más abstracta pero visualmente atractiva. Algunos algoritmos recursivos que producen imitaciones de formas encontradas en la naturaleza; luego los combinamos en lo que un colega mío denominó "Paisajes matemáticos", y finalmente vamos a abstraer las formas en diseños visualmente atractivos.

En este proyecto no vamos a hacer precisamente arte, sino más bien una construcción matemática que es un fractal conocido como el **triágulo de Sierpínski**. La alfombra de Sierpiński es un conjunto fractal descrito por primera vez por Wacław Sierpiński en 1916. El triángulo de Sierpinski ilustra un algoritmo recursivo de tres vías.

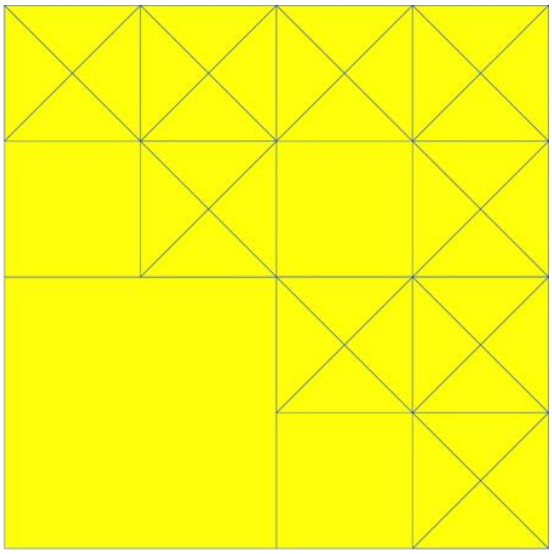


Como puedes ver, es una colección de cuadritos dibujados en un patrón particular dentro de una región cuadrada. Aquí está cómo dibujarlo. Comienza con la región completa cuadrada y divídela en cuatro secciones así:

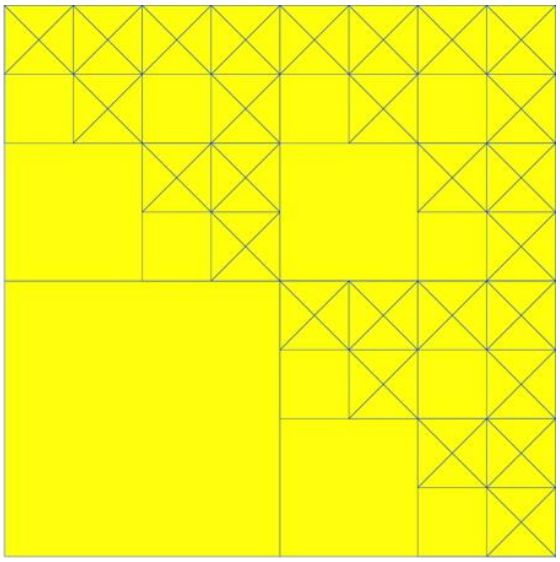


Triángulo de Sierpinski de 2 por 2

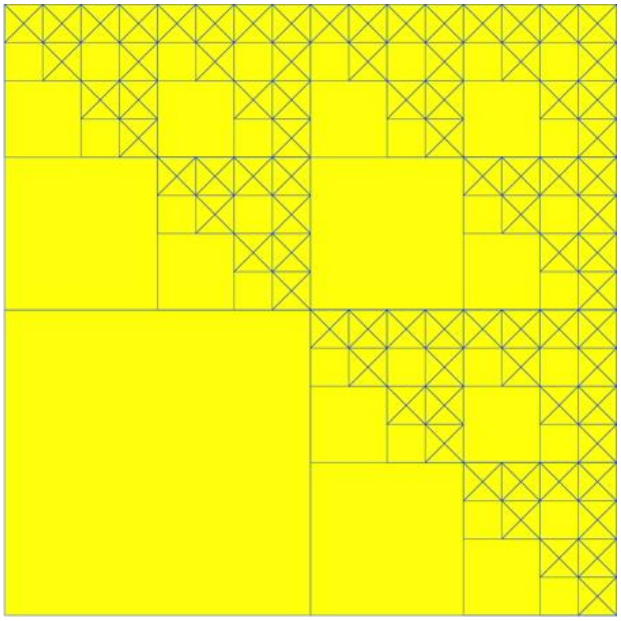
Toma los tres cuadrados con una × en ellos (la parte superior izquierda, la parte superior derecha y la parte inferior derecha) y divídelas en cuatro secciones de la misma manera. Continúa. Divide cada cuadrado con una × en cuatro secciones y pon una × en los cuadrados en la parte superior izquierda, superior derecha, superior e inferior derecha, pero nunca en la parte inferior izquierda.



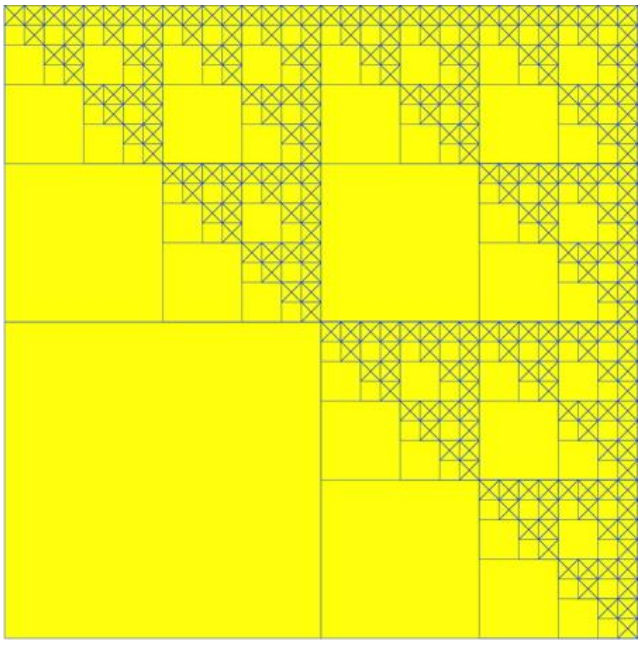
Triángulo de Sierpinski de 4 por 4



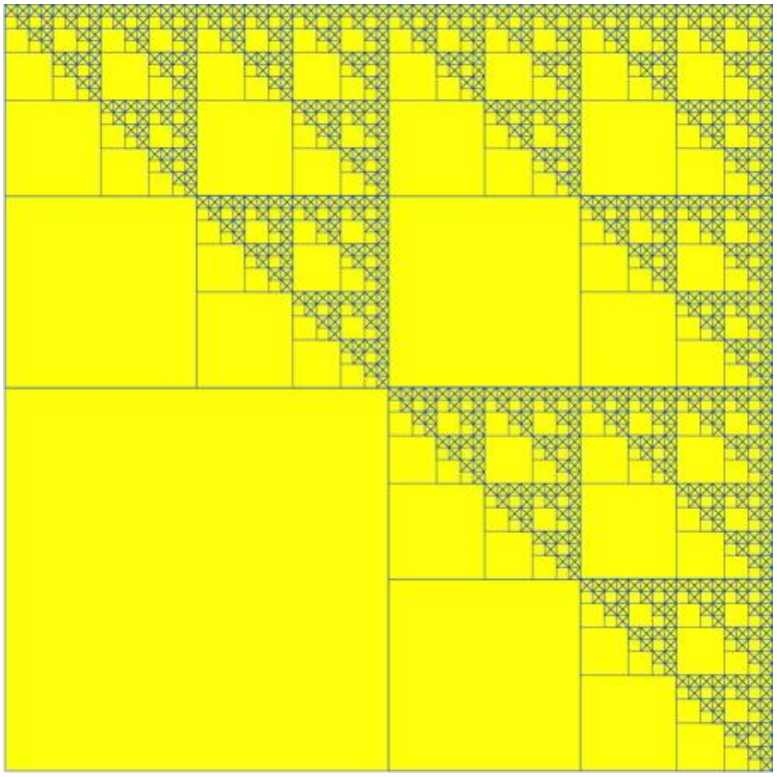
Triángulo de Sierpinski de 8 por 8



Triángulo de Sierpinski de 16 por 16

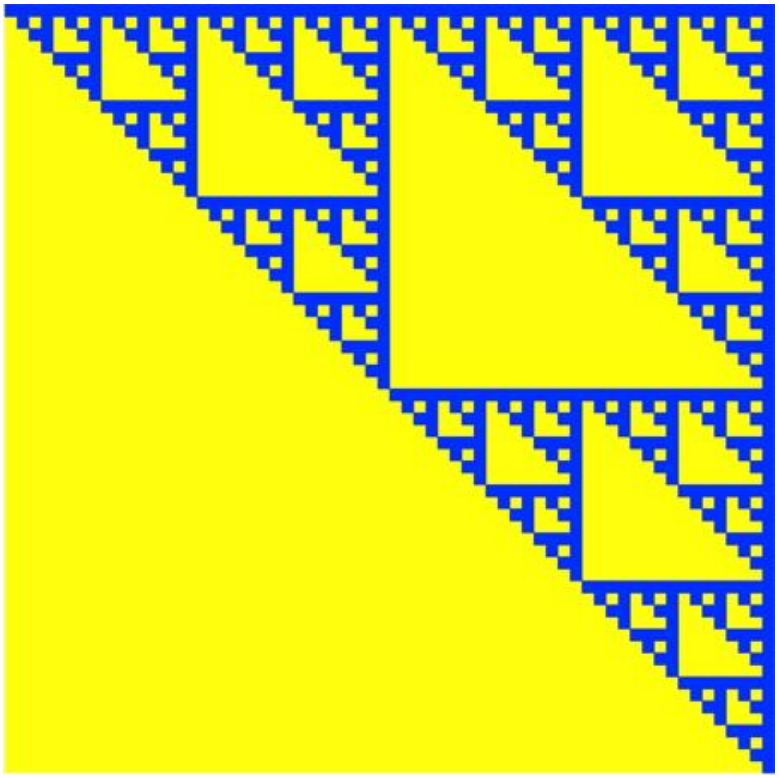


Triángulo de Sierpinski de 32 por 32



Triángulo de Sierpinski de 64 por 64

Una vez los cuadrados se hagan lo suficientemente pequeños, deja de dividir. Si rellenas cada cuadrado con una × y te olvidas de todos los demás cuadrados, obtienes el triángulo de Sierpinski. Aquí está una vez más:



Triángulo de Sierpinski completo

Para resumir, aquí está cómo dibujar un triángulo de Sierpinski en un cuadrado:

- Puesto que podemos seguir aplicando el algoritmo indefinidamente, ¿cuál es el caso base? El caso base se establece arbitrariamente como el número de veces que queremos dividir el cuadrado en partes. A veces llamamos a este número el "grado" del fractal. Cada vez que hacemos una llamada recursiva, le restamos 1 al grado hasta llegar a 0. Cuando alcancemos un grado de 0, dejaremos de hacer llamadas recursivas
- De lo contrario, divide el cuadrado en cuatro: parte superior izquierda, parte superior derecha, parte inferior derecha y parte inferior izquierda.
- Resuelve de manera recursiva tres subproblemas:
    - 1. Dibuja un triángulo de Sierpinski en el cuadrado superior izquierdo.
    - 2. Dibuja un triángulo de Sierpinski en el cuadrado superior derecho.
    - 3. Dibuja un triángulo de Sierpinski en el cuadrado inferior derecho.
- Puedes escoger cualesquiera 3 de los cuatro cuadrados en los cuales dibujas triángulos de Sierpinski de manera recursiva. El resultado solo saldrá rotado por un múltiplo de 90 grados del dibujo de arriba.

Consideraciones:

- Se debe modelar el problema utilizando adecuadamente el paradigma Orientado a Objetos. Esto es, identificar clases, objetos y relaciones
- Se debe resolver dando una solución recursiva y como dato de entrada se debe dar el grado del fractal.
- La "animación" del triángulo de Sierpinski es a consola

---

3. Agencia de autos

Se requiere desarrollar un sistema que simule el funcionamiento de cualquier empresa de compra-venta que cuente con catálogos de productos, por ejemplo: agencia de autos, inmobiliaria, lote de autos, maquinaria, boutique, ferretera, etc. y que puedan dar sus productos en pagos mensuales. Además, La empresa debe ser tal, que requiera de Altas, Bajas, Cambios y visualización de sus productos. Se usará la clase **vector** de la biblioteca de C++, como estructura de almacenamiento.

El funcionamiento básico del programa debe ser el siguiente:

a. ostrar un menú con las siguientes opciones
    1. Personal Autorizado: Requiere contraseña (password) de acceso
    2. Vendedor: Entra sin password
b. Las opciones que puede realizar el **personal autorizado** son:
    1. Altas. En esta opción la empresa capturará la información de algún producto y lo agregará a la lista de productos ordenados por una **clave** que debe ser **única**. La información que se solicitará al usuario es la siguiente:
        - ✓ Clave
        - ✓ Nombre
        - ✓ Descripción
        - ✓ Foto [opcional]
        - ✓ Precio
        - ✓ Status (apartado, libre) – La empresa permite apartados de productos
        - ✓ Cualquier otra característica, por ejemplo dirección, o medidas, etc.
    2. Bajas: Si se desea dar de baja (eliminar) un producto, se deberá checar que el producto no este apartado, de lo contrario deberá enviar un mensaje indicando que fue imposible realizar la baja de ese producto y el motivo.
    3. Modificaciones: Si se requiere hacer alguna modificación al producto esta se deberá hacer por clave. Si no existe la clave se le deberá indicar al usuario.
    4. Visualizar: Esta opción permitirá ver en una tabla, todos los productos existentes, ordenados por clave.
c. Las opciones para los vendedores son:
    1. Cotizar en pagos: Esta opción permitirá a un cliente cotizar un producto en pagos mensuales dependiendo de su sueldo. El cliente podrá seleccionar la marca del auto que elija y dependiendo de su sueldo mensual, la agencia obtendrá el número de mensualidades y las cuotas a pagar cada mes bajo el siguiente criterio:
        - ✓ Un producto X cuesta P pesos y se puede pagar hasta en cinco mensualidades que representan el 50%, 40%, 30%, 20% y 10% del precio total del producto respectivamente.
        - ✓ Si una persona que gana N pesos mensuales, desea adquirir un producto, puede hacerlo en 5 meses siempre y cuando su sueldo le permita pagar completamente cada mensualidad.
        - ✓ Si una de las cuotas a pagar es mayor que su sueldo, la empresa le permite pagar dicha cuota en cinco sub-cuotas mensuales, que representan también el 50%, 40%, 30%, 20% y 10% del monto de esa cuota
        - ✓ Si alguna de esas cinco sub-cuotas son aún mayores que su sueldo, dicha cuota puede pagarse en cinco sub-sub-cuotas mensuales con los mismos porcentajes, y así sucesivamente.

Por ejemplo, para un terreno que cuesta 80,000 y un cliente que gana 20,000 mensual, la empresa le asignaría 21 mensualidades como sigue:

Mensualidad	Cuota
1	10,000
2	8,000
3	6,000
4	4,000
5	2,000
6	16,000
7	12,000
8	8,000
9	4,000
10	16,000
11	12,800
12	9,600
13	6,400
14	3,200
15	12,000
16	9,600
17	7,200
18	4,800
19	2,400
20	16,000
21	8,000

Se debe mostrar esta tabla de cuotas mensuales.

2. Vender producto: En esta opción se capturará la clave del producto, se mostrará la información de este y se solicitará al vendedor que confirme que es el producto deseado. Automáticamente se debe eliminar el auto de la lista. En caso de que el producto esté apartado, se deberá indicar al cliente y esperar que el personal autorizado cambie el status de acuerdo a las reglas del negocio.
3. Buscar productos. De deberá mostrar el listado con la información completa de los productos que cumplan la búsqueda. Deberpan existir diferentes criterios, dependiendo del producto, por ejemplo:
    - ✓ Por un rango de costo (ejemplo 100,000-120,000)
    - ✓ Por ubicación (en caso de casas)
    - ✓ Por modelo (en caso de autos)
    - ✓ Por color
    - ✓ Etc.
4. Apartar producto: Esta opción permite al vendedor seleccionar un auto por clave, mostrar su información completa y si es correcto, confirmar el apartado. Automáticamente deberá cambiar el status del producto a “apartado”
5. Terminar: Terminará la ejecución del programa hasta que el usuario seleccione esta opción

1. En equipo seleccionen el giro de su empresa
2. Escriban la descripción general del problema, analicen y diseñen su solución UML.

Fecha de entrega:

- ✓ Miercoles 16 de Mayo 2018, en el horario de 10:00 a 3:00 hrs. El tiempo estimado de entrega es de 30 mins.
- ✓ La entrega es por EQUIPO. Cada equipo debe entregar lo especificado en los *entregables.*
- ✓ Se establecerán por sorteo **el horario** de entrega por equipo y se publicará en la plataforma Moodle y Facebook con anticipación. Además se indicará el lugar de entrega. NO HABRÁ prórrogas (no cambios de día). Si algún equipo por razones plenamente justificadas no pueden en el horario asignado, deberán comunicármelo vía inbox, **al menos un día antes de la entrega.**

Forma de Revisión:

1. Preparar una presentación en Powerpoint, que incluya una breve descripción del problema a resolver, alguna investigación al respecto, explicación/ejemplo de los algoritmos utilizados, Modelo UML y lo que consideren relevante.
2. Todos los integrantes del equipo deberán estar presentes para ser considerados en la calificación.
3. Cualquier integrante del equipo puede será seleccionado **al azar** para hacer la presentación completa del trabajo y responder a las preguntas que se le indiquen.
4. El resto del equipo responderá solamente a preguntas cuando se le indique.

Entregables:

En un CD con PORTADA entregar;

- ✓ La carpeta del proyecto completa (programa fuente, presentación).
- ✓ Los archivos fuente deben estar **identados y comentados** y contener como encabezado los nombres de los integrantes del equipo.

Rúbrica de evaluación

Criterio	Pre-formal	Receptivo	Resolutivo	Autónomo	Estratégico
Presentación 15%	Se presentó informalmente el proyecto	Se presentó el proyecto usando alguna herramienta de presentaciones, no se incluyó UML	Se presentó el proyecto usando alguna herramienta de presentaciones. Se mostró diagrama, se explicó algoritmo recursivo básico.	Se presentó el proyecto y su solución formalmente, se investigó aplicaciones relacionadas y se explicó el algoritmo recursivo	No solo se presentó su solución formalmente, sino se justificó su solución y se propuso un algoritmo óptimo
Modelado de la solución 30%	Utilizó el paradigma Orientado a Objetos de forma aceptable	Utilizó el paradigma Orientado a Objetos, bien modelado y aceptablemente codificado	Utilizó el paradigma Orientado a Objetos bien modelado y codificado	Utilizó el paradigma Orientado a Objetos cuidadosamente modelado y codificado respetado estádares	Utilizó el paradigma Orientado a Objetos y correctamente, modelando todos los elementos y utilizando algún patrón de diseño (MVC)
Funcionamiento del programa 30%	Entrega un programa ejecutable que resuelve el problema con posibles errores	Entrega un programa ejecutable que resuelve el problema correctamente, al menos para un caso de prueba	Entrega un programa ejecutable que resuelve el problema correctamente para varios casos de prueba	Entrega un programa ejecutable que resuelve el problema correctamente en todos los casos probados	Entrega un programa ejecutable que resuelve el problema correctamente en todos los caso y además en un tiempo óptimo.
Interface 25%	Algunos resultados se muestran en pantalla	Se muestra una interfaz a consola con una animación básica o corrida básica	Se muestra una interfaz a consola con una animación creativa	Se muestra una GUI muy básica	Se muestra una interfaz gráfica, que muestra la solución de forma clara y creativa