GAME PROGRAMMER

Los postulantes para el área de programación deben poder resolver algunos de los problemas enunciados en esta página. No es necesario resolver todos los problemas.

Envía tus respuestas con los archivos requeridos a jobs@bamtang.com

No olvides de enviar también tu hoja de vida con nombre, teléfono y estudios realizados.

Si tienes un blog de programación o portafolio online con ejemplos de tu trabajo no olvides mencionarlo para obtener puntos adicionales.

Sobre el Examen

1. Cada problema a continuación tiene un puntaje asociado.
2. Escoge los problemas que llamen tu atención y envía sus respuestas por correo.
3. Para la solución emplea solo C++
4. De ser posible, trata de usar solo un archivo por problema
5. Cualquier nota u observación ponla como comentario al inicio del problema.
6. Esta bien si todo tu código y comentarios está en inglés.

Sobre la Evaluación

1. No enviar ejecutables, solo proyectos con fuentes para cada problema. Deben compilar en Visual C++.
2. Si un proyecto no compila, será calificado con nota cero. Por favor revisa bien tu proyecto antes de enviarlo.
3. Si tu código esta desordenado y no es fácil de leer (pero funciona) puede ser penalizado hasta con un 35% del puntaje del problema.
4. El puntaje mínimo aprobatorio es de 11 puntos, no es necesario resolver todos los problemas.
5. Después de aprobar este examen, se te tomará un examen similar en nuestras oficinas, aprobarlo sólo te servirá si lo resuelves personalmente.
6. Puedes usar cualquier referencia (Wikipedia, libros, blogs, etc) con tal que el código que envíes sea completamente tuyo y puedas explicarlo linea por linea.

1. CIPHER

Desencripta el siguiente mensaje:

```
Bgc-bfufb tegaedppqna ql aggv zge xof tegaedppfe'l lgjb.
Xof adpf vflqanfe logjbvn'x hf pdwqna d cgebv qn coqro xof
tbdkfe ql mjlx d lpdbb tdex. Xof tbdkfe QL XOF HGLL; qx'l
kgje vjxk xg fnxfexdqn oqp ge ofe.
Zgrjl ql d pdxxfe gz vfrqvqna codx xoqnal kgj def ngx agqna
xg vg.
Xof rglx gz dvvqna d zfdxjef qln'x mjlx xof xqpf qx xdwfl
xg rgvf qx. Xof rglx dblg qnrbjvfl xof dvvqxqgn gz dn
```

```
gntxdrbf xg zjxjef fstdntqgn. xof xeqrw qt xg tqrw xof
zfdxjefl xodx vgn'x zqaox fdro gxofe. - Mgon Rdepdrw.
(ccc.adpdljxed.rgp/uqfc/nfcl/234346?utkjpvbjr)
(ccc.hedqnkijgxf.rgp/ijgxfl/djxogel/m/mgon_rdepdrw.oxpb)
```

Si se sabe que solo se han encriptado las letras del alfabeto (a - z).

Para ello ordena las letras del mensaje encriptado de acuerdo al numero de veces que se repiten (de mayor a menor) y compáralo con la tabla de frecuencia de caracteres del lenguaje en que esta escrito este mensaje, que para este caso es:

freqLang = "TEOAISRHNUCMDLGWFPYKJBVQX" (de mas frecuente a menos frecuente)

Implementar la función: **decryptMessage**(message, freqLang)

Tu programa debe imprimir el texto desencriptado en pantalla.

No uses archivos ni pidas ingresar datos.

Mantener las mayúsculas y minúsculas del mensaje, así como los saltos de linea.

(4 PUNTOS)

2. CHESS BOARD

Implementar una función que dibuje un tablero de ajedrez con la posición de las piezas pasada como parámetro.

El parámetro a pasar será una cadena compuesta solamente de los siguientes caracteres:

p, r, b, n, q, k, /, 1, 2, 3, 4, 5, 6, 7, 8

Donde cada pieza es representada por una carácter, siendo las negras caracteres en minúscula y las blancas caracteres en mayúsculas. Los caracteres son:

p = peón

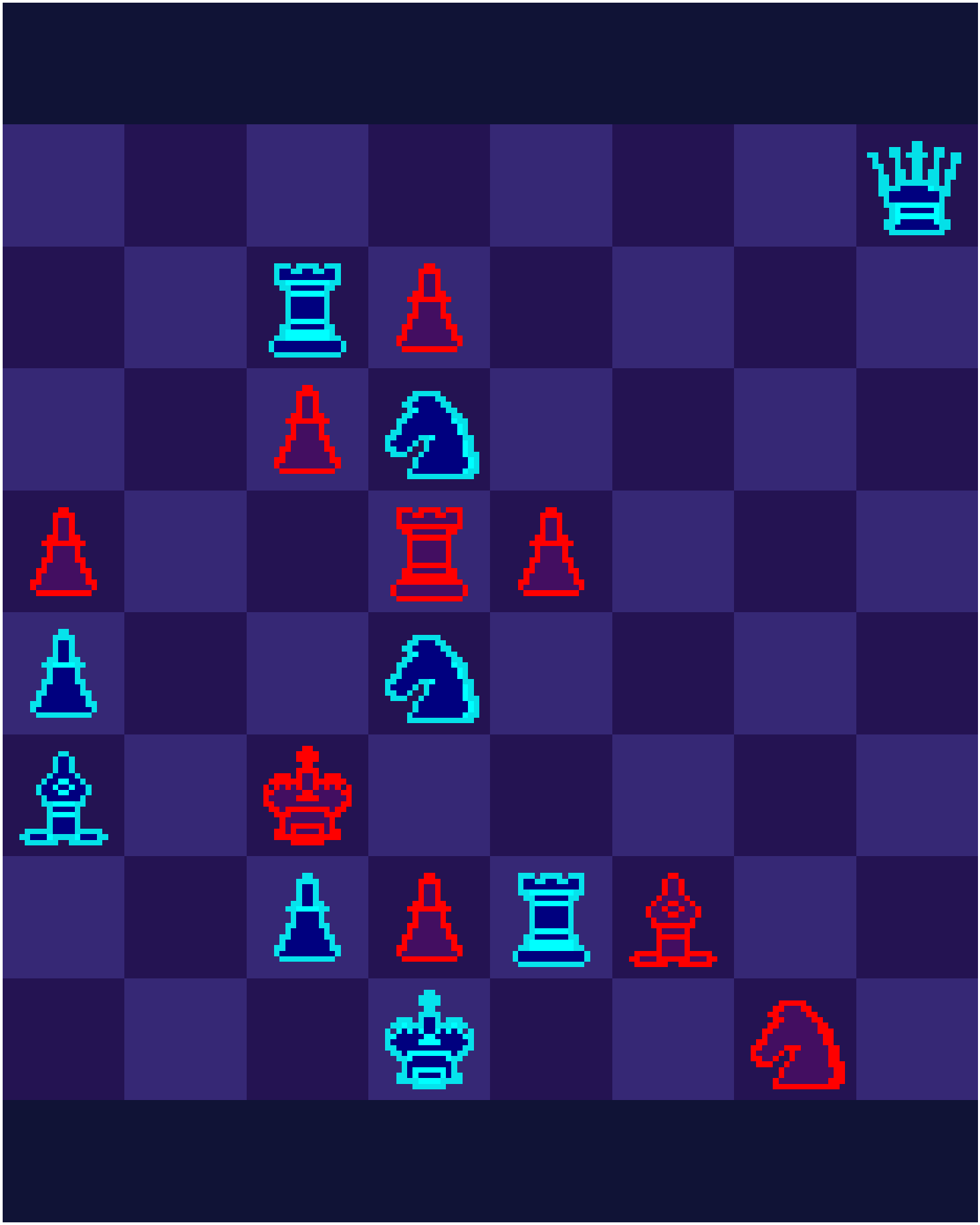
r = torre

b = alfil

n = caballo

q = reina

k = rey



Para separar las filas dentro de la cadena se emplea el carácter "/", existen exactamente 8 filas en la cadena.

Para indicar espacios vacíos en la fila se colocan los números del 1 al 8. Por ejemplo la fila "4p3" esta compuesta por 4 celdas vacías, un peón y finalmente tres celdas vacías El numero de celdas por fila debe ser siempre 8.

Así por ejemplo la posición inicial de ajedrez se representa por:

"rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR"

Implementar la función: **drawChessBoard**(initPosition)

Usa cualquier imagen para las piezas así como la librería gráfica que prefieras, **pero debes cargar y mostrar imágenes BMP, PNG o JPG en tu programa**,

No se considerará válidas soluciones que sólo usen la consola de texto.

Verifica que las imágenes estén ubicadas correctamente para que no hayan problemas al ejecutar tu proyecto.

(5 PUNTOS)

3. MICHI

Imprime una lista (en un archivo de texto) de todas las jugadas posibles de tic-tac-toe (michi) ordenadas de la siguiente manera:

```
  N     X 0 X 0 X 0 X 0 X Win
000001 1 2 3 4 5 6 7 0 0 X
000002 1 2 3 4 5 6 8 7 9 X
000003 1 2 3 4 5 6 8 9 7 X
000004 1 2 3 4 5 6 9 0 0 X
000005 1 2 3 4 5 7 6 8 9 X
000006 1 2 3 4 5 7 6 9 8 E
...
...
```

```
...
255162 9 8 7 6 5 3 2 4 1 X
255163 9 8 7 6 5 3 4 1 2 E
255164 9 8 7 6 5 3 4 2 1 X
255165 9 8 7 6 5 4 1 0 0 X
255166 9 8 7 6 5 4 2 1 3 X
255167 9 8 7 6 5 4 2 3 1 X
255168 9 8 7 6 5 4 3 0 0 X
```

Donde la primera columna es el número de la jugada y las siguientes nueve columnas son las jugadas realizadas por el jugador X y O consecutivamente. El dígito cero indica que no se realizó ninguna jugada en ese turno (pues el juego concluyó antes) y un dígito del 1-9 indica que se ha realizado la jugada en la posición definida por el siguiente tablero numerado:

```
-------------
| 1 | 2 | 3 |
-------------
| 4 | 5 | 6 |
-------------
| 7 | 8 | 9 |
-------------
```

Tu función debe imprimir además en la columna final el resultado de la partida:

(X: cuando X gane, O: cuando O gane, E: empate).

Implementar la función: **listAllTicTacToeGames**(fileName)

(6 PUNTOS)

4. POKER FACE

En esta versión simplificada del juego de Poker, existen 4 tipos de cartas (abreviaciones entre paréntesis):

Tréboles (**C**), Espadas (**S**), Corazones (**H**) y Diamantes (**D**).

Hay 13 cartas por cada tipo, dando un total de 52 cartas, las cartas ordenadas de menor valor a mayor valor son:

2, 3, 4, 5, 6, 7, 8, 9, 10 (**T**), Jack (**J**), Reina (**Q**), Rey (**K**), Ace (**A**).

Una mano consiste de 5 cartas. Existen 10 clases de manos, ordenadas de menor a mayor son:

- **HIGH CARD:** El valor de la carta de mayor valor
- **ONE PAIR:** Dos cartas del mismo valor (un par)
- **TWO PAIRS:** Dos pares diferentes.
- **THREE OF A KIND:** Tres cartas del mismo valor
- **STRAIGHT:** Todas las cartas son valores consecutivos
- **FLUSH:** Todas las cartas son del mismo tipo
- **FULL HOUSE:** Tres cartas del mismo valor y un par
- **FOUR OF A KIND:** Cuatro cartas del mismo valor
- **STRAIGHT FLUSH:** Todas las cartas son valores consecutivos del mismo tipo
- **ROYAL FLUSH:** 10, Jack, Queen, King, Ace, del mismo tipo

Si dos jugadores tienen manos de la misma clase entonces la mano que esta conformada por cartas de mayor valor gana; por ejemplo un par de ochos le gana a un par de cincos (ver ejemplo 1 de abajo). Si aun así hay empate, por ejemplo, ambos jugadores tiene un par de reinas, entonces las cartas de mayor valor de cada mano son comparadas (ver ejemplo 4 de abajo); si las cartas de mayor valor son iguales las cartas que le siguen de mayor valor son comparadas y así sucesivamente.

Ejemplos:

Mano	Jugador 1	Jugador 2	Ganador
1	5H 5C 6S 7S KD	2C 3S 8S 8D TD	Jugador 2

1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD Par de ochos	Jugador 2
2	5D 8C 9S JS AC HIGH CARD Ace	2C 5C 7D 8S QH HIGH CARD Queen	Jugador 1
3	2D 9C AS AH AC Tres Aces	3D 6D 7D TD QD FLUSH de diamantes	Jugador 2
4	4D 6S 9H QH QC Par de reinas Máxima carta 9	3D 6D 7H QD QS Par de reinas Máxima carta 7	Jugador 1
5	2H 2D 4C 4D 4S Full House de 4	3C 3D 3S 9S 9D Full House de 3	Jugador 1

Verifica tu código con el archivo de jugadas: poker.txt, el cual contiene 1000 manos aleatorias entregadas a dos jugadores. Cada línea del archivo contiene 10 cartas (separadas por un solo espacio): las primeras cinco son las cartas del jugador 1 y el resto son las del jugador 2. De las 1000 manos en este archivo el jugador 1 gana 376 manos.

Crea todas las funciones necesarias para resolver este problema de una modo genérico (por ejemplo: con otro archivo de jugadas). Asume que todas las manos son válidas, sin caracteres inválidos ni cartas repetidas, que las manos no están ordenadas y que por cada línea existe un claro vencedor.

(6 PUNTOS)

5. LABERINTO

Crear un generador de laberintos aleatorios de dos dimensiones de tamaño 'n' (n: impar > 3). El punto inicial **A** siempre se encontrará en la esquina superior izquierda. El punto final **B** deberá encontrarse al final de la ruta **mas larga** dentro de tu laberinto generado (que no necesariamente será la esquina superior derecha).

Ademas para ir de **A** a **B** deberá existir **solo un camino posible**, no se te permite volver a caminar sobre lo ya recorrido.

Ejemplo:

n = 11 (* = muro, A = inicio, B = fin)

```
***********
*A  *    B*
*** * *****
* * * *   *
* * * * * *
*   *   * *
* *** *** *
*   * *   *
*** *** * *
*       * *
***********
```

Implementar la función: **createMaze2D**(n)

No generar cualquier laberinto, recuerda que se pide un laberinto de un solo camino posible entre A y B.

No se considerán soluciones que generen laberintos con múltiples caminos entre A y B.

(7 PUNTOS)

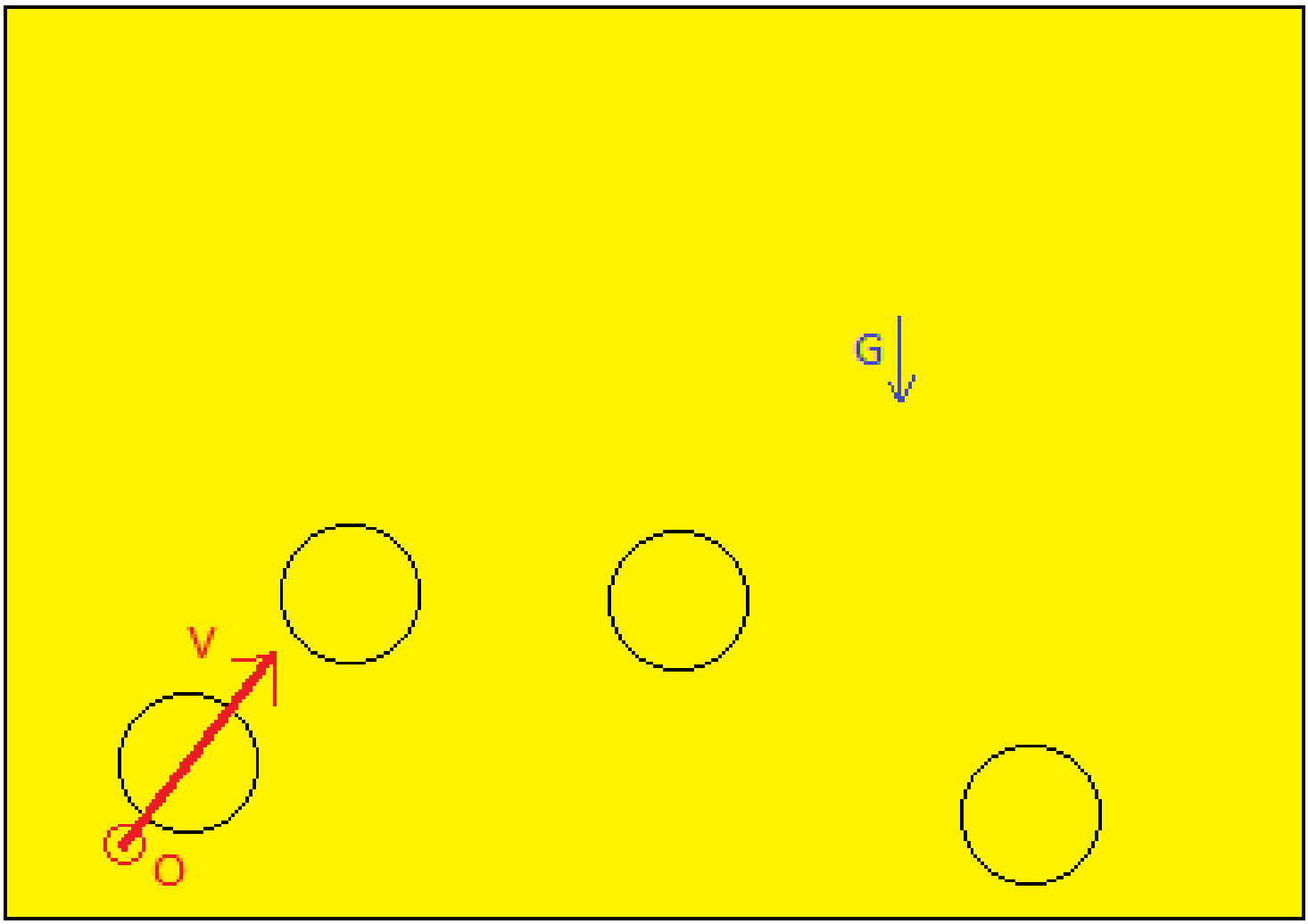
6. FÍSICA

Cree un simulador para disparar balas circulares. Considerar por simplicidad un cuarto rectangular cerrado y de gravedad **G**. Las balas deben de chocar contra las paredes y entre ellas, luego de un tiempo de estar inmóviles deben desaparecer.

Para este problema considere coeficientes de fricción y elasticidad convenientes y demás suposiciones que cree necesarias (y documéntelas en su código). Su programa debe permitir mover el origen de disparo **O** y la velocidad y angulo inicial de disparo **V**. Las balas deben ser disparadas al apretar la tecla de espacio. **No uses librerías físicas existentes**, pero puedes usar cualquier librería de gráficos 2D.

Implementar la función: **shootBall**()

(10 PUNTOS)


G
V
O

7. LABERINTO 3D

Implementar un visor 3D que permita visualizar y navegar los laberintos generados por el problema 6 (Reemplazar cada carácter '*' de la solución por un cubo y asignarle una textura arbitraria). Si tienes experiencia previa programando gráficos o videojuegos puedes considerar movimientos de cámara, colisiones, sombras, optimizaciones para cargar niveles grandes, shaders, modelos animados navegando el nivel, pathfinding, implementar un pequeño juego, etc.

Puedes utilizar librerías basadas en OpenGL o DirectX para resolver este problema, sin embargo si usas motores completos para el render como Unreal, Godot, Irrlicht, etc solo se considerará tu solución en base a los adicionales y si esta en C++, el objetivo de este problema es hacer el render tu mismo

Implementar la función: **render3D**()

(12 PUNTOS, en caso de implementar adicionales el puntaje máximo de este problema es 20)