Pregunta 1

Implementación:

Para el desarrollo de este problema se utilizó un código de búsqueda en espacio soluciones, ya que necesitamos conocer las distintas formas en las que se puede representar un par de tripletas, esta forma puede ser una mejor manera de abordarlo.

Para la realización de este código se define una posible solución como una matriz de 3x3 (arreglo de 3 enteros, en donde cada elemento del arreglo es otro arreglo de 3 enteros), esta matriz tendrá en ella solamente “1’s” o “0’s”, estos representan una casilla negra o una casilla blanca respectivamente.

El algoritmo utilizado para generar cada posible solución consiste de 9 ciclos for anidados uno dentro del otro, cada uno de estos ciclos tiene una variable (**a, b, c, d, e, f, g, h, i**) que tomará los valores entre de 0 o 1, cada una de estas variable representan una posición dentro de la matriz solución ([0,0], [0,1], [0,2], [1,0], [1,1], [1,2], [2,0], [2,1], [2,2]) respectivamente. Lo que se busca lograr con estos ciclos anidados es hacer todas las combinaciones posibles con estas 9 posiciones.

Una vez llegado al último for anidado, se llama a la función **crearCuadrado** entregando como argumentos las 9 variables de los for, esta función se encargar de crear una matriz 3x3, asignando cada variable en su posición correspondiente, finalmente la matriz retornada por la función será almacenada en la variable **cuadrado**, luego esta posible solución será agregada a una lista de soluciones (lista de matrices) por medio de la función **agregarSolucion**; esta función recibe como argumentos la lista de soluciones, la cantidad de elementos que hay dentro de la lista y la solución que se desea agregar a la lista. Lo que hace **agregarSolucion** es crear una nueva lista de soluciones con memoria para una solución más, copiar todos los elementos de la antigua lista y luego agregar la solución entregada al final de esta nueva lista; luego esta nueva lista de soluciones será el nuevo “valor” de la variable **listaSoluciones**.

Luego de que termine el algoritmo que genera todas las posibles soluciones, se procede a filtrar todas las posibles soluciones de la lista de soluciones mediante las 2 restricciones que tiene este problema. Estas restricciones están dadas por 2 tripletas de números entregadas por el usuario al ejecutar el programa:

1) La primera tripleta indica la cantidad de casillas negras (1’s) que tendrá cada fila de la matriz, esta restricción será comprobada por la función **verificarFilas**, esta función recibe como argumento una solución. Esta función comprobará que la suma de cada elemento de la primera fila sea igual al primer elemento de la primera tripleta, ya que si hay una casilla blanca sumará cero, la suma de cada elemento nos entregará la cantidad de casillas negras. Se hará esto comparando la segunda fila con el segundo elemento de la primera tripleta y la tercera fila con el tercer elemento, si la suma de cada fila es igual a su correspondiente elemento, la función retornara un 1 indicando que la solución cumple esta restricción, sino retornara un 0 indicando que no cumple la restricción.

En caso de que la solución cumpla esta restricción, se procederá a revisar al siguiente elemento de la lista, pero si no la cumple, esta solución será eliminada mediante la función **eliminarSolucion**, esta solución recibe como argumentos la lista de soluciones, la cantidad de soluciones que hay en ella y el índice de la solución a eliminar. La función creará una nueva lista con memoria para una solución menos y copiará todos los elementos de la lista antigua menos la solución que se desea eliminar, finalmente la función retornará la nueva lista de soluciones. Este proceso de comprobación se repetirá hasta haber revisado todas las posibles soluciones.

2) La segunda tripleta indica la cantidad de casillas negras (1's) que tendrá cada columna de la matriz, esta función será comprobada por la función **verificarColumnas** que al igual que la función **verificarFilas** comparará la suma de las filas con su respectivo elemento de la primera tripleta, pero en este caso será la suma de cada elemento de cada columna con su respectivo elemento de la segunda tripleta, además sus retornos serán los mismo, 1 para indicar que la solución cumple la restricción y 0 para indicar que no la cumple.

   En caso de que la solución cumpla la restricción, se procederá a revisar la siguiente solución de la lista de soluciones, pero en caso de no cumplirla, esta solución será eliminada por la función **eliminarSolucion** de la misma forma que en la verificación anterior; repitiendo este proceso hasta revisar todas las posibles soluciones.

Una vez filtradas todas las soluciones, solamente nos quedarán las formas de representar las tripletas entregadas por el usuario. Ahora ya que se desea saber si es que existe más de una manera de representar dichas tripletas, si es que la cantidad de soluciones finales es igual a uno, significa que solo existe una manera de representar estas tripletas, si la cantidad de soluciones es cero, significa que no existe una manera de representar estas tripletas y si la cantidad es mayor a 1, significa que existe más de una manera de representar dichas tripletas.

Criterio de optimización:

Como criterio de optimización tenemos la posibilidad de utilizar las restricciones dadas por las tripletas entregadas por el usuario para "limitar" los ciclos for, de esta manera al limitar la cantidad de ciclos realizados, podemos reducir el tiempo de ejecución y la memoria utilizada para la ejecución de este programa, de esta manera podríamos limitarlos de las manera de que solamente se guarden las soluciones que cumplen todos los requisitos y saber casi de manera inmediata si las tripletas solamente tienen una representación. Debido a que la cantidad de ciclo realizados y la cantidad de posibles soluciones generadas no es muy grande (comparada a otros problemas) (512 posibles soluciones) se guardan todas las posibles soluciones sin filtrar hasta haberlas generado todas.