CASO 1:

El fragmento de código que has proporcionado corresponde al "Caso 1" del algoritmo y se encarga de registrar la entrada de un vehículo en el estacionamiento. Aquí tienes una explicación detallada de cómo funciona este fragmento:

1. \*\*Comprobación de capacidad\*\*:

- El código comienza con una comprobación para verificar si el número de `autosEstacionados` es menor que 100. Esto se hace para asegurarse de que hay espacio disponible en el estacionamiento para registrar un nuevo vehículo.

2. \*\*Registro de entrada\*\*:

- Si la condición es verdadera (es decir, si hay espacio disponible en el estacionamiento), el programa procede con el registro de entrada.

- Incrementa el contador de `autosEstacionados` en 1 para reflejar la entrada del nuevo vehículo.

- Solicita al usuario que ingrese la matrícula del vehículo utilizando la instrucción "Escribir" y luego "Leer" para capturar la entrada del usuario.

- Luego, solicita al usuario que ingrese la hora de ingreso en formato HHMM y la captura en la variable `horaIngreso`.

- Almacena la matrícula y la hora de ingreso en los arreglos `matriculas` y `horasIngreso` en la posición `autosEstacionados`, que es el índice correspondiente al nuevo vehículo.

- Muestra un mensaje de confirmación indicando que la entrada ha sido registrada exitosamente.

3. \*\*Estacionamiento lleno\*\*:

- Si la condición en el primer "Si" no se cumple (es decir, si `autosEstacionados` es igual o mayor que 100), se muestra un mensaje indicando que el estacionamiento está lleno y no se puede registrar la entrada de un nuevo vehículo.

4. \*\*Espacios y mensajes\*\*:

- Después de realizar las operaciones necesarias, se agregan líneas en blanco ("Escribir " ")" para dar formato y dejar espacio entre los mensajes en la salida para que sea más legible.

Este fragmento de código es responsable de registrar la entrada de vehículos en el estacionamiento y asegurarse de que no se supere la capacidad máxima de 100 vehículos. Si el estacionamiento está lleno, se informa al usuario; de lo contrario, se registra la entrada y se actualiza la información de los arreglos correspondientes.

CASO 2:

El fragmento de código que proporcionaste corresponde al "Caso 2" del algoritmo, que se encarga de registrar la salida de un vehículo del estacionamiento y calcular la tarifa a pagar. Aquí tienes una explicación detallada paso a paso de cómo funciona este fragmento:

1. \*\*Comprobación de si hay autos estacionados\*\*:

- El código comienza con una condición que verifica si la variable `autosEstacionados` (la cantidad de vehículos en el estacionamiento) es mayor que 0. Esto se hace para asegurarse de que haya al menos un vehículo estacionado antes de permitir el registro de salida.

2. \*\*Solicitud de matrícula del vehículo que sale\*\*:

- Si hay al menos un vehículo estacionado (autosEstacionados > 0), el programa solicita al usuario que ingrese la matrícula del vehículo que está saliendo.

3. \*\*Búsqueda de la matrícula ingresada\*\*:

- El programa utiliza un bucle "Para" para iterar a través de los vehículos estacionados, desde 1 hasta `autosEstacionados`, para buscar la matrícula ingresada por el usuario en el arreglo `matriculas`.

4. \*\*Registro de salida y cálculo de tarifa\*\*:

- Cuando se encuentra una matrícula coincidente, el programa marca la variable `Encontrado` como Verdadero, lo que indica que se ha encontrado el vehículo.

- Luego, el programa solicita al usuario que ingrese la hora de salida en formato HHMM y la captura en la variable `horaSalida`.

- Se calcula el tiempo que el vehículo estuvo estacionado restando la hora de ingreso (`horasIngreso[i]`) de la hora de salida (`horaSalida`). El resultado se divide por 100 para obtener el tiempo en horas y se almacena en la variable `tiempoEstacionado`.

- Se calcula la tarifa a pagar multiplicando `tiempoEstacionado` por el `precioPorHora` y se almacena en la variable `costo`.

5. \*\*Información de salida y actualización de registros\*\*:

- Se muestra al usuario el tiempo estacionado y la tarifa a pagar.

- Luego, se actualizan los registros en los arreglos `matriculas` y `horasIngreso`. La matrícula del vehículo que sale se reemplaza por la matrícula del último vehículo estacionado (en la posición `autosEstacionados`) para mantener los registros ordenados. Esto también se aplica a las horas de ingreso.

- Se disminuye `autosEstacionados` en 1 para reflejar que un vehículo ha salido del estacionamiento.

6. \*\*Mensajes en caso de no encontrar la matrícula\*\*:

- Si no se encuentra ninguna matrícula coincidente durante la búsqueda en el bucle "Para", se muestra un mensaje indicando que la matrícula no se encontró.

7. \*\*Manejo de casos en los que no hay autos estacionados\*\*:

- Si no hay autos estacionados en el estacionamiento (autosEstacionados = 0), se muestra un mensaje indicando que no hay autos estacionados y, por lo tanto, no se puede registrar la salida.

8. \*\*Espaciado y mensajes\*\*:

- Entre cada paso y mensaje, se incluyen líneas en blanco ("Escribir " ")" para dar formato y dejar espacio en la salida del programa, lo que hace que la información sea más legible y clara para el usuario.

En resumen, este fragmento de código se encarga de manejar la salida de un vehículo del estacionamiento, calcular la tarifa a pagar y actualizar los registros. También maneja casos en los que no hay autos estacionados o si la matrícula ingresada no se encuentra en los registros.

CASO 3:

El fragmento de código que has proporcionado corresponde al "Caso 3" del algoritmo, que se encarga de generar un informe de los autos estacionados en el estacionamiento. Aquí tienes una explicación detallada paso a paso de cómo funciona este fragmento:

1. \*\*Comprobación de si hay autos estacionados\*\*:

- El código comienza con una condición que verifica si la variable `autosEstacionados` (la cantidad de vehículos en el estacionamiento) es mayor que 0. Esto se hace para asegurarse de que haya al menos un vehículo estacionado antes de generar el informe.

2. \*\*Mostrar cantidad de autos estacionados\*\*:

- Si hay al menos un vehículo estacionado (autosEstacionados > 0), el programa muestra la cantidad de autos estacionados utilizando la instrucción "Escribir".

3. \*\*Generación del informe\*\*:

- Luego, el programa utiliza un bucle "Para" para iterar a través de los vehículos estacionados, desde 1 hasta `autosEstacionados`, para generar el informe.

- En cada iteración del bucle, muestra información sobre un vehículo estacionado, incluyendo su número (índice), matrícula y hora de ingreso. Esta información se obtiene de los arreglos `matriculas` y `horasIngreso`.

4. \*\*Espaciado y mensajes\*\*:

- Entre cada vehículo en el informe, se incluyen líneas en blanco ("Escribir " ")" para dar formato y dejar espacio en la salida del programa, lo que hace que el informe sea más legible y claro para el usuario.

5. \*\*Manejo de casos en los que no hay autos estacionados\*\*:

- Si no hay autos estacionados en el estacionamiento (autosEstacionados = 0), se muestra un mensaje indicando que no hay autos estacionados y, por lo tanto, no se puede generar un informe.

En resumen, este fragmento de código se encarga de generar un informe que muestra la cantidad de autos estacionados y la información detallada de cada vehículo, incluyendo su matrícula y hora de ingreso. También maneja casos en los que no hay autos estacionados y muestra un mensaje apropiado en ese caso. El informe resultante proporciona una visión general de los vehículos que están actualmente en el estacionamiento.