CANDIDATO:

PRIMER PARTE

1. Diseñar un algoritmo para resolver la problemática de turnos en el departamento de urgencias de un hospital, las prioridades son las siguientes:

Por color, hay tres colores: rojo, naranja y verde, donde el rojo tiene más prioridad que el naranja y el naranja más que el verde.

Por tiempo de llegada, el primero que llega es el primero que atienden, pero respetando la prioridad por color, es decir, si un naranja llegó antes que un rojo, se atiende primero al rojo.

ALGORITMO

```
SI HAY PACIENTES ESPERANDO
      SE COMPARA EL COLOR DEL PACIENTE QUE VA LLEGANDO CON LOS COLORES DE LOS PACIENTES
EXISTENTES
            SEGÚN EL COLOR SE COMPARA EL TIEMPO DE LLEGADA CON EL RESTO DE PACIENTES
                  Y SE ASIGNA EL NUEVO PACIENTE AL FINAL DEL ÚLTIMO PACIENTE CON DICHO
COLOR
SINO
      PASA INMEDIATAMENTE
```

2. Diseñar un algoritmo que invierta el orden de un arreglo dado con longitud x, tratar de hacerlo con los menos pasos posibles.

Ejemplo de entrada [0,2,4,3,1,5,6,7,8] tendría de salida [8,7,6,5,1,3,4,2,0]

ALGORITMO

```
      PROCESO INVERTIR ARREGLO
      ASIGNAMOS UNA VARIABLE LLAMADA X CON UN VALOR 0
      ASIGNAMOS UNA VARIABLE AUXILIAR
      ASIGNAMOS EL ARREGLO LLAMADOR NÚMEROS

      MIENTRAS ((TAMAÑO DEL ARREGLO – X ) > X)
            A LA VARIABLE AUXILIAR LE ASIGNAMOS EL VALOR ACTUAL DEL ARREGLO EN LA
            POSICIÓN X

            AL VALOR EN EL ARREGLO CON LA POSICIÓN X SE ASIGNA EL VALOR DEL ARREGLO EN LA
            POSICIÓN TAMAÑO DEL ARREGLO MENOS EL VALOR ACTUAL DE LA VARIABLE X

            AL VALOR EN EL ARREGLO CON LA POSICIÓN TAMAÑO DEL ARREGLO –X LE ASIGNAMOS EL
            VALOR DE LA VARIABLE AUXILIAR

            E INCREMENTAMOS EL VALOR DE X EN UNA UNIDAD
      FIN MIENTRAS
```

JS: Explicar brevemente las diferencias entre callback, promesas sin usar el async/await, promesas usando el async/await y observadores.

Todas son distintas formas para resolver el problema de asincronía, la diferencia entre los callbacks y las promesas radica en la limpieza del código ya que los callbacks hacen más anidado el código lo que puede llevarnos a errores en código, cuando la promesa usa el async/await nos permite detener el hilo de ejecución hasta que la promesa se resuelva, en pocas palabras hace síncrono lo asíncrono, también permite hacer distintas llamadas a servicios y guardarlas en variables que podemos utilizar. Para todo esto es muy útil el uso de observadores ya que nos permiten escuchar los eventos o valores emitidos por los observables siendo estos los que van a recibir el valor de las llamadas asíncronas