```
Import random
#Importa el módulo random, que se utilizará para generar números aleatorios.
# Definir parámetros del algoritmo
n = 2
Q = n + 1
#Define los parámetros del algoritmo de planificación de lotería. En este caso, se utiliza un valor de Q igual a n + 1, donde n es igual a 2.
# Definición de procesos
procesos = [{'nombre': 'A', 'duracion': 4, 'llegada': 0, 'prioridad': 0},
        {'nombre': 'B', 'duracion': 8, 'llegada': 3, 'prioridad': 0},
        {'nombre': 'C', 'duracion': 3, 'llegada': 7, 'prioridad': 0},
        {'nombre': 'D', 'duracion': 5, 'llegada': 11, 'prioridad': 0}]
#Define una lista de procesos, cada uno representado como un diccionario con las claves 'nombre', 'duracion', 'llegada' y 'prioridad'. Cada proceso tiene un nombre, una duración (en ticks), un tiempo de llegada (en ticks) y una prioridad (que no se utiliza en este algoritmo).
# Imprimir tabla de carga de trabajo
print("Proceso Duración Llegada")
for proceso in procesos:
   print(f"{proceso['nombre']}      {proceso['duracion']}      {proceso['llegada']}")
#Imprime una tabla que muestra la información de cada proceso. La tabla incluye las columnas 'Proceso', 'Duración' y 'Llegada', y muestra la información de los procesos que se definieron anteriormente.
# Generar procesos aleatorios y agregarlos a la lista
procesos_aleatorios = []
primer_proc = 'E'
for i in range(random.randint(4,8)):
   # Genero los 4 a 8 procesos aleatorios
   procesos_aleatorios.append({'nombre': chr( ord(primer_proc)+i ),
               'llegada': random.randint(0, 10*i),
               'duracion': random.randint(4,10),
               'prioridad': 0
               })
#Genera un número aleatorio entre 4 y 8, y utiliza este número para generar una lista de procesos aleatorios. Cada proceso aleatorio tiene un nombre que comienza con la letra 'E', un tiempo de llegada aleatorio entre 0 y 10 veces su índice en la lista, una duración aleatoria entre 4 y 10 ticks, y una prioridad igual a 0. Los procesos aleatorios se agregan a la lista de procesos definida anteriormente
# Unir listas de procesos
procesos += procesos_aleatorios
#Concatena la lista de procesos aleatorios generada anteriormente con la lista de procesos definida al principio.
# Definimos la variable de tiempo
t = 0
# Definimos una cadena de texto vacía para ir agregando los procesos que se ejecutan
res = ''
# Creamos una copia de la lista de procesos original para ir eliminando los que ya se han ejecutado
```

```
procesos_restantes = list(procesos)
# Obtenemos el número total de procesos
num_procesos = len(procesos_restantes)

# Imprimimos mensaje de inicio
print('* Inicia ejecución')

# Ejecutamos el algoritmo de planificación de lotería mientras queden procesos por ejecutar
while num_procesos > 0:
    # Imprimimos el tiempo actual
    print("t=%d" % t)
    # Generamos un número aleatorio para la lotería
    r = random.randint(1, sum(p['duracion'] for p in procesos_restantes))
    i = 0
    # Recorremos la lista de procesos restantes y vamos restando la duración de cada proceso hasta llegar al número aleatorio generado
    while i < len(procesos_restantes) and r > 0:
        r -= procesos_restantes[i]['duracion']
        i += 1
    # Seleccionamos el proceso que se ejecutará y lo actualizamos
    i -= 1
    proceso_actual = procesos_restantes[i]
    # Agregamos el nombre del proceso a la cadena de texto res
    res += proceso_actual['nombre']
    # Obtenemos la duración de ejecución del proceso (Q o la duración restante del proceso si es menor que Q)
    duracion_ejecucion = min(Q, proceso_actual['duracion'])
    # Actualizamos la duración restante del proceso
    proceso_actual['duracion'] -= duracion_ejecucion
    # Si la duración restante del proceso es 0, lo eliminamos de la lista de procesos restantes y actualizamos el número de procesos restantes
    if proceso_actual['duracion'] == 0:
        num_procesos -= 1
        procesos_restantes.pop(i)
    else:
        # Si la duración restante del proceso es mayor que 0, lo actualizamos en la lista de procesos restantes
        procesos_restantes[i] = proceso_actual
    # Actualizamos el tiempo actual sumando la duración de ejecución del proceso
    t += duracion_ejecucion
    # Imprimimos un mensaje indicando el proceso que se ejecutó y su duración de ejecución
    print("    ⌚ %s %d tick" % (proceso_actual['nombre'], duracion_ejecucion))

# Imprimimos la cadena de texto res con los nombres de los procesos ejecutados en orden
print("Planificación realizada: \n" + res)
# Imprimimos la duración total de ejecución
print("\n\nDuración total: %d" % t)
```

```
# Ordenamos la lista de procesos por orden de ejecución (orden en el que aparecen en la cadena de texto res)
procesos_ordenados = sorted(procesos, key=lambda x: res.index(x['nombre']))

# Imprimir la tabla de forma ordenada
print("{:<10} {:<10} {:<10} {:<10} {:<10} {:<10}".format("Proceso", "Inicio", "Fin", "T", "E", "P"))

#En esta línea estamos imprimiendo la cabecera de la tabla de información detallada de cada proceso. Usamos la función print y la función format para dar formato a la cabecera de la tabla. Establecemos un ancho máximo de 10 caracteres para cada columna y usamos las letras "P", "T" y "E" para representar "Proporción", "Tiempo de ejecución" y "Tiempo de espera", respectivamente.

# Inicializamos variables para los tiempos totales
tiempo_espera_total = 0
tiempo_respuesta_total = 0

# Iteramos sobre todos los procesos y calculamos la información detallada de cada uno
for proceso in procesos:
    # Obtenemos el índice de inicio del proceso en la cadena de ejecución
    inicio = res.index(proceso['nombre'])

    # Calculamos el tiempo de finalización del proceso sumando la duración al inicio
    fin = inicio + proceso['duracion']

    # Calculamos el tiempo de respuesta del proceso restando el tiempo de llegada al tiempo de finalización
    t = fin - proceso['llegada']

    # Calculamos el tiempo de espera del proceso restando la duración al tiempo de respuesta
    e = t - proceso['duracion']

    # Calculamos la proporción de tiempo de espera respecto a la duración del proceso
    if proceso['duracion'] == 0:
        p = 0
    else:
        p = e / proceso['duracion']

    # Sumamos los tiempos de espera y respuesta a los totales
    tiempo_espera_total += e
    tiempo_respuesta_total += t

    # Imprimimos la información detallada del proceso en forma de tabla
    print("{:<10} {:<10} {:<10} {:<10} {:<10} {:<10.1f}".format(proceso['nombre'], inicio, fin, t, e, p))

# Calculamos el promedio de espera y respuesta
```

```
promedio_espera = tiempo_espera_total / len(procesos)
promedio_respuesta = tiempo_respuesta_total / len(procesos)

# Imprimimos una línea adicional para mostrar los promedios
print("{:<10} {:<10} {:<10} {:<10} {:<10} {:<10}".format("Prom", "-", "-",
"{:.1f}".format(promedio_respuesta), "{:.1f}".format(promedio_espera), "-"))
```

#En la primera línea, calculamos el tiempo de espera promedio dividiendo el tiempo de espera total por el número de procesos. De manera similar, en la segunda línea, calculamos el tiempo de respuesta promedio dividiendo el tiempo de respuesta total por el número de procesos.

#En la última línea, imprimimos una línea adicional en la tabla para mostrar los promedios. Usando la misma sintaxis que antes, usamos "{:<10}" para imprimir "Prom" en la primera columna, y "-" en las tres siguientes columnas ya que no son relevantes para el promedio. Luego, utilizamos "{:.1f}" para imprimir el promedio de respuesta y el promedio de espera en las columnas correspondientes. Finalmente, utilizamos "-" para la última columna ya que no es relevante para los promedios.