```python
#2022847 Phil Chiang
#2039152 Armin Ebrahimihardoroodi
#2087505 Timo Klabbers
#2036313 Dahli Koskamp
#2025574 Madelieve Pigmans

def arrival_service_uniform(N, Arrival, Service):
    return ([1/Arrival for n in [1]*N], [1/Service for n in [1]*N])


def calculate_arrival_service_per_C(arrivalrates, servicerates):
    last_arrival = 0
    last_service_start = 0
    last_service_end = 0

    arrival_times = []
    service_start = []
    service_finish = []

    for n in range(len(arrivalrates)):
        arrival_time = arrivalrates[n] + last_arrival
        last_arrival = arrival_time
        arrival_times.append(arrival_time)
        if last_service_end <= last_arrival:
            last_service_start = last_arrival
        else:
            last_service_start = last_service_end
        last_service_end = last_service_start + servicerates[n]
        service_start.append(last_service_start)
        service_finish.append(last_service_end)
    return (arrival_times, service_start, service_finish)


def arrival_service_exponential(N, Arrival, Service):
    import numpy as np
    np.random.seed(1)
    arrival_time = []
    service_time = []
    for x in range(N):
        arrival_time.append(np.random.exponential(1 / Arrival))
        service_time.append(np.random.exponential(1 / Service))
    return (arrival_time, service_time)


def Total_system_time(arrivaltimes, service_finish):
    waiting_time = []
    for x in range(len(arrivaltimes)):
        waiting_time.append(service_finish[x] - arrivaltimes[x])
    return (waiting_time, sum(waiting_time))


def Total_queue_time(arrivaltimes, service_start):
    queue_time = []
    for n in range(len(arrivaltimes)):
        queue_time.append(service_start[n] - arrivaltimes[n])
    return (queue_time, sum(queue_time))


def Total_system_time_CSV(file):
    import csv
    from pandas import pandas as pd

    tableDF=pd.read_csv(file, header=0)

    waiting_time=[]
    for index, row in tableDF.iterrows():
        waiting_time.append(row[1] - row[0])
    return(waiting_time, sum(waiting_time))


def Queue_length_someone_joins(arrivaltimes, service_finish):
    result = []
    for n in range(len(arrivaltimes)):
        result.append(n - sum(f < arrivaltimes[n] for f in service_finish))
    return result


def Queueremains (queuelength, C, Q):
    bounced = sum(n > Q for n in queuelength)
    return (bounced, "$" + str(C * bounced))


def QueueLeaves(arrivalrates, servicerates, C, Q):
    import numpy as np
    arrival_times = np.cumsum(np.array(arrivalrates))

    c_in_queue = []
    end_time = 0
    start_service=0
    queue = []

    for n in range(len(arrivalrates)):
        if arrival_times[n] >= end_time:
            start_service = arrival_times[n]
        else:
            queue.append(n)
        if len(queue) > Q:
            servicerates[n] = 0
            queue.remove(n)
        c_in_queue.append(len(queue))
        end_time = start_service + servicerates[n]
    bounced = sum(service_time == 0 for service_time in servicerates)

    return (c_in_queue, bounced, str("$" + str(bounced * C)))
```