```
from model import *


def WastedTime(carX,carY,startX,startY,CurrentTime,EarliestTime):
    distance_to_start = abs(carX-startX) + abs(carY-startY)
    wasted_time = EarliestTime - CurrentTime - distance_to_start
    return wasted_time



def IsRidePossible(carX,carY,startX,startY,endX,endY,TimeLatest,CurrentTime):
    distance_to_start = abs(carX-startX) + abs(carY-startY)
    distance_of_ride = abs(startX - endX) + abs(startY - endY)
    time_until_end_of_ride = TimeLatest - CurrentTime
    if time_until_end_of_ride < 0:return False
    elif distance_to_start + distance_of_ride <= time_until_end_of_ride: return True
    return False

def ClosestNextRide(endX,endY,rides):
    avis =[]
    for ride in rides:
        if ride.available == True:
            avis.append(abs(endX - ride.start_pos.row) + abs(endY - ride.start_pos.colum
    return min(avis)



def CalculatePoints(startX,startY,endX,endY,carX,carY,CurrentTime,EarliestTime,bo
    distance_of_ride = abs(startX - endX) + abs(startY - endY)
    wastedTime = WastedTime(carX,carY,startX,startY,CurrentTime,EarliestTime)
    distance_to_ride = abs(carX-startX) + abs(startY-carY)
    Bonus = 0
    if wastedTime >= 0: Bonus = bonus
    if wastedTime <0: wastedTime = 0
    denom = wastedTime + 1 +distance_to_ride * 2.5 + ride_wage * 7
    #print('wasted:',wastedTime,'distance:',distance_of_ride,'wage: ', ride_wage,'id:',id,
    wage = (distance_of_ride*10 + Bonus*5) / denom
    return wage, distance_of_ride+Bonus, CurrentTime + distance_of_ride +wastedTim

def CheckIfAllRidesTaken(rides):
    availablerides =[]
    for ride in rides:
        if ride.available == True:
            availablerides.append(ride.id)
    print(availablerides)
    if len(availablerides) == 0: return True
    else: return False

def GetAllAvailableVehicles(vehicles,step):
    available = []
    for vehicle in vehicles:
        if vehicle.unavailable_until <= step and vehicle.never_possible == False:
            available.append(vehicle.id)
    return available
```

```python
def GetDateOfNextAvailable(vehicles):
    dates = []
    for vehicle in vehicles:
        if vehicle.never_possible == False:
            dates.append(vehicle.unavailable_until)
    return min(dates, default = None)
```