public with sharing class MaintenanceRequestHelper {  
    public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case> nonUpdCaseMap) {  
        Set<Id> validIds = new Set<Id>();  
        For (Case c : updWorkOrders){  
            if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){  
                if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){  
                    validIds.add(c.Id);  
                }  
            }  
        }  
          
        //When an existing maintenance request of type Repair or Routine Maintenance is closed,   
        //create a new maintenance request for a future routine checkup.  
        if (!validIds.isEmpty()){  
            Map<Id,Case> closedCases = new Map<Id,Case>([SELECT Id, Vehicle\_\_c, Equipment\_\_c, Equipment\_\_r.Maintenance\_Cycle\_\_c,  
                                                          (SELECT Id,Equipment\_\_c,Quantity\_\_c FROM Equipment\_Maintenance\_Items\_\_r)   
                                                          FROM Case WHERE Id IN :validIds]);  
            Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();  
              
            //calculate the maintenance request due dates by using the maintenance cycle defined on the related equipment records.   
            AggregateResult[] results = [SELECT Maintenance\_Request\_\_c,   
                                         MIN(Equipment\_\_r.Maintenance\_Cycle\_\_c)cycle   
                                         FROM Equipment\_Maintenance\_Item\_\_c   
                                         WHERE Maintenance\_Request\_\_c IN :ValidIds GROUP BY Maintenance\_Request\_\_c];  
              
            for (AggregateResult ar : results){   
                maintenanceCycles.put((Id) ar.get('Maintenance\_Request\_\_c'), (Decimal) ar.get('cycle'));  
            }  
              
            List<Case> newCases = new List<Case>();  
            for(Case cc : closedCases.values()){  
                Case nc = new Case (  
                    ParentId = cc.Id,  
                    Status = 'New',  
                    Subject = 'Routine Maintenance',  
                    Type = 'Routine Maintenance',  
                    Vehicle\_\_c = cc.Vehicle\_\_c,  
                    Equipment\_\_c =cc.Equipment\_\_c,  
                    Origin = 'Web',  
                    Date\_Reported\_\_c = Date.Today()   
                );  
                  
                //If multiple pieces of equipment are used in the maintenance request,   
                //define the due date by applying the shortest maintenance cycle to today’s date.  
                If (maintenanceCycles.containskey(cc.Id)){  
                    nc.Date\_Due\_\_c = Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));  
                } else {  
                    nc.Date\_Due\_\_c = Date.today().addDays((Integer) cc.Equipment\_\_r.maintenance\_Cycle\_\_c);  
                }  
                  
                newCases.add(nc);  
            }  
              
            insert newCases;  
              
            List<Equipment\_Maintenance\_Item\_\_c> clonedList = new List<Equipment\_Maintenance\_Item\_\_c>();  
            for (Case nc : newCases){  
                for (Equipment\_Maintenance\_Item\_\_c clonedListItem : closedCases.get(nc.ParentId).Equipment\_Maintenance\_Items\_\_r){  
                    Equipment\_Maintenance\_Item\_\_c item = clonedListItem.clone();  
                    item.Maintenance\_Request\_\_c = nc.Id;  
                    clonedList.add(item);  
                }  
            }  
            insert clonedList;  
        }  
    }  
}