```
# ===================================== #
# TRI FUSION                            #
# Etude de la complexité moyenne        #
# Version du 25/11/2015                 #
# ===================================== #

# Importations
# ============
from random import randint
from time import perf_counter
from math import log

# La fonction de fusion
# =====================
# Cette fonction reçoit en argument deux listes TRIEES L1 et L2 et renvoie
# une liste L triée contenant tous les éléments de L1 et L2.
def fusion(L1,L2):
    global c
    # Initialisations
    L = []
    # On vide une des deux listes (mais on ne sait pas quelle liste sera vidée
    # la première !)
    while L1 and L2:
        c += 1
        if L1[0] < L2[0]:
            L.append(L1.pop(0))
        else:
            L.append(L2.pop(0))
    # On renvoie la concaténation de L, L1 et L2 sachant que L1=[] ou L2=[]
    return L + L1 + L2

# La fonction de tri
# ==================
def tri_fusion(L):
    l = len(L)
    if l <= 1:
        return L
    else:
        m = l//2
        return fusion(tri_fusion(L[:m]),tri_fusion(L[m:]))

# Saisie des paramètres et initialisations.
# La variable c pour compter le nombre de comparaisons effectuées
# ================================================================
c = 0
ntris = int(input('Combien de tris souhaitez-vous ? '))
n = int(input('\nLongueur des listes à trier ? '))
valmax = int(input('\nValeur maxi des nombres générés ? '))
t_ellapse_total = 0

for i in range(ntris):
    LNA = [randint(0,valmax) for i in range(n)]
    t_start = perf_counter()
    LT = tri_fusion(LNA)
    t_end = perf_counter()
    t_ellapse_total += t_end - t_start

# Affichage de la durée moyenne du tri et du nombre moyen de comparaisons divisé
# par nxln(n) qui doit être compris entre 1/(2*ln(2)) et 2/ln(2).
# ============================================================================
print('\nDurée moyenne du tri (secondes) : ',t_ellapse_total/ntris)
print('\nNombre total de comparaisons : ',c)
print('\nComparaisons : ',c/(ntris*n*log(n)))

# ===============
# FIN DU PROGRAMME
```