```
## this code reads two datasets that we are using to compare two medians
## calculates the test statistic T for MWW test
## does NOT compute p value

dataset_A = as.matrix(read.table("folderpath/filename.extension", header = T
 or F))
dataset_B = as.matrix(read.table("folderpath/filename.extension", header = T
 or F))

n_A = length(dataset_A)
n_B = length(dataset_B)

## Now find which dataset has more observations
## will be necessary to determine the labels

## remember to relabel the dataset with smaller sample size as from
 "Population 1"

if (n_A <= n_B)
{
    dataset_01 = dataset_A ## smaller data size coresponds to Population 1
    dataset_02 = dataset_B  ## larger data size coresponds to Population 2
}else
{
    dataset_01 = dataset_B ## smaller data size coresponds to Population 1
    dataset_02 = dataset_A ## larger data size coresponds to Population 2
}
n1 = length(dataset_01)
n2 = length(dataset_02)

print(paste("Sample size for dataset from Population 1 is = ",n1,sep=""))
print(paste("Sample size for dataset from Population 2 is = ",n2,sep=""))

## now construct the table to compute the test statistics

## now pool the data, with dataset_01 first, then dataset_02
data_pooled =  c(dataset_01,dataset_02)

## initiate construction of the indicator column (2nd column of the table)
## since we have not sorted the pooled data yet, first n1 positions correspond
 to dataset 01
## next n2 positions correspond to dataset 02
## "YES" for positions corresponding to dataset 01, "NO" for positions
 corresponding to dataset 02
unsorted_indicator = c(rep("Yes",n1),rep("No",n2))

## now sort the pooled data from smallest to largest
data_pooled_sorted = sort(data_pooled)

##  rearrange the indicator column depending the order of the pooled data
sorted_indicator = unsorted_indicator[order(data_pooled)]
```

```
## the 3rd column has all ranks from 1 to (n1+n2)
all_ranks = 1:(n1+n2)

## combine the columns
output = data.frame( sorted_values = data_pooled_sorted,
 Indicator_1st_population = sorted_indicator, all_ranks = all_ranks )

## print it

print("-----------------------------------------------------")
print(output,row.names=F)
print("-----------------------------------------------------")

## compute the test statistic by summing ranks corresponding to "YES" in
 indicator column
T = sum(all_ranks[(sorted_indicator== "Yes")])

print(paste("Test Statistic T = ", T, sep=""))

## Now use this value of T to find range for the p-value
## following the guidelines specified in the classnote
```