```r
## this code reads a bivariate dataset on two variables X and Y
## performs simple linear regression of Y on X
## using Kendall-Theil (KT) regression line
## Also, computes the test statistic T to perform
## KT test for significance of regression
## does NOT compute p value for significance of regression

## this code asumes there are no ties in X values
## if this assumption is not true, this code will result in error

## read the data file that includes observations from two variables in
 two-column format

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

## Input the positions of covariate column and response column
## as understood from the statement of the problem
covariate_column_position =   ## 1 or 2 depending on which column is covariate
response_column_position =  ## 1 or 2 depending on which column is response

## Accordingly, define the original (unsorted) X and Y values)
X_original = dataset[ ,covariate_column_position]
Y_original = dataset[ ,response_column_position]

## now pair them
Pairs_original = cbind(X_original,Y_original)

## calculate sample size
n = length(X_original)
print(paste("Sample size = n = ",n,sep=""))

## Now sort the pairs according to values of X ONLY from smallest to largest
Pairs_sorted_using_X_original_only =  Pairs_original[order(X_original), ]

## now create the pairwise slopes

## extract X and Y from sorted pairs
X = Pairs_sorted_using_X_original_only[ ,1]
Y = Pairs_sorted_using_X_original_only[ ,2]

## initiate the empty matrix of pairwise slopes
slope_matrix = matrix(" ",n,n)

## fill in only the positions above the diagonal line
for (i in 1:(n-1)) ## i-th row
{
    for (j in (i+1):n) ## j-th column
    {
        ##j > i implies (i,j) position is above the diagonal
        slope_matrix[i,j] = round((Y[j] - Y[i])/(X[j] - X[i]),2)
```

```
    }
}

## want to use the sorted pairs as row headers as well as column headers of
## the matrix of pairwise slopes

## initiate empty name fields for columns
colnames(slope_matrix) = array("",n)
## allocate sorted pairs as column names
for (i in 1:n) colnames(slope_matrix)[i] = paste("(",X[i],",",Y[i],")",sep="")
## do the same for row names as well
rownames(slope_matrix) = colnames(slope_matrix)

## Now print the matrix of pairwise slopes
print("-------------------------------")
print("Matrix of pairwise slopes")
print("--------------------------------")
print(noquote(slope_matrix))
print("--------------------------------")

## Now, to estimate the slope b, extract the values of
##pairwise slopes from above matrix
collection_of_pairwise_slopes =
 as.numeric(slope_matrix[!is.na(as.numeric(slope_matrix))])

## calculate the median of pairwise slopes
b_hat = median(collection_of_pairwise_slopes)

print(paste("Estimated slope = b_hat = ",b_hat,sep=""))

## now, to estimate intercept a
## calculate a1,a2,...,an, as in classnote
a_vector = round(Y - c(b_hat)*X,2)

## compute the median of a1,a2,...,an
a_hat = median(a_vector)

print(paste("Estimated intercept = a_hat = ",a_hat,sep=""))

## plot the estimated regression line and the data points
x11()
## plot the observations first
plot(X,Y,type="p",col="RED",xlab
 =colnames(dataset)[covariate_column_position],
 ylab=colnames(dataset)[response_column_position], main="")
## now draw a straight line with intercept a_hat and slope b_hat
abline(a=a_hat,b=b_hat,col="BLUE")

## prediction for a new observation

## specify the covariate value for the new observation as X_new
```

```
X_new =

# Predict the median of response value for the new observation
## using formula for estimated regression line
median_Y_new_prediction = a_hat + b_hat*X_new

print(paste("Predicted median of the distribution of Y for the new observation
 = ",median_Y_new_prediction,sep=""))

## Now, test for significance of regression

## the test statistic is difference between
## number of positive pairwise slopes and
## number of negative pairwise slopes
Number_of_positive_pairwise_slopes =
 length(which(collection_of_pairwise_slopes>0))
Number_of_negative_pairwise_slopes =
 length(which(collection_of_pairwise_slopes<0))

T = Number_of_positive_pairwise_slopes - Number_of_negative_pairwise_slopes

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

## now, compute p value using appropriate probability tables
## as mentioned in class note
```