```
#GENERATING DEGREE SEQUENCE
random_degree_sequence <- function(n) {
  seq <- floor(c(runif(1:n,0,n)))
  while (!is_graphical(seq)) { #making sure that we obtain a graphical sequence
    seq <- sample(1:(n-1), n, replace = TRUE)
  }
  return(seq)
}

#GRAPHICALITY CHECK
is_graphical <- function(seq) {
  while (TRUE) {
    seq <- sort(seq, decreasing = TRUE)
    if (all(seq == 0)) {
      return(TRUE)
    }
    if (any(seq < 0) || any(is.na(seq))) {
      return(FALSE)
    }
    k <- seq[1]
    seq <- seq[-1]
    n <- length(seq)
    if (k > n) {
      return(FALSE)
    }
    seq[1:k] <- seq[1:k] - 1
  }
}

#CONNECTIVITY CHECK
is_connected <- function(adj_matrix) {
  start<-1
  n <- nrow(adj_matrix)
  visited <- rep(FALSE, n) #This vector keeps track of whether each vertex has been visited during the search for connected components.
  queue <- c()
  visited[start] <- TRUE
  queue<-c(queue,start)

  while (length(queue) > 0) {
    current <- queue[1]
    queue <- queue[-1]
    neighbors <- which(adj_matrix[current, ] == 1)
    for (neighbor in neighbors) {
      if (!visited[neighbor]) {
        visited[neighbor] <- TRUE
        queue <- c(queue, neighbor)
      }
    }
  }
  return(all(visited))
}

#EDGE SWAPS FOR CONNECTIVITY
swap_edges <- function(adj_matrix, niter = 100) {
  n <- nrow(adj_matrix)
  number_swaps <- 0
  for (iter in 1:niter) {
    edges <- which(adj_matrix == 1, arr.ind = TRUE)
    e1 <- edges[sample(nrow(edges), 1), ]
    e2 <- edges[sample(nrow(edges), 1), ]
    max_attempts <- 100 #to avoid infinite loop
    for (attempt in 1:max_attempts) {
      e2 <- edges[sample(nrow(edges), 1), ]
      if (!any(is.na(e1)) && !any(is.na(e2)) && e1[1] != e2[1] && e1[1] != e2[2] && e1[2] != e2[1] && e1[2] != e2[2]) {
        break
      }
    }

    if (adj_matrix[e1[1], e2[1]] == 0 && adj_matrix[e1[2], e2[2]] == 0) {
      number_swaps <- number_swaps+1
      #removing original edges
      adj_matrix[e1[1], e1[2]] <- 0
      adj_matrix[e1[2], e1[1]] <- 0
      adj_matrix[e2[1], e2[2]] <- 0
      adj_matrix[e2[2], e2[1]] <- 0

      #adding new edges
      adj_matrix[e1[1], e2[1]] <- 1
      adj_matrix[e2[1], e1[1]] <- 1
      adj_matrix[e1[2], e2[2]] <- 1
      adj_matrix[e2[2], e1[2]] <- 1
    }
  }

  return(number_swaps)
}

#First Algorithm: Random Vertex Selection, distributed to the Highest Degree.
random_vertex_highest_degree <- function(n,m,input_index,output_index) {
  directory <- "C:\\Users\\fatma\\Desktop\\IE456_Project"
  algorithm_name <- "random_vertex_highest_degree"
  set.seed(m)
  degree_seq <- random_degree_sequence(n)
  write(degree_seq,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,".txt"), sep=" ",ncolumns=length(degree_seq))
  start_time <- Sys.time() #measuring the time taken to generate a graph
  output_list <- c() #collecting to  the output
  n <- length(degree_seq)
  adj_matrix <- matrix(0, nrow = n, ncol = n) #building the adjacency matrix

  while (sum(degree_seq) > 0) {
    idx <- sample(n, 1) #selecting random vertex
    if (degree_seq[idx] == 0) {
      next
    }
```

```
    indices <- order(degree_seq, decreasing = TRUE) #ranking the degree sequence in decreasing order
    target_vertices <- indices[degree_seq[indices] > 0]
    target_vertices <- target_vertices[target_vertices != idx]
    target_vertices <- target_vertices[1:degree_seq[idx]]

    for (i in 1:length(target_vertices)) {
      adj_matrix[idx, target_vertices[i]] <- 1 #1 in the adjacency matrix meaning that there is a connection between the corresponding vertices
      adj_matrix[target_vertices[i], idx] <- 1 #the connection is bidirectional
    }
    degree_seq[idx] <- 0
    degree_seq[target_vertices] <- degree_seq[target_vertices] - 1
  }
  connectivity <- is_connected(adj_matrix)
  if (!is_connected(adj_matrix)) {
    num <- swap_edges(adj_matrix, niter=n*(n-1)/2)
  }
  end_time <- Sys.time()
  elapsed_time <- end_time - start_time
  output_list <- append(output_list,c(connectivity, num,elapsed_time))
  write(output_list,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,"-Output-",output_index,"-",algorithm_name,
".txt"),sep="\n",ncolumns=length(output_list))
  for(i in 1:ncol(adj_matrix)){
    neighbors_list<-c()
    for(j in 1:nrow(adj_matrix)){
      if(adj_matrix[i,j]==1){
        neighbors_list<-append(neighbors_list,j)
      }
    }
    write(neighbors_list,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,"-Output-",output_index,"-",algorithm_name, ".txt"),
append=TRUE,sep=" ")
  }
  return(output_list)
}

#Second Algorithm: Highest Vertex Selection, distributed to the Highest Degree.
highest_vertex_highest_degree <- function(n,m,input_index,output_index) {
  directory <- "C:\\Users\\fatma\\Desktop\\IE456_Project"
  algorithm_name <- "highest_vertex_highest_degree"
  set.seed(m)
  degree_seq <- random_degree_sequence(n)
  write(degree_seq,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,".txt"), sep=" ",ncolumns=length(degree_seq))
  start_time <- Sys.time() #measuring the time taken to generate a graph
  output_list <- c() #collecting to  the output
  n <- length(degree_seq)
  adj_matrix <- matrix(0, nrow = n, ncol = n) #building the adjacency matrix

  while (sum(degree_seq) > 0) {
    idx <- which.max(degree_seq)  # Select vertex with the highest degree
    if (degree_seq[idx] == 0) {
      next
    }

    indices <- order(degree_seq, decreasing = TRUE)
    target_vertices <- indices[degree_seq[indices] > 0]
    target_vertices <- target_vertices[target_vertices != idx]
    target_vertices <- target_vertices[1:degree_seq[idx]]

    for (i in 1:length(target_vertices)) {
      adj_matrix[idx, target_vertices[i]] <- 1
      adj_matrix[target_vertices[i], idx] <- 1
    }
    degree_seq[idx] <- 0
    degree_seq[target_vertices] <- degree_seq[target_vertices] - 1
  }
  connectivity <- is_connected(adj_matrix)
  if (!is_connected(adj_matrix)) {
    num <- swap_edges(adj_matrix, niter=n*(n-1)/2)
  }
  end_time <- Sys.time()
  elapsed_time <- end_time - start_time
  output_list <- append(output_list,c(connectivity, num,elapsed_time))
  write(output_list,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,"-Output-",output_index,"-",algorithm_name,
".txt"),sep="\n",ncolumns=length(output_list))
  for(i in 1:ncol(adj_matrix)){
    neighbors_list<-c()
    for(j in 1:nrow(adj_matrix)){
      if(adj_matrix[i,j]==1){
        neighbors_list<-append(neighbors_list,j)
      }
    }
    write(neighbors_list,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,"-Output-",output_index,"-",algorithm_name, ".txt"),
append=TRUE,sep=" ")
  }
  return(output_list)
}

#Third Algorithm: Smallest Vertex Selection, distributed to the Highest Degree.
smallest_vertex_highest_degree <- function(n,m,input_index,output_index) {
  directory <- "C:\\Users\\fatma\\Desktop\\IE456_Project"
  algorithm_name <- "smallest_vertex_highest_degree"
  set.seed(m)
  degree_seq <- random_degree_sequence(n)
  write(degree_seq,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,".txt"), sep=" ",ncolumns=length(degree_seq))
  start_time <- Sys.time() #measuring the time taken to generate a graph
  output_list <- c() #collecting to  the output
  n <- length(degree_seq)
  adj_matrix <- matrix(0, nrow = n, ncol = n) #building the adjacency matrix

  while (sum(degree_seq) > 0) {
    idx <- which(degree_seq == min(degree_seq[degree_seq > 0]))[1]  #selecting vertex with the smallest degree
    indices <- order(degree_seq, decreasing = TRUE)
    target_vertices <- indices[degree_seq[indices] > 0]
    target_vertices <- target_vertices[target_vertices != idx]
    target_vertices <- target_vertices[1:degree_seq[idx]]

    for (i in 1:length(target_vertices)) {
```

```
      adj_matrix[idx, target_vertices[i]] <- 1
      adj_matrix[target_vertices[i], idx] <- 1
    }
    degree_seq[idx] <- 0
    degree_seq[target_vertices] <- degree_seq[target_vertices] - 1
  }
  connectivity <- is_connected(adj_matrix)
  if (!is_connected(adj_matrix)) {
    num <- swap_edges(adj_matrix, niter=n*(n-1)/2)
  }
  end_time <- Sys.time()
  elapsed_time <- end_time - start_time
  output_list <- append(output_list,c(connectivity, num,elapsed_time))
  write(output_list,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,"-Output-",output_index,"-",algorithm_name,
".txt"),sep="\n",ncolumns=length(output_list))
  for(i in 1:ncol(adj_matrix)){
    neighbors_list<-c()
    for(j in 1:nrow(adj_matrix)){
      if(adj_matrix[i,j]==1){
        neighbors_list<-append(neighbors_list,j)
      }
    }
    write(neighbors_list,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,"-Output-",output_index,"-",algorithm_name, ".txt"),
append=TRUE,sep=" ")
  }
  return(output_list)
}

#Fourth Algorithm: Smallest Vertex Selection, distributed to the Smallest Degree.
smallest_vertex_smallest_degree <- function(n,m,input_index,output_index) {
  directory <- "C:\\Users\\fatma\\Desktop\\IE456_Project"
  algorithm_name <- "smallest_vertex_smallest_degree"
  set.seed(m)
  degree_seq <- random_degree_sequence(n)
  write(degree_seq,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,".txt"), sep=" ",ncolumns=length(degree_seq))
  start_time <- Sys.time() #measuring the time taken to generate a graph
  output_list <- c() #collecting to  the output
  n <- length(degree_seq)
  adj_matrix <- matrix(0, nrow = n, ncol = n) #building the adjacency matrix

  while (sum(degree_seq) > 0) {
    idx <- which(degree_seq == min(degree_seq[degree_seq > 0]))[1]  # Select vertex with the smallest degree
    target_vertices <- which(degree_seq > 0)
    target_vertices <- target_vertices[target_vertices != idx]
    target_vertices <- target_vertices[order(degree_seq[target_vertices])]

    if (length(target_vertices) == 1) {
      if (degree_seq[idx] == degree_seq[target_vertices]) {
        adj_matrix[idx, target_vertices] <- 1
        adj_matrix[target_vertices, idx] <- 1
        degree_seq[idx] <- 0
        degree_seq[target_vertices] <- 0
      } else {
        return(NULL)
      }
    } else {
      target_vertices <- target_vertices[1:min(length(target_vertices), degree_seq[idx])]

      for (i in 1:length(target_vertices)) {
        adj_matrix[idx, target_vertices[i]] <- 1
        adj_matrix[target_vertices[i], idx] <- 1
      }

      degree_seq[target_vertices] <- degree_seq[target_vertices] - 1
      degree_seq[idx] <- 0
    }
  }
  connectivity <- is_connected(adj_matrix)
  if (!is_connected(adj_matrix)) {
    num <- swap_edges(adj_matrix, niter=n*(n-1)/2)
  }
  end_time <- Sys.time()
  elapsed_time <- end_time - start_time
  output_list <- append(output_list,c(connectivity, num,elapsed_time))
  write(output_list,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,"-Output-",output_index,"-",algorithm_name,
".txt"),sep="\n",ncolumns=length(output_list))
  for(i in 1:ncol(adj_matrix)){
    neighbors_list<-c()
    for(j in 1:nrow(adj_matrix)){
      if(adj_matrix[i,j]==1){
        neighbors_list<-append(neighbors_list,j)
      }
    }
    write(neighbors_list,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,"-Output-",output_index,"-",algorithm_name, ".txt"),
append=TRUE,sep=" ")
  }

  if (any(is.na(output_list))==TRUE){
    print("Could not generate a connected graph.")
  }
  else{
    return(output_list)
  }
}

#Fifth Algorithm: The Sequential Algorithm
the_sequential_algorithm <- function(n,m,input_index, output_index){
  directory <- "C:\\Users\\fatma\\Desktop\\IE456_Project"
  algorithm_name <- "the_sequential_algorithm"
  set.seed(m)
  degree_seq <- random_degree_sequence(n)
  write(degree_seq,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,".txt"), sep=" ",ncolumns=length(degree_seq))
  start_time <- Sys.time()
  n <- length(degree_seq)
  index_list <- c(1:n)
  chosen_list <- c()
```

```
  remaining_list <- c(1:n)
  list_of_edges <- c()
  output_list <- c() #collecting to  the output
  adj_matrix <- matrix(0, nrow = n, ncol = n) #building the adjacency matrix
  while(!all(degree_seq==0)){
    vec_no_zeros <- degree_seq[degree_seq!=0]
    min_index_no_zeros <- which.min(vec_no_zeros)
    min_index <- min(which(degree_seq==vec_no_zeros[min_index_no_zeros]))
    chosen_list <- c(chosen_list,min_index)
    remaining_list <- remaining_list[-which(remaining_list==min_index)]
    while(!degree_seq[min_index]==0){
      candidate_j=c()
      for(j in 1:length(remaining_list)){
        degree_seq_trial = degree_seq
        degree_seq_trial[min_index] = degree_seq[min_index] - 1
        degree_seq_trial[remaining_list[j]] = degree_seq_trial[remaining_list[j]] - 1
        if(is_graphical(degree_seq_trial)==TRUE){
          candidate_j=c(candidate_j,remaining_list[j])
        }
      }
      probs <- candidate_j/sum(candidate_j)
      random_j <- sample(length(candidate_j),1,prob=probs)
      degree_seq[min_index] = degree_seq[min_index] - 1
      degree_seq[candidate_j[random_j]] = degree_seq[candidate_j[random_j]] - 1
      adj_matrix[min_index, candidate_j[random_j]] <- 1
      adj_matrix[candidate_j[random_j], min_index] <- 1
    }
  }
  connectivity <- is_connected(adj_matrix)
  num<-0
  if (!is_connected(adj_matrix)) {
    num <- swap_edges(adj_matrix, niter=n*(n-1)/2)
  }
  end_time <- Sys.time()
  elapsed_time <- end_time - start_time
  output_list <- append(output_list,c(connectivity, num,elapsed_time))
  write(output_list,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,"-Output-",output_index,"-",algorithm_name,
".txt"),sep="\n",ncolumns=length(output_list))
  for(i in 1:ncol(adj_matrix)){
    neighbors_list<-c()
    for(j in 1:nrow(adj_matrix)){
      if(adj_matrix[i,j]==1){
        neighbors_list<-append(neighbors_list,j)
      }
    }
    write(neighbors_list,file=paste0(directory,"\\Group7-",n,"-",m,"-","Input-",input_index,"-Output-",output_index,"-",algorithm_name, ".txt"),
append=TRUE,sep=" ")
  }
  return(output_list)
}


#RESULTS
input_index <- 1
output_index <- 1
random_vertex_highest_degree(10,50,input_index,output_index)
output_index<-output_index+1
highest_vertex_highest_degree(10,50,input_index,output_index)
output_index<-output_index+1
smallest_vertex_highest_degree(10,50,input_index,output_index)
output_index<-output_index+1
smallest_vertex_smallest_degree(10,50,input_index,output_index)
output_index<-output_index+1
the_sequential_algorithm(10,50, input_index, output_index)
output_index<-output_index+1
input_index <- input_index+1
random_vertex_highest_degree(20,70,input_index,output_index)
output_index<-output_index+1
highest_vertex_highest_degree(20,70,input_index,output_index)
output_index<-output_index+1
smallest_vertex_highest_degree(20,70,input_index,output_index)
output_index<-output_index+1
smallest_vertex_smallest_degree(20,70,input_index,output_index)
output_index<-output_index+1
the_sequential_algorithm(20,70, input_index, output_index)
```