```
dir_name="NOVL_0"
nw_file="./"$dir_name"/novl_0.nsa"

#nnode: number of nodes in the network
nnode=5000

#alpha: if 1, then only the edge weights will be used i.e. the global importance of nodes will not be used.
alpha=1

#maxk: Number of columns you want to use in laplacian matrix. Maximum value could be the nnode.
#k1: number of communities you want to have
#k2: number of communities you want to have
maxk=50
k1=15
k2=15

#op1: weighted adjacency matrix
#op2: for running maximal clique finding algorithm mentioned below (step 2)
#op3: page rank score of the nodes
#op4: file containing maximal clique information
#op6: betweenness scores of the nodes
#op7: normalized laplacian matrix containing maxk number of columns
op1="ori_weight.txt"
op2="edge_list.txt"
op3="pr_score.txt"
op4="clq.txt"
op5="cl_adj.txt"
op6="bwsc_node.txt"
op7="lap_mat.txt"


#required to run maximal clique finding algorithm mentioned below (step 2)
ip="./"$dir_name"/"$op2
op="./"$dir_name"/clq.txt"


#Step 1:
Rscript --vanilla source_code_P1.R $dir_name $nw_file $op1 $op2 $op3

#Step 2:
#finding maximal cliques in sparse networks can be done using Quick Cliques:
#Listing All Maximal Cliques in Large Sparse Real-World Graphs in Near-Optimal Time, D. Eppstein, M. Löffler, and D. Strash, Journal of Experimental Algorithmics, 18 (3): 3.1, 2013, doi:10.1145/2543629
#use --input-file=$ip and output file $op


#Step 3:
```

```
gcc -o step_12345 step_12345.c -lm
./step_12345 $nnode $dir_name $op1 $op3 $op4 $alpha $op5

#Step 4:
Rscript --vanilla source_code_P2.R $dir_name $maxk $op5 $nw_file $op6 $op7

#Step 5:
gcc -o comm_algo comm_algo.c -lm
./comm_algo $nnode $maxk $k1 $k2 $dir_name $op7 $op6 $nw_file

#community membership file (comm.txt) vector will be generated in $dir_name directory
```