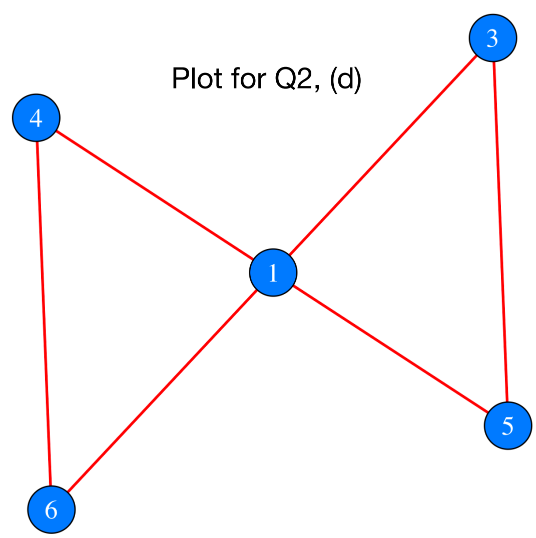
```
> ## STA352 Coding Homework
> ## Zishan (Bruce) Shao
>
> ## Q1: Consider the following network
> # (a) Give the number of edges and the number of vertices in the network
> library(igraph)


> g<-graph.formula(1-3,1-4,1-5,1-6,2-4,2-5,2-6,3-5,4-6) # create the graph
>
> vcount(g) # provide count of vertices
[1] 6
> ecount(g) # provide count of edges
[1] 9
> V(g)
+ 6/6 vertices, named, from 205c7b8:
[1] 1 3 4 5 6 2
> E(g)
+ 9/9 edges from 205c7b8 (vertex names):
[1] 1--3 1--4 1--5 1--6 3--5 4--6 4--2 5--2 6--2
>
>
> # (b): degree distribution of network
> degree(g) # calculate degrees for each of th vertices
1 3 4 5 6 2
4 2 3 3 3 3
> sort(degree(g))
3 4 5 6 2 1
2 3 3 3 3 4
> degree_distribution(g)
[1] 0.0000000 0.0000000 0.1666667 0.6666667 0.1666667
>
>
> # (c): average nearest neighbor degree for vertices 2,3, and 4
> knn(g)$knn # construct a nearest neighbor network
       1        3        4        5        6        2
2.750000 3.500000 3.333333 3.000000 3.333333 3.000000
> neighborhood(g,order=1,nodes=1,mindist=1)
[[1]]
+ 4/6 vertices, named, from 205c7b8:
[1] 3 4 5 6

> lapply(neighborhood(g,order=1,mindist=1),degree,g=g)
[[1]]
3 4 5 6
2 3 3 3

[[2]]
1 5
4 3

[[3]]
1 6 2
4 3 3
```

```
[[4]]
1 3 2
4 2 3

[[5]]
1 4 2
4 3 3

[[6]]
4 5 6
3 3 3

> sapply(lapply(neighborhood(g,order=1,mindist=1),degree,g=g), mean)
[1] 2.750000 3.500000 3.333333 3.000000 3.333333 3.000000
>
>
> # (d):
> A<-get.adjacency(g)
> A<-A[order(rownames(A)),order(rownames(A))]
> g<-graph_from_adjacency_matrix(A,mode="undirected")
>
> distances(g)
  1 2 3 4 5 6
1 0 2 1 1 1 1
2 2 0 2 1 1 1
3 1 2 0 2 1 2
4 1 1 2 0 2 1
5 1 1 1 2 0 2
6 1 1 2 1 2 0
> diameter(g) # get the diameter
[1] 2
> # get the shortest path (explore the pair with no 2,3,4 vertices)
> which(shortest.paths(g)==diameter(g),arr.ind=TRUE)
  row col
2   2   1
1   1   2
3   3   2
2   2   3
4   4   3
6   6   3
3   3   4
5   5   4
4   4   5
6   6   5
3   3   6
5   5   6
>
>
> # (e):
> g_less <- delete_edges(g, get.edge.ids(g, c(1,4)))
> A_less<-get.adjacency(g_less)
> A_less<-A_less[order(rownames(A_less)),order(rownames(A_less))]
> g_less<-graph_from_adjacency_matrix(A_less,mode="undirected")
```

```
>
> diameter(g_less) # proven to increase by 1
[1] 3
>
>
> # (f):
> closeness(g)
        1         2         3         4         5         6 
0.1666667 0.1428571 0.1250000 0.1428571 0.1428571 0.1428571 
> apply(distances(g),1,sum)
1 2 3 4 5 6 
6 7 8 7 7 7 
> 1/apply(distances(g),1,sum) # proven that it is 1, 3
        1         2         3         4         5         6 
0.1666667 0.1428571 0.1250000 0.1428571 0.1428571 0.1428571 
>
>
> # (g)
> betweenness(g) # vertex betweeness --> proved
        1         2         3         4         5         6 
3.0000000 1.0000000 0.0000000 0.3333333 1.3333333 0.3333333 
>
>
> # (h)
> E(g)
+ 9/9 edges from 3d11cbb (vertex names):
[1] 1--3 1--4 1--5 1--6 2--4 2--5 2--6 3--5 4--6
> edge_betweenness(g) # edge betweeness --> proved
[1] 3.000000 2.833333 2.333333 2.833333 1.833333 3.333333 1.833333 2.000000 1.000000
>
>
> # (i)
> table(sapply(cliques(g),length))

1 2 3 
6 9 3 
> cliques(g,min=3,max=3)
[[1]]
+ 3/6 vertices, named, from 3d11cbb:
[1] 2 4 6

[[2]]
+ 3/6 vertices, named, from 3d11cbb:
[1] 1 4 6

[[3]]
+ 3/6 vertices, named, from 3d11cbb:
[1] 1 3 5

>
>
> # (j)
> gtemp<-g
> gtemp[1,2]<-1
```

```
> cliques(gtemp,min=4,max=4)
[[1]]
+ 4/6 vertices, named, from dc5e9d1:
[1] 1 2 4 6

>
>
> # (k)
> max_cliques(g,min=2,max=2)
[[1]]
+ 2/6 vertices, named, from 3d11cbb:
[1] 2 5

>
>
>
>
> ## Q2:
>
> # (a)
> edge_density(g)
[1] 0.6
> ecount(g)/choose(vcount(g),2)
[1] 0.6
>
> # (b)
> transitivity(g)
[1] 0.4736842
>
> # (c)
> choose(degree(g),2)
1 2 3 4 5 6
6 3 1 3 3 3
> sum(choose(degree(g),2))
[1] 19
> transitivity(g,type="local")  # This proves my result
[1] 0.3333333 0.3333333 1.0000000 0.6666667 0.3333333 0.6666667
>
>
> # (d)
> ge<-make_ego_graph(g,nodes=1)[[1]]
> #(ecount(ge)-vcount(ge)+1)/choose(vcount(ge)-1,2)
> plot(ge)
```

Plot for Q2, (d)

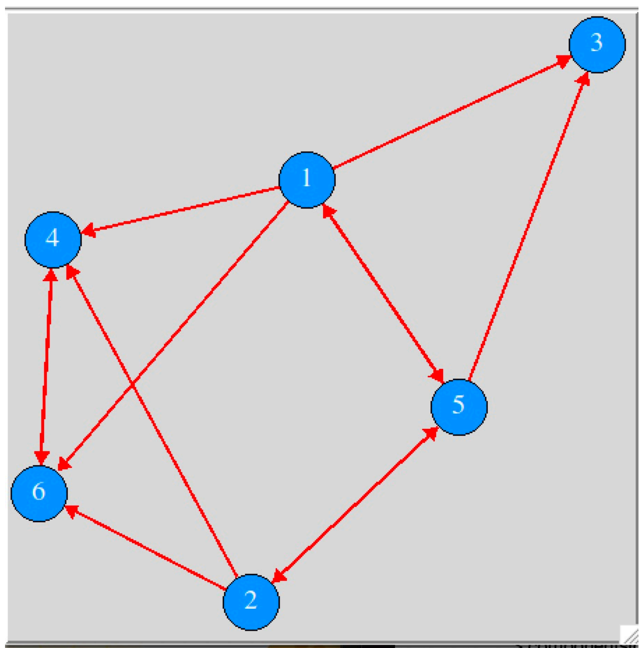



```
>
> # (e)
> # vertice connectivity
> vertex_connectivity(g)
[1] 2
> components(g-vertices(c(1,5)))
$membership
2 3 4 6
1 2 1 1
```

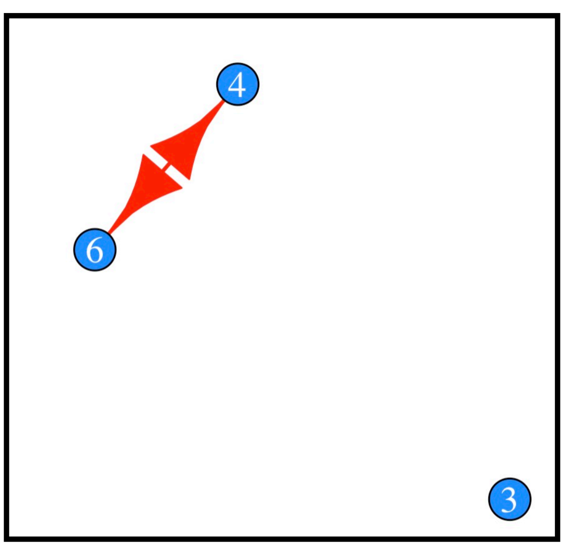
```
$csize
[1] 3 1

$no
[1] 2

>
> # (f)
> # edge connectivity
> edge_connectivity(g)
[1] 2
> components(g-edges(1:3, 3:5))
$membership
1 2 3 4 5 6
1 2 2 2 2 2

$csize
[1] 1 5

$no
[1] 2
```

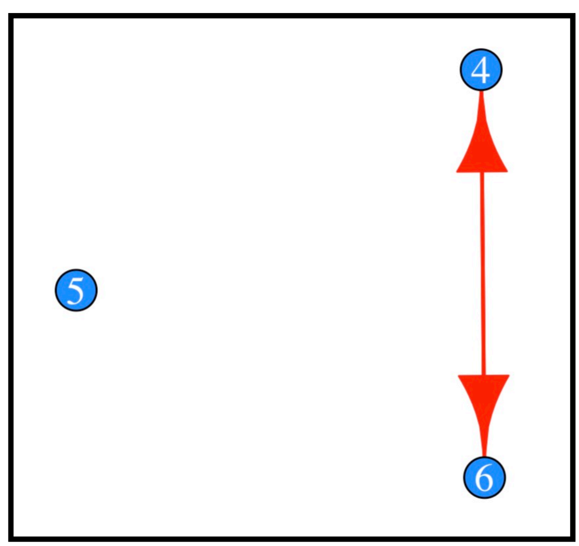
```
> ## Q3:
>
> # Define the directed graph here
> library(igraph)

> gd <- graph.formula(1-+3, 1-+4, 1++5, 1-+6, 2-+4, 2++5, 2-+6, 4++6, 5-+3)
>
> igraph.options(vertex.color="dodgerblue", vertex.size=20,
+               vertex.label.cex=1.25, vertex.label.color="white",
+               edge.color="red", edge.arrow.size=1, edge.width=2)
>
> A <- get.adjacency(gd)
> A <- A[order(rownames(A)), order(rownames(A))]
> gd <- graph_from_adjacency_matrix(A)
>
> # Create a tkplot and save the plot ID
> tkp.id <- tkplot(gd)
>
> # Use the plot ID to manipulate the plot
> L <- tk_coords(tkp.id, norm=TRUE) * 1.5
> L[, 1] <- L[, 1] * 1.5
> # plot(gd, layout=L, rescale=FALSE)
>
>
> # (a)
> plot(gd,layout=L,rescale=FALSE)
> dyad.census(gd)
$mut
[1] 3
```

```
$asym
[1] 6

$null
[1] 6

> triad_census(gd)
 [1] 0 5 2 2 2 0 0 5 0 0 1 2 1 0 0 0
>
>
> # (c)
> plot(induced.subgraph(gd, c(1,3,5)))
```




```
> # (d)
> plot(induced.subgraph(gd, c(3,4,6)))
> plot(induced.subgraph(gd, c(5,6,4)))
```





```
> # (e)
> reciprocity(gd,mode="default")
[1] 0.5
>
>
> # (f)
> components(gd,mode="strong")
$membership
1 2 3 4 5 6
1 1 3 2 1 2

$csize
[1] 3 2 1

$no
[1] 3

> sort(components(gd,mode="strong")$membership)
1 2 5 4 6 3
1 1 1 2 2 3
>
```