INDEX

EXPERIMENT -5

AIM: To implement Caesar Cipher receiving input from a file.

Program:

```
#include <bits/stdc++.h>
using namespace std;
static int substitution_index;


char encrypt(char c){

	if (isdigit(c))
		return (c -'0' +substitution_index)%10 + '0';

	else if (isupper(c))
		return char((65 + (int(c)  - 65 +substitution_index)%26 ));

	else return char((97 + (int(c)  - 97 +substitution_index)%26 ));

}

void solve(vector<char> & input){
	vector<char>:: iterator it = input.begin();
	vector<char> output;

	cout<<"\n The Decrypted Message: ";

	for(;it!=input.end();it++){
		cout<<*it;

		if (isalnum(*it))
			output.push_back(encrypt(*it));
		else
			output.push_back(*it);
	}

	vector<char>:: iterator ot = output.begin();
	cout<<"\n The Encrypted Message: ";
	for(;ot!=output.end();ot++)
		cout<<*ot;

}


int main(){
```

```
        substitution_index=3;

        ifstream inf("input.txt");
        char c;
        vector<char>input;

        while((c=inf.get())!=EOF)
                input.push_back(c);

        inf.close();
        solve(input);
        cout<<endl;

        return 0;
}
```

OUTPUT:

```
aman@aman ~/Desktop/prog/NS/9-23 Cipher $ cat input.txt

^&*^m&&*yCommpuTER%878@& is A &%& @netwoRking prodigy!!

aman@aman ~/Desktop/prog/NS/9-23 Cipher $ ./caesar

 The Decrypted Message:
^&*^m&&*yCommpuTER%878@& is A &%& @netwoRking prodigy!!


 The Encrypted Message:
^&*^p&&*bFrppsxWHU%101@& lv D &%& @qhwzrUnlqj surgljb!!
```

EXPERIMENT -6

AIM: Implement Playfair Cipher receiving input from a file and key string for the matrix from the user.

Program:

```
#include <bits/stdc++.h>
using namespace std;

static int substitution_index;
char *mat[8];

void build_matrix(string &s1,char ** mat){

	for(int i=0 ; i<8; i++)
		mat[i]= new char[8];

	for(int i=0; i<8; i++)
		for(int j=0; j<8; j++)
			mat[i][j]='!';

	set<char> unique;
	for(int i=0; i<s1.size(); i++)
		unique.insert(s1[i]);

	set<char>::iterator sit=unique.begin();
	int last_row,last_col;
	bool over=false;

	for(int i=0; i<8; i++){
		for(int j=0; j<8; j++)
			if (sit!=unique.end()){

				mat[i][j]=*sit;
				sit++;
			}
			else{
				last_col=j;
				last_row=i;
				over=true;
				break;
			}
		if (over)break;
	}
```

```
	vector<char>not_added_lower;
	for(char x='a'; x<='z'; x++)
		if (find(unique.begin(), unique.end(),x)==unique.end())
			not_added_lower.push_back(x);

	for(int i=0; i<not_added_lower.size(); i++){

		if ((last_col)%8==0){
			last_row+=1;
			last_col=0;
			mat[last_row][last_col++]=not_added_lower[i];
		}
		else mat[last_row][last_col++]=not_added_lower[i];

	}

	vector<char>not_added_upper;
	for(char x='A'; x<='Z'; x++)
		if (find(unique.begin(), unique.end(),x)==unique.end())
			not_added_upper.push_back(x);

	for(int i=0; i<not_added_upper.size(); i++){
		if ((last_col)%8==0){
			last_row+=1;
			last_col=0;
			mat[last_row][last_col++]=not_added_upper[i];
		}
		else mat[last_row][last_col++]=not_added_upper[i];
	}

	for (int i=0; i<10; i++){
		if ((last_col)%8==0){
			last_row+=1;
			last_col=0;
			mat[last_row][last_col++]=i+'0';
		}
		else mat[last_row][last_col++]=i+'0';
	}

	cout<<"\n";
	for(int i=0; i<8; i++){
		for(int j=0; j<8; j++)
			cout<<" "<<mat[i][j];
		cout<<endl;
	}
}
```

```
char encrypt(char a,char b,vector<char>& output){

	int x1,x2,y1,y2;

	for(int i=0; i<8; i++)
		for(int j=0; j<8; j++)
			if (mat[i][j]==a){
				x1=i;
				y1=j;
				break;
			}

	for(int i=0; i<8; i++)
		for(int j=0; j<8; j++)
			if (mat[i][j]==b){
				x2=i;
				y2=j;
				break;
			}

	if (x1==x2){
		output.push_back(mat[x1][(y1+1)%8]);
		output.push_back(mat[x2][(y2+1)%8]);
		output.push_back(' ');
	}

	else if (y1==y2){
		output.push_back(mat[(x1+1)%8][y1]);
		output.push_back(mat[(x2+1)%8][y2]);
		output.push_back(' ');
	}

	else{
		output.push_back(mat[x2][y1]);
		output.push_back(mat[x1][y2]);
		output.push_back(' ');
	}
}

void solve(vector<char> & input_1,vector<char> & input_2){

	vector<char>:: iterator it = input_1.begin();
	vector<char>:: iterator it2 = input_2.begin();
	vector<char> output;

	cout<<"\n The Decrypted Message: ";

	for(;it!=input_1.end();it++,it2++){
```

```
		cout<<*it<<*it2<<" ";
		encrypt(*it,*it2,output);
	}

	vector<char>:: iterator ot = output.begin();
	cout<<"\n The Encrypted Message: ";
	for(;ot!=output.end();ot++)
		cout<<*ot;
}

int main(){

	string s1;
	cout<<"\n Enter key string for the matrix: ";
	cin >> s1;

	build_matrix(s1,mat);

	ifstream inf("input.txt");
	char c,i1,i2;
	vector<char>input_1;
	vector<char>input_2;

	while((c=inf.get())!=EOF){

		if(!isalnum(c))
		continue;

		i1=c;
		i2=inf.get();

		while(!isalnum(i2))
			i2=inf.get();

		if(i1==i2){
			input_1.push_back(c);
			input_2.push_back('!');
			input_2.push_back(c);
			input_1.push_back('!');
		}
		else{
			input_1.push_back(i1);
			input_2.push_back(i2);
		}

	}
```

```
        inf.close();
        solve(input_1,input_2);
        cout<<endl<<endl;


        return 0;
}
```

OUTPUT:

```
aman@aman ~/Desktop/prog/NS/9-23 Cipher $ cat input.txt

^&*^m&&*yCommpuTER%878@& is A &%& @netwoRking prodigy!!

aman@aman ~/Desktop/prog/NS/9-23 Cipher $ ./play_fair

 Enter key string for the matrix: JobanpreetMaam

 J M a b e m n o
 p r t c d f g h
 i j k l q s u v
 w x y z A B C D
 E F G H I K L N
 O P Q R S T U V
 W X Y Z 0 1 2 3
 4 5 6 7 8 9 ! !

 The Decrypted Message: my Co m! !m pu TE R8 78 is An et wo Rk in gp ro di gy
 The Encrypted Message: Ba nD 9n n9 ig KO 7S 89 ju eC da JD lQ Ju hr Mh qp Ct
```

EXPERIMENT - 7

AIM: To implement unicasting in a LAN using TCP and UDP connections.

Scenario: A local area network of three nodes is connected to multiple nodes and the analysis of the network is made on the variation of congestion window of TCP agents.

Tcl Program:

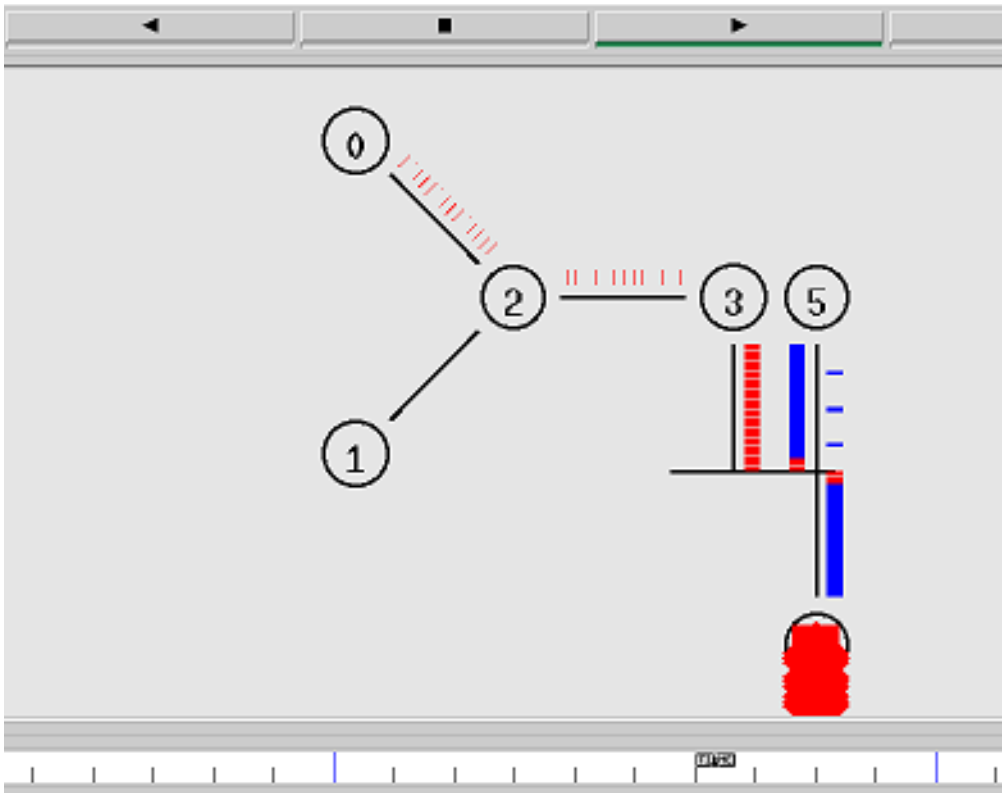
```
set ns [new Simulator]


set nt [open democong.tr w]
$ns trace-all $nt
set nf [open namfile1.nam w]
$ns namtrace-all $nf
set out [open congestion.xg w]


#creating nodes

set s0 [$ns node]


set s1 [$ns node]

set s2 [$ns node]
set s3 [$ns node]

set s4 [$ns node]


set s5 [$ns node]


set lan [$ns newLan "$s3 $s4 $s5" 0.5Mb 40ms LL MAC/Csma/Cd Channel]

#creating link
#puts "hello"

$ns duplex-link $s0 $s2 3Mbps 20ms DropTail
$ns queue-limit $s0 $s2 5
$ns duplex-link-op $s0 $s2 orient right-down


$ns simplex-link $s1 $s2 1.5Mbps 20ms DropTail
```

```
$ns queue-limit $s1 $s2 7
$ns simplex-link-op $s1 $s2 orient right-up

$ns duplex-link $s2 $s3 2Mbps 20ms DropTail
$ns queue-limit $s2 $s3 5
$ns duplex-link-op $s2 $s3 orient right


#creating agents

set n0 [new Agent/TCP]
$ns attach-agent $s0 $n0

set n1 [new Agent/UDP]
$ns attach-agent $s1 $n1

set destudp [new Agent/Null]
$ns attach-agent $s5 $destudp

set desttcp [new Agent/TCPSink]
$ns attach-agent $s5 $desttcp

#setting Traffic
set traffic1 [new Application/Traffic/CBR]
$traffic1 attach-agent $n1
$traffic1 set packetSize_ 50B
$traffic1 set interval_ 3ms


set traffic2 [new Application/FTP]
$traffic2 attach-agent $n0
$traffic2 set packetSize_ 50B
$traffic2 set interval_ 1ms

#creating connect
$ns connect $n0 $desttcp
$ns connect $n1 $destudp

#creating LAN

$n0 set window_ 15


proc plotwindow {n0 out} {
global ns
set now1 [$ns now]
set cwnd1 [$n0 set cwnd_]
puts $out "$now1 $cwnd1"
```

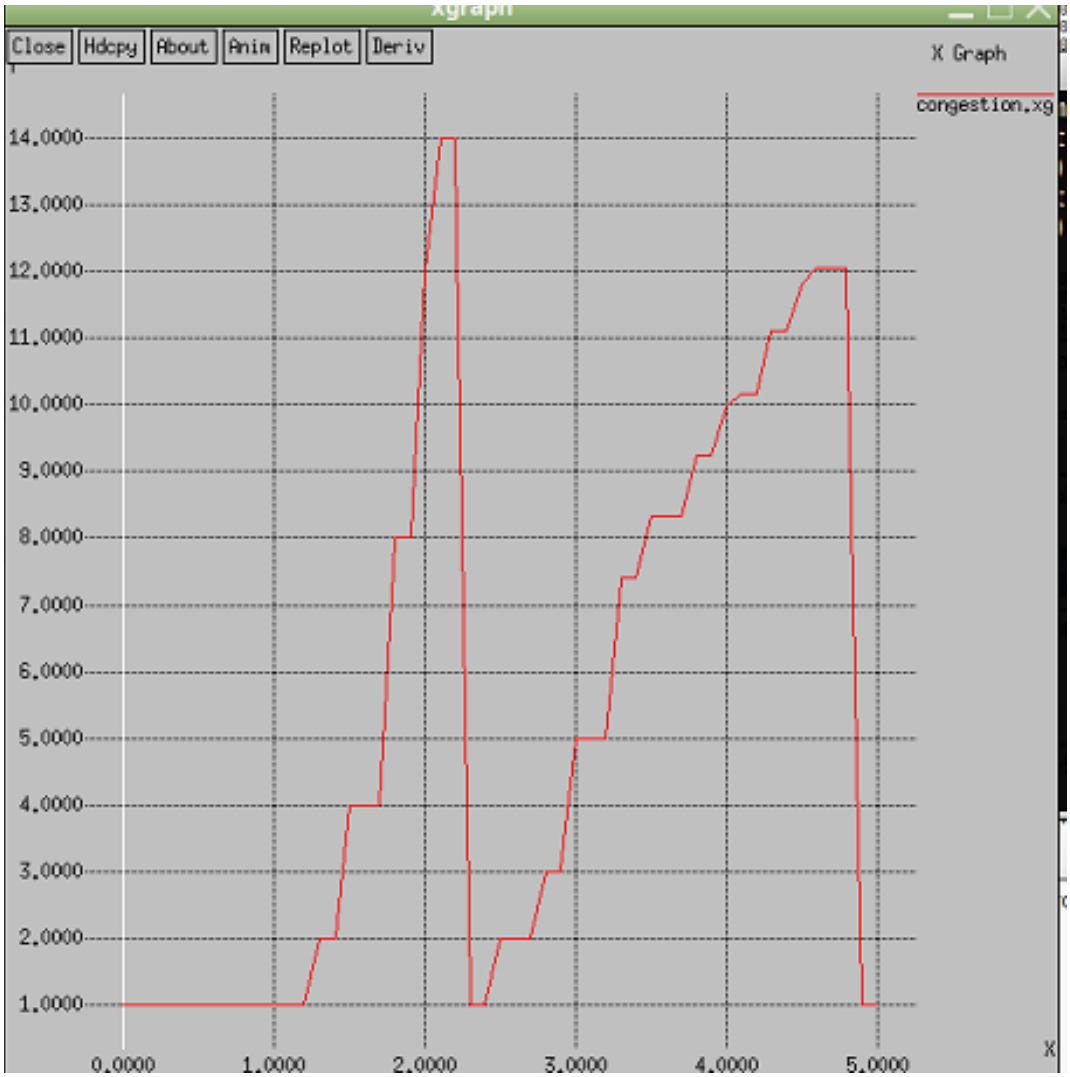
```
$ns at [expr $now1 + 0.1] "plotwindow $n0 $out"
}

proc finish { } {
global ns nf
$ns flush-trace
close $nf
exec nam namfile1.nam
exit 0
}

$ns at 0.0 "plotwindow $n0 $out"
$ns at 1.0 "$traffic1 start"
$ns at 1.0 "$traffic2 start"
$ns at 4.4 "$traffic1 stop"
$ns at 4.4 "$traffic2 stop"

$ns at 5.0 "finish"
$ns run
```

OUTPUT:

The snapshot of NAM file of unicasting in a LAN using TCP and UDP connections:



The snapshot of congestion window graph of TCP agent



EXPERIMENT - 8

AIM: To implement multicasting in a wired network

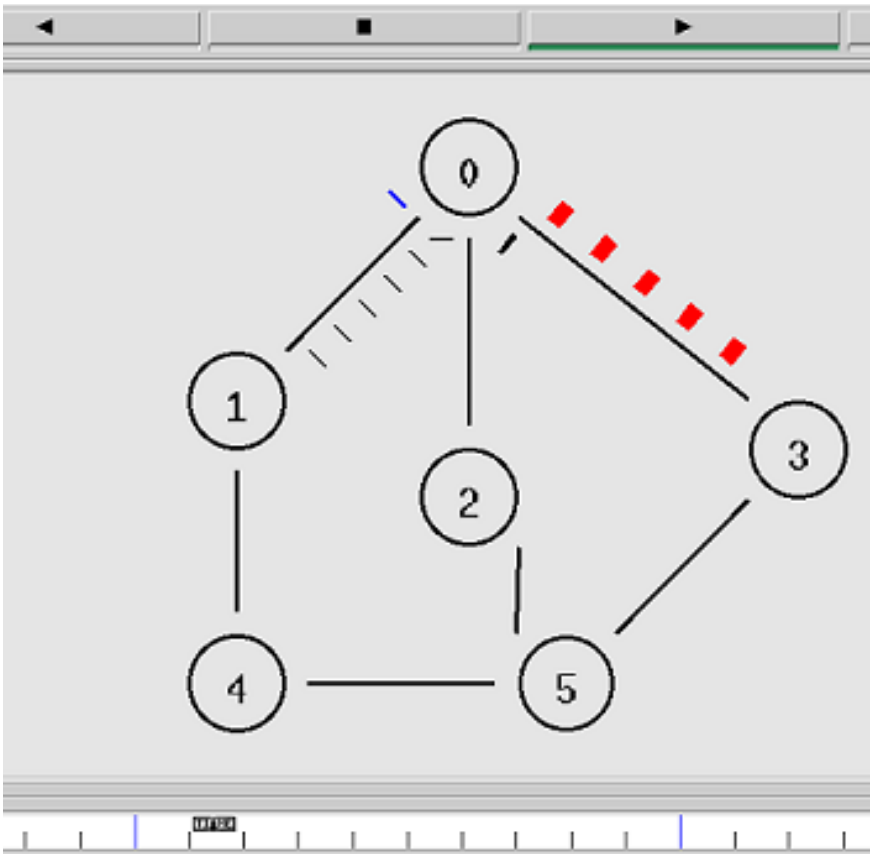
```
#To implement a multicasting topology

set ns [new Simulator]
$ns multicast

set nt [open trace1.tr w]
$ns trace-all $nt

set nf [open namfile.nam w]
$ns namtrace-all $nf

proc finish { } {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam namfile.nam
        exit 0
}

#Initialise all nodes

for { set i 1 } { $i <7} {incr i} {
        set n$i [$ns node]
}

#Give different color to source nodes
$n1 shape "square"
$n1 color Green
$n6 shape "square"
$n6 color Green

#Initialise duplex connections between all given links

$ns duplex-link $n1 $n2 2Mbps 5ms DropTail
$ns duplex-link $n1 $n4 2Mbps 5ms DropTail
$ns duplex-link $n2 $n3 2Mbps 5ms DropTail
$ns duplex-link $n4 $n3 2Mbps 5ms DropTail
```

```
$ns duplex-link $n3 $n5 2Mbps 5ms DropTail
$ns duplex-link $n5 $n6 2Mbps 5ms DropTail

$ns duplex-link-op $n1 $n2 orient left-down
$ns duplex-link-op $n1 $n4 orient right-down
$ns duplex-link-op $n2 $n3 orient left-down
$ns duplex-link-op $n4 $n3 orient right-down
$ns duplex-link-op $n3 $n5 orient right
$ns duplex-link-op $n5 $n6 orient up

#Insert sources
set src1 [new Agent/UDP]
set src2 [new Agent/UDP]
$ns attach-agent $n1 $src1
$ns attach-agent $n6 $src2

#Set groups
set group1 [Node allocaddr]
set group2 [Node allocaddr]

#Set protocol
set mproto DM
set mrthandle [$ns mrtproto $mproto]
$ns color 30 Green
$ns color 31 Red

#Set destinations
set dest1 [new Agent/Null]
set dest2 [new Agent/Null]
$src1 set dst_addr_ $group1
$src2 set dst_addr_ $group2

#Initiate traffic from the two sources
set traffic1 [new Application/Traffic/CBR]
$traffic1 attach-agent $src1
$traffic1 set packetSize_ 150B
$traffic1 set interval_ 1ms

set traffic2 [new Application/Traffic/CBR]
$traffic2 attach-agent $src2
$traffic2 set packetSize_ 150B
```

```
$traffic2 set interval_ 1ms

#Simulate the topology
$ns at 1.0  "$traffic1 start"
$ns at 1.5  "$n2 join-group $dest1 $group1"
$ns at 5.0  "$n4 join-group $dest1 $group1"
$ns at 5.0  "$n3 join-group $dest1 $group1"
$ns at 7.0  "$traffic1 stop"
$ns at 8.0  "$traffic2 start"
$ns at 9.0  "$n5 join-group $dest2 $group2"
$ns at 15.0 "$traffic2 stop"
$ns at 16.0 "finish"
$ns run
```

OUTPUT:

The snapshot of NAM file of multicasting in wired network:



EXPERIMENT - 9

AIM: Program to implement RSA cryptography algorithm.

Program:

```
#include <bits/stdc++.h>
using namespace std;

typedef long long int llu;

llu power_exp(llu a,llu p,llu MOD)
{
   llu RES = 1;
   llu B = (llu) a;
   while(p > 0)
   {
      if(p%2 == 1) RES = (RES*B)%MOD;
      B = (B*B)%MOD;
      p = p/2;
   }
   llu res = (llu) RES;
   return res;
}


llu compute_d(llu e, llu z){

        llu d = 1;
        bool count= true;

        while(count){
                d+=1;
                if ((e*d) % z == 1)
                        count=false;
        }
        return d;
}

bool isCoprime(llu &a, llu &b){
        set<llu>factor_a;

        for(llu i = 1 ; i<=sqrt(a); i++)
                if (a % i == 0){
                        factor_a.insert(i);
                        factor_a.insert(a/i);
                }
```

```
	set<llu> :: iterator si = factor_a.begin();
	for(; si!=factor_a.end(); si++)
		if (b % *si ==0 && *si != 1)
			return false;
	return true;
}

inline llu encrypt(llu input, llu e , llu n){
	llu enc = power_exp(input, e, n);
	cout<<"\n Encrypted "<<input<<" as "<<enc;
	return enc;
}

inline llu decrypt(llu cipher, llu d, llu n){
	llu dec = power_exp(cipher, d, n);
	cout<<"\n Decrypted "<<cipher<<" as "<<dec;
}


int main(){

	llu a,b,e = 1;
	llu input;

	cout <<"\nEnter two space separated lluegers: "; cin >> a >> b;
	assert (isCoprime(a,b));

	llu n = a*b , z =  (a-1) * (b-1);
	cout <<"\n N: "<< n;
	cout <<"\n Z: " << z;

	while(++e){
		if(e<z){
			if(gcd(z,e)==1)
				break;
		}
		else{
			if(gcd(e,z)==1)
				break;
		}
	}

	llu d = compute_d(e,z);
	cout <<"\n E: " << e;
	cout <<"\n D: " << d;

	cout << "\n\n Public Key: (" << e << "," << n << ")";
```

```
        cout << "\n Private Key: (" << d << "," << n << ")"<<endl;

        while(true){
                cout<<"\n\n Enter confidential text : "; cin>>input;
                llu enc = encrypt(input, e, n);
                decrypt(enc, d, n);
        }
        return 0;
}
```

OUTPUT:

```
Enter two space separated lluegers: 61 53

 N: 3233
 Z: 3120
 E: 17
 D: 2753

 Public Key: (17,3233)
 Private Key: (2753,3233)


 Enter confidential text : 61

 Encrypted 61 as 610
 Decrypted 610 as 61

 Enter confidential text : 65

 Encrypted 65 as 2790
 Decrypted 2790 as 65

 Enter confidential text : 6

 Encrypted 6 as 824
 Decrypted 824 as 6

 Enter confidential text : 811

 Encrypted 811 as 2466
 Decrypted 2466 as 811
```