Client :

./client 127.127.127.0
The Socket was created
The connection was accepted with server 127.127.127.0
Enter filename to Request : test.txt
Request accepted by server ... Writing to receive contents of file.
Result obtained , The contents of file are :..

Hey, This is Komal Jain
Bye

Server :

./server 0.0.0.0 15000
The socket was created
Binding socket to IP : 0.0.0.0
The client 127.0.0.1 is connected ...
A request for filename test.txt received ...

Reading the file contents
Sending contents to client
Request completed

Experiment Name / No.: _____ Camlin Page No. Date / /

1. Implement a client & server communication using sockets programming.

Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include <netinet/in.h>
int main(int argc, char *argv[])
{
    int create_socket, buffersize = 1024, count;
    char *buffer = malloc(buffersize), fname[256];
    struct sockaddr_in address;

    if ((create_socket = socket(AF_INET, SOCK_STREAM, 0)) > 0)
        printf("Socket was created\n");
    else exit(0);
    address.sin_family = AF_INET;
    address.sin_port = htons(15002);
    inet_pton(AF_INET, argv[1], &address.sin_addr);
    if (connect(create_socket, (struct sockaddr *) &address, sizeof(address)) == 0)
        printf("Connection accepted with server %s...\n", argv[1]);
    else exit(0);
```

Teacher's Signature: _____

```
printf ("Enter filename to Request: ");
scanf ("%s", fname);
Send (create_socket, fname, sizeof (fname), 0);
printf ("Request Accepted by server. Waiting to recieve
                                    content of file... \n\n");
printf ("Recvd obtained, the contents of file are... \n\n");
while ((cont = recv (create_socket, buffer, bufsiz, 0))>0)
    write (1, buffer, cont);
printf ("\nEOF\n");
return close (create_socket);
}
```

Server.c:

```
int main (int argc, char *argv[])
{
    int cont, create_socket, new_socket, addrlen, fd;
    int bufsize = 1024;
    char *buffer = malloc (bufsize);
    char fname[256];
    struct sockaddr_in address;
    if ((create_socket = socket (AF_INET, SOCK_STREAM, 0))>0)
        printf ("Socket was created \n");
    else exit(0);
    inet_pton (AF_INET, SOCK_STREAM, 0)) > 0)
        printf ("The socket was created \n");
    if (bind (create_socket, (struct sockaddr*) &address, sizeof (address))==0)
        printf ("Binding socket to IP: %s\n", inet_ntoa (address.sin_addr));
    else exit(0);
```

Teacher's Signature:

listen (create_socket, 3);
addrlen = sizeof (struct sockaddr_in);
new_socket = accept (create_socket, (struct sockaddr*)
&address, &addrlen);

if (new_socket >0)
printf ("Client is connected ... \n");
inet_ntoa (address.sin_addr);

char ext[0];
read (new_socket, frame, 251, 0);
printf ("A request for file name to be received ... \n", frame);

if ((fd = open (frame, O_RDONLY)) < 0)
{ perror ("File open failed"); exit(0); }

while ((count = read (fd, buffer, buf_size)) > 0)
{
printf ("Reading file contents \n");
printf ("Sending contents to client \n");
send (new_socket, buffer, count, 0);
}
printf ("Request completed \n");
close (new_socket);
close (create_socket);
}

Test Input :
Hey! This is Berserk Fans.

Teacher's Signature

- Distance vector

From Router 4 to Router 2
Cost: 7 | Path: 2 ← 3 ← 4

From Router 4 to Router 3
Cost: 4 | Path: 3 ← 4

From Router 4 to Router 4
Cost: 0 | Path: 4

From Router 5 to Router 1
Cost: 10 | Path: 1 ← 2 ← 3 5

From Router 5 to Router 2
Cost: 9 | Path 2 ← 5

From Router 5 to Router 4
Cost: 2 | Path: 4 ← 5

From Router 5 to Router 5
Cost: 0 | Path: 5

Experiment Name / No.: _____ Camlin Page No. Date / /

Q. Write a program to implement distance vector routing protocol for a simple topology of routers.

```
#include <stdio.h>
int A[10][10], n, d[10], p[10];
int bellman_ford(){
    int i, u, v;
    for(int i=0; i<n; i++){
        for(u=0; u<n; u++){
            for(v=0; v<n; v++){
                if(d[v] > d[u] + A[u][v]){
                    d[u] = d[u] + A[u][v];
                    p[v] = u; }
            }
        }
    }
    for(u=0; u<n; u++){
        for(v=0; v<n; v++){
            if(d[v] > d[u] + A[u][v]){
                printf("Detected negative edge \n");
                return -1; }
        }
    }
    return 0;
}

int main(){
```

Teacher's Signature: _____

Experiment Name / No.: _____ Camlin Page No. Date / /

```
printf ("N: ");
scanf ("%d", &n);
printf ("Matrix:\n");
int source=0, destination=0;
for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
                scanf("%d", &A[i][j]);
        }
}
for(source=0; source<n; source++){
    for(destination=0; destination<n; destination++)
    { for(int i=0; i<n; i++) {
            d[i]=999; p[i]=-1; }
    d[source]=0;
    int valid = bellman_ford();
    if(valid==-1){
        printf("Graph contains negative cycle\n");
        return 0;}
    printf("\n From Router %d to Router %d\n", source+1,
    printf("Cost: %2d|Path:", d[destination], destination+1);
    if(destination!=source){ int j=destination;
            while(p[j]!=-1){
                printf("%d <- ", j+1); j=p[j];
            }
    }
    printf("%d\n", source+1);
}
}
```

Teacher's Signature: _____

Experiment Name / No.: _____ Camlin Page No. Date / /

3. Write a program to implement error detection and correction concept using Checksum and Hamming code.

Checksum

```
#include <stdio.h>
unsigned fields[10];
unsigned short checksum()
{
    int i; sum = 0;
    printf("Enter IP header information in 16 bit words \n");
    for(i=0; i<9; i++)
    {
        printf("field %d \n", i+1);
        scanf("%x", &fields[i]);
        sum = sum + (unsigned short) fields[i];
        while(sum >> 16) sum = (sum & 0xffff) + (sum >> 16);
    }
    sum = ~sum;
    return (unsigned short) sum;
}
int main(){
    unsigned short result1, result2;
    result1 = checksum(); // sender
    printf("computed checksum at sender %o \n", result1);
    result2 = checksum(); // receiver
    printf("Computed checksum at receiver %o \n", result2);
    if(result1 == result2) printf("No error");
    else printf("Error in data received"); }
```

Teacher's Signature: _____

Experiment Name / No.: _____ Camlin Page No. Date / /

Hamming Code

```
#include <stdlib.h>
#include <stdio.h>
int main()
{
int a[4], b[4], r[3], s[3], i, q[3], c[7];
printf("\n enter 4 bit data word:\n");
for(i=3; i>=0; i--)
    scanf("%d", &a[i]);
r[0] = (a[3] + a[1] + a[0]) %2;
r[1] = (a[0] + a[2] + a[3]) %2;
r[2] = (a[1] + a[2] + a[3]) %2;
for(i=3; i>=0; i--)
    printf("\n enter 7 bit hamming codeword:");
for(i=7; i>0; i--)
    scanf("%d", r[i]);
printf("\n");
printf("\n enter 7 bit received codeword");
for(i=7; i>0; i--)
    scanf("%d", &c[i]);
b[3] = c[7]; b[2] = c[6]; b[1] = c[5]; b[0] = c[4];
r[2] = c[3]; r[1] = c[2]; r[0] = c[1];
s[0] = (b[0] + b[1] + b[3] + r[0]) %2;
s[1] = (b[0] + b[2] + b[3] + r[1]) %2;
s[2] = (b[1] + b[2] + b[3] + r[2]) %2;
printf("\n syndrome\n");
```

Teacher's Signature: _____

Experiment Name / No.: _____ Camlin Page No. Date / /

```
for(i=2;i>=0;i--)
    printf("%d",s[i]);
if(s[2]==0 && s[1]==0 && s[0]==0)
    printf("\nReceived word has error");
if(s[2]==1 && s[1]==1 && s[0]==1){
    printf("\nError in received codeword, position - 7th
if(c[7]==0)                    bit from right\n");
    c[7]=1;
else
    c[7]=0;
printf("\nCorrected codeword is\n");
for(i=7;i>0;i--)
    printf("%d\t",c[i]);}
if(s[2]==1 && s[1]==0 && s[0]==0){
    printf("\nError in received codeword, Position
                4th - bit from right\n");
    if(c[4]==0)
        c[4]=1;
    else c[4]=0;
    printf("\nCorrected codeword\n");
    for(i=7;i>0;i--);
        printf("\nCorrected word\n");}
    if(s[2]==0 && s[1]==1 && s[0]==0){
        printf("\nError in received codeword, Position
                    2nd - bit from right\n");
    if(c[2]==0)
        c[2]=1;
```

Teacher's Signature: _____

```
else
  c[2] = 0;
printf("Corrected word \n");
for(i = 7; i > 0; i--)
    printf("%d\t", c[i]); }
if(s[2] == 0 && s[1] == 0 && s[0] == 1)
{
  printf("Error received codeword, Possible 1st
                    bit from right\n");
  if(c[1] == 0)
    c[1] = 1;
  else
    c[1] = 0;
  printf("Corrected word is \n");
  for(i = 7; i > 0; i--)
  {  printf("%d\t", c[i]);
  }
  return 1;
}
```

gcc -D list listserver.c
./list

RVCE - CSE
RVCE - CSE
RVCE - CSE
RVCE - CSE
RVCE - CSE
RVCE - CSE
RVCE - CSE
;

Experiment Name / No.: _____ Camlin Page No. Date / /

4) Implement a simple multicasting routing algorithm.

listener.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"
#define MSGBUFSIZE 25

int main (int argc, char *argv[])
{
    struct sockaddr_in addr;
    int fd, nbytes, addrlen;
    struct ip_mreq mreq;
    char msgbuf[MSGBUFSIZE];
    u_int yes = 1;
    if ((fd = socket (AF_INET, SOCK_DGRAM, 0)) < 0){
        perror ("socket"); exit(1); }
    if (setsockopt (fd, SOL_SOCKET, SO_REUSEADDR, &yes,
                    sizeof(yes)) < 0){
        perror ("Reusing ADDR failed"); exit(1); }
    memset (&addr, 0, sizeof(addr));
```

Teacher's Signature: _____

Experiment Name / No.: _____ Camlin Page No. Date / /

```
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = htonl(INADDR_ANY);
addr.sin_port = htons(MCAST_PORT);
if(bind(fd, (struct sockaddr *)&addr, sizeof(addr))<0)
{ printf("bind"); exit(1); }
mreq.imr_multiaddr.s_addr = inet_addr(MCAST_GROUP);
mreq.imr_interface.s_addr = htonl(INADDR_ANY);
if(setsockopt(fd, IPPROTO_IP, IP_ADD_MEMBERSHIP,
                &mreq, sizeof(mreq))<0){
    perror("setsockopt"); exit(1); }
while(1) {
    addrlen = sizeof(addr);
if((nbytes = recvfrom(fd, msgbuf, MSGBUFSIZE, 0,
            (struct sockaddr *)&addr, &addrlen))<0){
    perror("recvfrom");
    exit(0);
}
puts(msgbuf);
}
}
```

server.c

```
int main(int argc, char *argv[]) {
    struct sockaddr_in addr;
    int fd, nbytes;
    struct ip_mreq mreq;
    char *message = "HELLO";
    if((fd = socket(AF_INET, SOCK_DGRAM, 0))<0){
```

Teacher's Signature: _____

Experiment Name / No.: _____ Camlin Page No. Date / /

```
perror("socket"); exit(1); }
memset(&addr, 0, sizeof(addr));
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr(HELLO_GROUP);
addr.sin_port = htons(HELLO_PORT);
while(1){
    if(sendto(fd, message, sizeof(message), 0,
              (struct sockaddr *) &addr,
              sizeof(addr)) < 0) {
        perror("sendto");
        exit(1);
    }
    sleep(1);
}
return 0;
}
```

Teacher's Signature: _____

•/cli
Connection established to Rounak's server
•/cli
Connection established to Rounak's server

Rounak's Server started listening on port 8080 .....
Client 1 connected.
Client 2 connected.

Experiment Name / No.: _____ Camlin Page No. Date / /

5) Write a program to implement concurrent chat server that allows current logged in users to communicate one with other.

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <pthread.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
void* doReceiving (void * sockID){
        int clientSocket = *((int*) sockID);
        while (1){
            char data[1024];
            int read = recv (clientSocket, data, 1024, 0);
            data[read] = '\0';
            printf ("%s\n", data);
        }
}
int main(){
    int clientSocket = socket (PF_INET, SOCK_STREAM, 0);
    struct sockaddr_in serverAddr;
    serverAddr.sin_family = AF_INET;
```

Teacher's Signature: _____

Experiment Name / No.: _____ Camlin Page No. Date / /

```
serverAddr.sin_port = htons(8800);
serverAddr.sin_addr.s_addr = htonl(INADDR_ANY);
if(connect(clientSocket, (struct sockaddr*)&serverAddr,
                    sizeof(serverAddr)) < 0) return 0;
printf("Connection established to Rounak's server
                    ....\n");
pthread_t thread;
pthread_create(&thread, NULL, doReceiving,
                    (void*)&clientSocket);

while(1){
    char input[1024];
    scanf("%s", input);
    if(strcmp(input, "LIST") == 0){
        send(clientSocket, input, 1024, 0);}
    if(strcmp(input, "SEND") == 0){
        send(clientSocket, input, 1024, 0);
        scanf("%s", input);
        send(clientSocket, input, 1024, 0);
        scanf("%[^\n]s", input);
        send(clientSocket, input, 1024, 0);
    }
}
```

main.c

```
int clientCount = 0;
static pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALISER;
static pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
struct client{
    int index;
```

Teacher's Signature: _____

Experiment Name / No.: _____ Camlin Page No. Date / /

```
    int sockID;
    struct sockaddr_in clientAddr;
    int len;
};

struct client clients[1024];
pthread_t threads[1024];
void *doNetworking(void *clientDetail){
    struct client *clientDetail = (struct client*)clientDetail;
    int index = clientDetail->index;
    int clientSocket = clientDetail->sockID;
    printf("Client %d connected\n", index+1);
    while(1){
        char data[1024];
        int read = recv(clientSocket, data, 1024, 0);
        data[read] = '\0';
        char output[1024];
        if(strcmp(data, "LIST")==0){
            int k=0;
            for(int i=0; i<clientCount; i++){
                if(i!=index)
                    k+=sprintf(output+k, 1024,
                        "Client %d is at socket %d\n",
                            i+1, clients[i].sockID);
            }
        send(clientSocket, output, 1024, 0);
        continue;
    }
```

Teacher's Signature: _____

Experiment Name / No.: _____ | Camlin | Page No. | Date / /

```
if(strcmp(data, "SEND") == 0){
        read = recv(clientSocket, data, 1024, 0);
        data[read] = '\0';
        int id = atoi(data) - 1;
        read = recv(clientSocket, data, 1024, 0);
        data[read] = '\0';
        send(*(client[id].sockID), data, 1024, 0);
    }
}
return NULL;
}

int main(){
int serverSocket = socket(PF_INET, SOCK_STREAM, 0);
struct sockaddr_in serverAddr;
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(8080);
serverAddr.sin_addr.s_addr = htonl(INADDR_ANY);
if(bind(serverSocket, (struct sockaddr*)&serverAddr,
                    sizeof(serverAddr)) < 0) return 0;
if(listen(serverSocket, 1024) < 0) return 0;
printf("Rahul's Server started listening on port 8080....\n");
while(1){
    client[clientCount].sockID = accept(serverSocket, (struct sockaddr*)
                    &client[clientCount].
                    , &client[clientCount].clientAddr);
    client[clientCount].index = clientCount;
    pthread_create(&thread[clientCount], NULL, &Networking,
                (void*)&client[clientCount]);
```

Teacher's Signature

Experiment Name / No.: _____

Camlin | Page No.
Date / /

```
    clientCount++;
}
for(int i=0; i<clientCount; i++)
    pthread_join(thread[i], NULL);
}
```

Date
Page No.
No. / Name :

```
    if(n<0 && errno == EINTR)
            goto again;
    else if (n<0)
        perror ("str_echo: read error");
}

int main() {
    int listenfd, connfd;
    id_t childpid;
    socklen_t clien;
    struct sockaddr_in cliaddr, servaddr;
    listenfd = socket (AF_INET, SOCK_STREAM, 0);
    bzero(& servaddr, sizeof (servaddr));
    servaddr. sin_family = AF_INET;
    servaddr. sin_addr. s_addr = htonl (INADDR_ANY);
    servaddr. sin_port = htons (SERV_PORT);

    bind( listenfd, (struct sockaddr*) & servaddr, sizeof (servaddr));
    listen( listenfd, LISTENQ);

    for (;;) {
        clien = sizeof (cliaddr);
        connfd = accept (listenfd, (struct sockaddr*) &cliaddr,
                                        & clilen);
        str_echo(connfd);
        close(connfd);
    }
}
```

/ Name :

```
int main(){
    int sockfd;
    struct sockaddr_in servaddr;
    if (argc != 2){
        perror("usage: udpcli <IPaddress>");
        exit(-1);
    }
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(SERV_PORT);
    inet_pton(AF_INET, argv[1], &servaddr.sin_addr);
    sockfd = socket(AF_INET, SOCK_DGRAM,0);
    dg_cli(stdin, sockfd, (struct sockaddr *)&servaddr,
                                sizeof(servaddr));
    exit(0);
}
```

server.c

```
void dg_echo(int sockfd, struct sockaddr *pcliaddr,
                                socklen_t clilen){
    int n;
    socklen_t len;
    char mesg[MAXLINE];
    bzero(mesg, MAXLINE);

    for(;;){
        len = clilen;
        n = recvfrom(sockfd, mesg, MAXLINE, 0, pcliaddr,
                                &len);
```

Teacher's Signature

Expt. No. / Name :

6

Client.c

```
#include <stdio.h>
#include <stlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>

#define MAXLINE 1000
#define SERV_PORT 9002

void str_cli (FILE *fp, int sockfd){
    char sendline[MAXLINE], recvline[MAXLINE];
    while (fgets(sendline, MAXLINE, fp)){
        write (sockfd, sendline, strlen(sendline));
        if (read(sockfd, recvline, MAXLINE)==0){
            perror("str_cli: server terminated prematurely");
            exit (EXIT_FAILURE);
        }
        fputs(recvline, stdout);
    }
}
}
```

Teacher's Signature

./ Name :

```
    sendto( sockfd, mesg, strlen(mesg), 0, pcliaddr, len);
    bzero( mesg, MAXLINE);
}
}

int main( int argc, char ** argv) {
    int sockfd;
    struct sockaddr_in servaddr, cliaddr;
    sockfd = socket( AF_INET, SOCK_DGRAM, 0);

    bzero (& servaddr, sizeof (servaddr));
    servaddr. sin_family = AF_INET;
    servaddr. sin_addr. s_addr = htonl(INADDR_ANY);
    servaddr. sin_port = htons( SERV_PORT);

    bind (sockfd, (struct sockaddr *) & servaddr,
                    sizeof (servaddr));
    dg_echo( sockfd, (struct sockaddr *)& cliaddr,
                    sizeof (cliaddr));
}
```

```
int main (int argc, char ** argv) {
    int sockfd;
    struct sockaddr_in servaddr;

    if (argc != 2){
        perror("usage: tcpcli <IP> ");
        exit (EXIT_FAILURE);
    }
    sockfd = socket (AF_INET, SOCK_STREAM, 0);
    servaddr. sin_family = AF_INET;
    servaddr.sin_port = htons (SERV_PORT);
    inet_pton (AF_INET, argv[1], & servaddr.sin_addr);

    connect (sockfd, (struct sockaddr * )& servaddr, sizeof (
    str_cli (stdin, sockfd);                    servaddr));
    exit (0);
}
```

Sever.c

```
#define LISTENQ 1024

void str_echo (int sockfd) {
    ssize_t n;
    char buf [1000];
again:
    while ((n = read (sockfd, buf, 1000)) > 0) {
        write (sockfd, buf, n);
        memset (buf, '\0', strlen (buf));
    }
```

Page No.

No. / Name :

UDP

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <errno.h>

#define SERV_PORT 9002
#define MAXLINE 1024

void dg_cli(FILE *fp, int sockfd, const struct sockaddr
                                *pservaddr, socklen_t servlen){
    int n;
    char sendline[MAXLINE], recvline[MAXLINE+1];
    while(fgets(sendline, MAXLINE, fp) != NULL){
        sendto(sockfd, sendline, strlen(sendline), 0,
                                pservaddr, servlen);
        n = recvfrom(sockfd, recvline, MAXLINE, 0,
                                NULL, NULL);
        recvline[n] = '\0';
        fputs(recvline, stdout);
    }
}
```

Teacher's

7) Implementation of remote command execution using socket system calls.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <errno.h>
```

server.c

```
int main()
{
    int sd, acpt, len, bytes, port;
    char send[50], receiv[50];
    struct sockaddr_in serv, cli;
    if ((sd = socket(AF_INET, SOCK_STREAM, 0)) < 0){
        printf("Error in socket\n"); exit(0); }
    bzero(&serv, sizeof(serv));
    serv.sin_family = AF_INET;
    serv.sin_port = htons[15002];
    serv.sin_addr.s_addr = htonl(INADDR_ANY);
    if(bind(sd, (struct sockaddr *)&serv, sizeof(serv)) < 0)
    { printf("Error in bind\n"); exit(0); }
    if(listen(sd,2) < 0){
```

Teacher's Signature:

```
printf("Error in listening"); exit(0); }
while(1) { bytes = recv(acpt, recvbuf, 50, 0); recvbuf[bytes]='\0';
if (strcmp(recvbuf, "end") == 0)
{ close(acpt); close(sd); exit(0); }
else {
    printf("Command received: %s", recvbuf); system(recvbuf);
    printf("\n"); }}
```

Client

```
int main(){
int sd, acpt, len, bytes, port;
char sendbuf[50], recvbuf[50];
struct sockaddr_in serv, cli;
if ((sd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
    printf("Error in socket\n");
    exit(0);
}
bzero(&serv, sizeof(serv));
serv.sin_family = AF_INET;
serv.sin_port = htons(15002);
serv.sin_addr.s_addr = htonl(INADDR_ANY);
if (connect(sd, (struct sockaddr *)&serv, sizeof(serv)) < 0)
{
    printf("Error in connection\n"); exit(1);
}
while(1) {
```

Teacher's Signature

```
printf("Enter the command :");
gets(send1);
if(strcmp(send1,"end")!=0)
    send(sd,send1,50,0);
else{
    send(sd,send1,50,0);
    close(sd);
    break;
}
}
}
```

Date

Expt. No. 8

Page No. 45

Write a program to encrypt and ecrypt data using RSA & exchange the key securely using Diffie-Hellman key exchange protocol.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
long int gcd( long int a, long int b)
{ if (a==0) return b;
  if (b==0) return a;
  return gcd(b, a%b);}
long int isprime (long int a)
{ int i; for(i=2; i<a; i++) { if (a%i==0) return 0;} return 1;}
long int encrypt( long int ch, long int n, long int e)
{ long int i, temp=ch; for(i=1; i<e; i++) temp= (temp*ch)%n;
  return temp;}
long int decrypt ( long int ch, long int n, long int d){
  long int i; temp=ch; for(i=1; i<d; i++) temp=(temp*ch)%n;
  return temp;}
int main()
{ long int i, p, q, len, e, d, n, phi, ciphar[50]; char text[50];
printf("Enter the text to be encrypted"); scanf("%s", text);
len = strlen(text);
do{p=rand()%30;}while (!isprime(p));
do{ q= rand()%30;} while (!isprime(q)); n=p*q; phi= (p-1)*(q-1);
do{ e=rand()%phi; }while (gcd(phi,e) !=1);
do{ d=rand()%phi;} while((d*e)%phi!=1);
printf("Two prime numbers (p and q) are: %d and %d\n", p,q);
printf("n(p*q) = %d*%d = %d", p,q,p*q);
```

Teacher's Signature

Date

Expt. No. 8

Page No. 47

```
printf("\n(p-1)*(q-1) = %ld\n", phi);
printf("Public key (n,e) : (%ld, %ld)\n", n, e);
printf("Private key (n,d) : (%ld, %ld)\n", n, d);
for (i=0; i<len; i++)
    ciplen[i] = encrypt(text[i], n, e);
printf("Encrypted message : \n");
    for(i=0; i<len; i++) printf("%ld", ciplen[i]);
for(i=0; i<len; i++)
    text[i]= decrypt(ciplen[i], n, d); printf("\n");
printf("Decrypted message : ");
    for(i=0; i<len; i++)
     printf("%c", text[i]);
     printf("\n");
    return 0;
}
```

Output:

Enter the text to be encrypted : Hello

Two prime numbers (p and q) are : 13 and 23

n (p*q) = 13*23 = 299

(p-1)*(q-1) = 264

Public key (n, e) = (299, 103)

Private key (n, d) = (299, 223)

Encrypted message: 58 75 147 147 227

Decrypted message: Hello.

Date

Expt. No. 8

Page No. 49

```
/* Diffie-Hellman key exchange algorithm */
#include<stdio.h>
#include<math.h>
long int power(long int a, long int b, long int P)
{ if (b==1) return a;
  else return (((long int) pow(a,b)) % P); }
int main()
{ long int P, G, x, a, b, y, ka, kb;
  P=23; G=9;
  printf("The value of P: %ld\nThe value of G: %ld\n", P, G);
  a=4; b=3;
  printf("The private key a for Alice: %ld\n", a);
  printf("The private key b for Bob: %ld\n", b);
  x=power(G, a, P);
  y=power(G, b, P);
  ka=power(y, a, P);
  kb=power(x, b, P);
  printf("Secret key for Alice is: %ld\n", ka);
  printf("Secret key for Bob is: %ld\n", kb);
  return 0;
}
```

Teacher's Signature

Output:

The value of P: 23
The value of q: 9
The private key a for Alice: 4
The private key b for Bob: 3
Secret key for Alice: 9
Secret key for Bob: 9