Date

Expt. No. 1

Page No.

Implement a client and server communication using sockets programming.

Server

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

void str_echo(int confd)
{
    int n, bufsize = 1024;
    char *buff = malloc(bufsize);

again:
        while ((n=recv(confd, buff, bufsize,0))>0)
            send (confd, buff, n,0);
        if( n < 0)
            goto again;

        free(buff);
}

int main( )
{
```

Teacher's Signature

Expt. No. ______ Date ______ Page No. ______

```
int listenfd, connfd, pid, addrlen;
struct sockaddr_in address, cli_address;

if((listenfd =socket(AF_INET, SOCK_STREAM,0))>0)
   printf("Socket created\n");

address.sin_family=AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons(15001);

if(bind(listenfd,(struct sockaddr*) &address,
                         sizeof(address))==0)
   printf("Binding socket\n");

listen(listenfd, 3);

for (;;)
{
   addrlen= sizeof(struct sockaddr_in);
   connfd = accept(listenfd,(struct sockaddr*)&
                   cli_address, &addrlen);
   if((pid = fork()) ==0)
   {
      close(listenfd); str_echo(connfd);
      exit(0);
   }
   close(connfd);
}
 return 0;
}
```

Teacher's Signature ______

Expt. No. ______ Date ______ Page No. ______

Client

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

void str_cli( FILE *fd, int sockfd)
{
  int bufsize = 1024;
  char * buff = malloc( bufsize);

  while( fgets( buff, bufsize, fp) != NULL)
  {
    send( sockfd, buff, sizeof(buff), 0 );
    if ( recv( sockfd, buff, bufsize, 0) >0 )
        fputs( buff, stdout);
  }

  free( buff);
}

int main( int argc, char * argv[])
{
  int create_socket;
  struct sockaddr_in address;
```

Teacher's Signature ______

Date

Expt. No.

Page No.

```
if((create_socket = socket(AF INET, SOCK-STREAM,0)
                                          )>0)
    printf("Socket created \n");

address. sin_family = AF INET;
address. sin_port = htons (15001);
inet-pton (AF INET, argv[1], &address. sin_addr);

if(connect(create socket,(struct sockaddr *)&address,
                            sizeof(address)) = =0)
    printf("Connection was accepted by server\n");

str-cli( stdin, create_socket);

return close(create_socket); return 0;
}
```

Teacher's Signature

Date

Expt. No. 2

Page No.

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];

void BellmanFord(int s)
{
    int i, u, v;

    for(i=1; i<n; i++){
        for (u=0; u<n; u++){
            for( v=0 ; v<n; v++) {
                if( d[v] > d[u] + A[u][v]){
                    d[v]=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("Negative Edge");
        }
    }
  }
}
```

Teacher's Signature

Expt. No. Date Page No.

```
int main()
{
   printf("Enter no. of edges\n");

   scanf("%d", &n);
   printf("~~%d~~Enter adjacency matrix\n");
   int i, j;
   for( i=0; i<n; i++)
     for( j=0; j<n; j++)
         scanf("%d", & A[i][j]);
  int s;
  for( s=0; s<n; s++)
 {
     for(i=0; i<n; i++)
       {   d[i]= 999;
           p[i] = -1;
       }
     d[s]=0;
     BellmanFord(s);
     printf("Router %d\n", s);
     for(i=0; i<n; i++)
     { if(i!= s)
         {  j=i;
            while(p[j]!=-1){ printf("%d<-", j);
                              j=p[j]; }
         }
         printf("%d\t Cost %d\n\n", s,d[i]);
     }
 }
 return 0
}
```

Teacher's Signature

Expt. No. 3

Date

Page No.

Write a program to implement Error detection and correction concept using Checksum and Hamming code

Checksum Program

```
#include<stdio.h>
~~int~~ unsigned fields[10];
unsigned short checksum( )
{
  int i, sum=0;
  printf("Enter IP header info 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 res1, res2;
  res1 = checksum( );
  printf(" computed checksum at sender %x\n",
         res1);
```

Teacher's Signature

Date

Expt. No.

Page No.

```
    res2 = checksum()
    printf("computed checksum at receiver
              %x\n", res2);

   if(res1==res2)
      printf(" No error");
   else
      printf(" Error in data received\n");
}
```

Hamming Code Program

```
#include<stdlib.h>
#include<stdio.h>
int main()
{
  int a[4], b[4], r[4], s[4], i, q[3], c[7];
  printf("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;
  printf("Enter 7 bit hamming code word:\n");
  for(i=3; i>=0; i--)  printf("%d\t", r[i]);

  printf(" Enter 7 bit received code word:\n");

  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];
```

Teacher's Signature

Expt. No. ______ Date ______

Page No. ______

```
r[0] = c[1];
// calculate syndrome bits

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 is:\n");

for( i = 2; i >= 0; i--)  printf("%d", s[i]);

if((s[2]==0) && (s[1] ==0) && s[0]==0))
printf(" Received word is Error Free\n");

if (s[2]==1) && (s[1]==1) && (s[0]==1))
{
  printf("Error in received codeword, position
  -7th bit from right\n");
  if( c[7] == 0)   c[7] = 1;
  else
       c[7] = 0;
  printf(~~"%d~~ Corrected codeword is\n");

  for ( i=7; i>0; i--)
      printf("%d\t", c[i]);
}
if((s[2] == 1) && (s[1]==1) && (s[0]==0))
{
 printf ("Error in received codeword Position
```

Teacher's Signature ______

Date

Expt. No.

Page No.

```
6th bit from right\n");

if(c[6]==0)    c[6]=1;
else c[6]=0
printf("corrected codeword is\n");
for(i=7; i>0; i--)
    printf("%d\t", c[i]);
}

if((S[2]==1) && (S[1]==0) && S[0]==1))
{
  printf("Error in received codeword. Position
         5th bit from right");
  if(c[5]==0)    c[5]=1;
  else  c[5]=0;
  printf("corrected 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("Error in received codeword
    Position 4th bit from right");
   if(c[4]==0)  c[4]=1;
   else   c[4]=0;
   printf("corrected codeword is \n");
   for(i=7; i>0; i--)
       printf("%d\t", c[i]);
}
```

Teacher's Signature

Expt. No. ____________ Date ____________

Page No. ____________

```
if((S[2]==0) && (S[1]==1) && (S[0]==1))
{
  printf("Error in received codeword.
           Position - 3rd bit from end\n");
  if( C[3]==0)   C[3]=1;
  else     C[3] = 0;
  printf("Corrected codeword is \n");
  for (i=7; i>0; i--)
       printf("%d \t", C[i]);
}

if((S[2]==0 )&&(S[1]==1 )&&(S[0]==0))
{
   printf("Error in received codeword.
            Position - 2nd bit from end\n");
   if (C[2]==0)  C[2]=1;
   else    C[2] = 0;
   printf("Corrected codeword is \n");
    for (i=7; i>0; i--)
       printf("%d\t", C[i]);
}

if((S[2]== 0 )&&(S[1]==0)&&(S[0]==1))
{  printf("Error in received codeword
            Position 1st bit from end\n");
   if (C[1]==0)   C[1]=1;
   else   C[1]=0;
    printf("Corrected codeword is \n");
   for(i=7; i>0; i--)
       printf("%d\t", C[i]);
}
return(0);
}
```

Teacher's Signature ____________

Expt. No. 4

Date

Page No.

Implement a simple multicast routing mechanism.

```
#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"

int main(int argc, char *argv[])
{
    struct sockaddr_in addr;
    int fd, cnt;
    struct ip_mreq mreq;
    char *message = "RVCE-CSE";
    if((fd=socket(AF_INET, SOCK_DGRAM, 0))<0)
    {  perror("socket"); exit(1); }
    addr.sin_port = htons(HELLO_PORT)
    addr.sin_family= AF_INET
    addr.sin_addr.s_addr= inet_addr(HELLO_GROUP)
    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

```
#include< sys/types.h>
#include< sys/ socket.h>
#include<netinet /in.h>
#include< arpa/inet.h>
#include< time.h>
#include< stdio.h>
#include <stdlib.h>
int main()
{  struct sockaddr_in addr;
   int fd, nbytes, addrlen;
   struct ip_mreq mreq;
   char msgbuf[25];
   uint 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);

  addr.sin_family= AF_INET
  addr.sin_addr.s_addr = htonl(INADDRANY);
  addr.sin_port = htons(25);

  if( bind( fd, (struct sockaddr *) &addr, sizeof(
        addr)) <0)
   { perror("bind"); exit(1);}
```

Date

Expt. No.

Page No.

```
mreq. imr_multiaddr.s_addr = inet_addr("255.0.037")
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))
   { perror("recvfrom"); exit(1);}

   puts(msgbuf);
}
}
```

Teacher's Signature

Expt. No. 5 Date

Page No.

Write a program to implement concurrent chat server that allows current logged in users to communica with other.

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

void str_echo(int cfd, int port)
{
  int n,
  int bufsize= 1024;
  char * ~~m~~ buf= malloc(bufsize);

again:
     while( (n=recv(cfd, buf, bufsize,0))>
     {
       ~~(printf(~~ fputs( buf, stdout);
        fgets ( buf, bufsize, stdin);
        send( cfd, buf, bufsize, 0);
     }
   if( n<0)
      goto again;
}
```

Teacher's Signature

Expt. No. ______ Date ______

Page No. ______

```
int main()
{  int listenfd, cfd


   struct sockaddr_in addr, caddr;

   if((listenfd = socket(AF_INET,SOCK STREAM,0))>
   {  printf("socket created\n");
   }
   addr.sin_port = htons(15001);
   addr.sin_family = AF_INET;
   addr.sin_addr.s_addr = INADDR_ANY;

   if( ~~(~~bind(listenfd,(struct sockaddr *)&addr,
      sizeof(addr)) ==0)
    printf("Binding Socket");

  listen(listenfd, 3);
  while(1)
  { addrlen= sizeof(struct sockaddr_in);
   cfd= accept(listenfd,(struct sockaddr *)&caddr,
               &addrlen);

    if((pid = fork()) ==0)
         close(listenfd);
         str_echo(~~confd~~, htons(cliaddress.sin_
    }
    close(cfd);
  }
 return0;
}
```

Teacher's Signature ______

Expt. No. ______ Date ______ Page No. ______

clint.c

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

void str_cli(FILE * fp, int sfd)
{
    int bufsize = 1024; cont;
    char * buf = malloc(bufsize);

    while (fgets(buf, bufsize, fp)!=NULL)
    {
      send(sfd, buf, bufsize, 0);
      if( (cont=recv(sfd, buf, bufsize, 0))>0)
        {
           fputs(buf, stdout);
        }
    }
}
int main(int argc, char * argv[])
{
   int create_socket;
   struct sockaddr_in address;
```

Teacher's Signature ______

Expt. No. Date
Page No.

```
if((create_socket = socket(AF INET, SOCK-STREAM,0)
                                              )>0)
    printf("Socket created \n");

address.sin_family = AF INET;
address.sin_port = htons(15001);
inet-pton(AF INET, argv[1], &address.sin_addr);

if(connect(create socket,(struct sockaddr *)&address,
                         sizeof(address)) = =0)
  printf("Connection was accepted by server\n");

str-cli( stdin, create_socket);

return close(create_socket); return 0;
}
```

Teacher's Signature

6 Implementation of concurrent & iterative echo server using both connection and connectionless socket syste calls.

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

                                                        int cle
void str echo (int sockfd, struct sockaddr* cli_addr,
{     int   n;
      int  bufsize =1024;

      char * buff= malloc(bufsize);
      int addrlen ;

    for(; ;)
    {
      addrlen=clen;
      n= recvfrom(sockfd, buff, bufsize, 0,
                   cli addr, &addrlen);

      sendto(sockfd, buff, n, 0, cli_addr,
                addrlen);
    }
}
```

Expt. No. ______ Date ______ Page No. ______

```
int main( )
{
  int sockfd;
  struct sockaddr_in serv_addr, cli_addr,

  if( (sockfd =socket(AF_INET, SOCK_DGRAM,0))>0)
     printf("Socket created\n");

  serv_addr. sin_family = AF_INET;
  serv_addr. sin_addr. s_addr = INADDR_ANY;
  serv_addr. sin_port = htons (15001);

  if( bind( sockfd, (struct sockaddr *)&serv_addr,
            sizeof (serv_addr))==0)
    printf (" Binding socket\n");

  str_echo(sockfd, (struct sockaddr *)& cli_address,
            sizeof(cli_address));

  return 0;
}


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

Teacher's Signature ______

Expt. No. ______________ Date ______________

Page No. ______________

```
void str_cli(FILE * fp, int sfd, struct sockaddr *
                    serv_addr, int servlen)
{ int bufs =1024, cont; char * buff= malloc(bufs);
  int addrlen= sizeof(struct sockaddr_in);

  while( fgets(buff, bufs, fp)!=NULL)
  {
   sendto(sockfd, buff, sizeof(buff),0,& serv_addr, servlen);

   if((cont=recvfrom (sockfd, buff, bufs, 0,
                  NULL,NULL)>0))
      {
        fputs(buff, stdout);
      }
  }
}

int main(int argc, char * argv[])
{ int sfd; struct sockaddr_in serv_address;
  if( ~~sockfd~~ = socket(AF_INET, SOCK_DGRAM,0))>0)
     printf("Socket created");
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_port = htons(16001);
  inet_pton( AF_INET, argv[1], & serv_address.sin_addr

  str_cli(stdin, ~~sock~~ sfd, (struct sockaddr *) &
              serv_address, sizeof(serv_address));
  exit(0);
}
```

Teacher's Signature ______________

Date

Expt. No.

Page No.

Iterative_Server.c

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

void str_echo(int confd)
{
   int n, bs = 1024, len,
   char * buf = malloc(bs);
   struct sockaddr_in addr;
   again: while((n=recv(confd, buf, bs, 0))>0)
               send(confd, buf, n, 0);

          if(n<0)
               goto again;
}

int main()
{
   int lfd, cfd, addrlen,
   struct sockaddr_in address;

   if((lfd = socket(AF_INET, SOCK_STREAM, 0))>0)
        printf("socket created\n");
```

Teacher's Signature

Expt. No.

Date

Page No.

```
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons(15001);

if( bind( lfd, (struct sockaddr *) &address,
                        sizeof(address)) == 0)
   printf(" Binding socket\n");

~~if( 0 lfd~~ listen( lfd, 3);

for(;;)
{
 addrlen = sizeof(struct sockaddr_in);
 cfd = accept( lfd, (struct sockaddr *) & address,
                                  &addrlen);
 printf(" Connection accepted from
        client %s\n", inet_ntoa(address.sin_addr))

 str_echo(cfd);

 close(cfd);
}
return 0;
}
```

Teacher's Signature

Expt. No. ______ Date ______ Page No. ______

Client

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

void str_cli( FILE *fd, int sockfd)
{
  int bufsize = 1024;
  char * buff = malloc( bufsize);

  while( fgets( buff, bufsize, fp) != NULL)
  {
    send( sockfd, buff, sizeof(buff), 0);
    if ( recv( sockfd, buff, bufsize, 0)>0)
        fputs( buff, stdout);
  }

  free( buff);
}

int main( int argc, char * argv[])
{
  int create_socket;
  struct sockaddr_in address;
```

Teacher's Signature ______

Expt. No. ______ Date ______

Page No. ______

```
if((create_socket = socket(AF INET, SOCK-STREAM,0)
                                              )>0)
    printf("Socket created \n");

address.sin_family = AF INET;
address.sin_port = htons(15001);
inet-pton(AF INET, argv[1], &address.sin_addr);

if(connect(create socket,(struct sockaddr *)&address,
                         sizeof(address)) = =0)
   printf("Connection was accepted by server\n");

str-cli( stdin, create_socket);

return close(create_socket); return 0;
}
```

Teacher's Signature ______

Expt. No. 7 Date Page No.

Implementation of remote command execution using socket system calls

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

void str_echo(int confd)
{
    int n, bufsize = 1024, len;
    char * buff = malloc( bufsize);
    struct sockaddr_in addr;
    again: while( (n= recv(confd, buff, bufsize,0))>0)
          {
              system( buff);
          }
    if( n <0)
          goto again;
}

int main( )
{
    int listenfd, confd, addrlen, pid;
```

Teacher's Signature

Expt. No. ____________ Date ____________

Page No. ________

```
    struct sockaddr_in address.
    if((listenfd=socket(AE INET, SOCK STREAM, 0)>0)
         printf("Socket created\n");

   address.sin_family = AF_INET;
   address.sin_port = htons(15001);
   address.sin_addr.s_addr= INADDR_ANY;

  if( bind(listenfd, (struct sockaddr *) saddress,
                           sizeof(address))==0)
     printf("Binding socket\n");

  listen(listenfd, 3)

  for(;;)
{
  addrlen = sizeof( struct sockaddr_in);
  confd = accept(listenfd, (struct sockaddr*)saddress,
                                    saddrlen);

  if((pid = fork())==0)
   {
     close(listenfd);
     str_echo(confd);
     exit(0);
   }
  close(confd);
}
 return 0;
}
```

Teacher's Signature ____________

Expt. No. ______ Date ______

Page No. ______

Client
```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <netinet/in.h>
void str_cli ( FILE *fp, int sockfd)
{
     int  bufs =1024, cont;
     char * buff = malloc ( bufs);
     while ( (fgets (buff, bufs, fp ) != NULL))
       {
          send (sockfd, buff, sizeof (bufs), 0);
       }
}

int main (int argc, char *argv[])
{
   int  cs, ret;
   struct  sockaddr_in address;

   if( (cs = = socket(AF INET, SOCK_STREAM,0))>0)
      printf(" Socket created \n");

   address. sin_family = AF INET;
```

Teacher's Signature ______

Expt. No. ______ Date ______

Page No. ______

```
address.sin_port = htons (15001)
inet_pton(AF INET, argv[1], &address.sin_addr);
if(((ret = connect (cs,(struct sockaddr *)&address,
                  sizeof (address))) == 0)
    printf("connected \n");
str_cli( stdin, cs);
return 0;
}
```

Date

Expt. No. 8

Page No.

Write a program to encrypt and decrypt the data using RSA and Exchange the key securely using Diffe-Hellman Key exchange protocol.

```
#include< stdio.h>
#include< stdlib.h>
#include< math.h>
#include < string.h>
~~#include <)~~

long ~~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
   }
  return1;
}
```

Teacher's Signature

Expt. No. ______________ Date ______________

Page No. ______________

```
long int encrypt( ~~long int~~ char ch, long int n,
                          long int e)
{  int i;
   long int temp=ch;
   for( i=1; i<e; i++)
       temp =(temp * ch)% n;
   return temp;
}
char decrypt(long int ch, long int n, long int d)
{
    int i;
    long int temp=ch;
    for( i=1; i<d; i++)
        ch= (temp * ch) % n;
    return ch;
}

int main()
{
  long int i, len;
  long int p, q, n, phi, e, d, cipher[50];
  char text[50];
  cout<<"Enter text for encryption";
  cin.getline(text, sizeof(text));
  len= strlen(text);
  do {   p = rand() % 50;
     } while( !isprime(p));

  do {   q = rand() % 50;
```

Teacher's Signature ______________

Expt. No. ____________ Date ____________

Page No. ____________

```
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);

cout<<"Two prime no's are :"<<p << q<<endl;
cout<< " n = p*q " << p*q << endl;
coup<< "phi =(p-1)*(q-1)"<< phi<< endl;

cout<< "Public Key(n, e):"<< n << e<< endl;
cout<< "Private Key(n, d):"<<n <<d<< endl;

for (i = 0; i<len; i++)
  cipher [i] = encrypt (text[i], n, e);

cout << "Encryped message"<< endl;

for (i = 0; i< len; i++) cout<< cipher[i];

for ( i = 0; i < len; i++)
  text [i] = decrypt ( cipher [i], n, d);

cout << endl;
```

Teacher's Signature ____________

Expt. No. ________ Date ________

Page No. ________

```
cout<<"Decryped messagi"<<endl;
for(i=0; i<len; i++)
    cout<< text[i];

cout<<endl;

return 0;
}
```

Diffie - Hellman Key Exchange

```
#include<stdio.h>
#include<math.h>
long long int power(long long int a,
  long long int b, long long int P)
{
  if( b==1) return a;
 else
    return(((long long int)power(a,b))%P);
}

int main()
{
 long long int P, G, x, a, y, b, ka, kb;
 P=23.
 printf("Enter the value of P");
 scanf("%lld", &P);

 G=9;
```

Teacher's Signature ________

Expt. No. 3.

Date

Page No.

```
printf ("Enter the value of G:");
scanf ("%lld", &G);

a = 4;
printf("Private Key for Alice :%lld", a);
x = power (G, a, P);

b = 3;
printf("Private Key for Bob: %lld", b);
y = power (G, b, P);

ka = power( y, a, P);
kb = power( x, b, P);

printf("Secret Key for Alice :%lld\n", ka);
printf("Secret Key for Bob :%lld\n", kb);

return 0;
}
```

Teacher's Signature