

NATIONAL INSTITUTE OF TECHNOLOGY MANIPUR

(An Autonomous Institute under MHRD, Govt. of India)



COMPUTER NETWORK LAB (CS336) **VI Semester**

Submitted by:

Name: Nongmaithem Miranda Devi

Enrollment Number: 21103003

Submitted to: Nirvana Mam

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY MANIPUR

EXPERIMENT 1

AIM: To write a C program for error detecting code using CRC-CCITT (16bit).

SOURCE CODE:

```
#include<stdio.h>

int a[100],b[100],i,j,len,k,count=0;
int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,};

int main(){
    void div();

    printf("\nEnter the length of Data Frame:");
    scanf("%d",&len);

    printf("\nEnter the message :");
    for(i=0;i<len;i++)
        scanf("%d",&a[i]);
    for(i=0;i<16;i++)
        a[len++]=0;
    for(i=0;i<len;i++)
        b[i]=a[i];
    k=len-16;
    div();
    for(i=0;i<len;i++)
        b[i]=b[i]^a[i];
    printf("\nData to be transmitted: ");
    for(i=0;i<len;i++)
        printf("%2d",b[i]);
    printf("\n\nEnter the Received Data: ");
    for(i=0;i<len;i++)
        scanf("%d",&a[i]);
    div();
    for(i=0;i<len;i++)
        if(a[i]!=0)
        {
```

```

        printf("\nERROR in Received Data");
        goto END;
    }

    printf("\nData Received is ERROR FREE");
    END: printf("\nRemember is : ");
    for(i=(len-16);i<len;i++)
        printf("%d",a[i]);
    printf("\n");
}

void div()
{
    for(i=0;i<k;i++)
    {
        if(a[i]==gp[0])
        {
            for(j=i;j<17+i;j++)
                a[j]=a[j]^gp[count++];
        }
        count=0;
    }
}

```

OUTPUT:

```

PS C:\Users\Miranda\Desktop\practice\cnlab> gcc crc.c
PS C:\Users\Miranda\Desktop\practice\cnlab> ./a.exe

Enter the length of Data Frame:4

Enter the message :1 0 1 1

Data to be transmitted: 1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 1

Enter the Received Data: 1 0 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 1

ERROR in Received Data
Remember is : 0000000100000000

```

EXPERIMENT 2

AIM: To write a C program for frame sorting technique using buffers.

SOURCE CODE:

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int count=0,i,j,total,size,num,tsize;

char val[100];

void frames();

void trans();

void sort();


struct frame
{
    int seq;

    int len;

    int flag;

    char data[20];
}n[20],m[20],temp;

int main(){

    printf("\nEnter the data:");

    scanf("%s",val);

    tsize=strlen(val);

    frames();

    trans();

    sort();

}

void frames(){

    int ch;

    total=0;

    printf("\nSegmented Frames and transmitting order at Sender: \n") ;
```

```

while(total<tsize){
    n[count].seq=count+1;
    size=rand()%10+1;
    n[count].len=size;
    n[count].flag=0;
    j=0;
    if((total+size)<tsize){
        for(i=total,j=0;i<total+size,j<size;i++,j++)
            n[count].data[j]=val[i];
        n[count].data[j]='\0';
    }
    else{
        n[count].len=tsize-total;
        j=0;
        while(total<tsize){
            n[count].data[j++]=val[total];
            total++;
        }
        n[count].data[j]='\0';
    }
    total+=size;
    count++;
}
printf("\nFseq No.  Flen  FFlag  FData \n");
for(i=0;i<count;i++){
    printf("  %d\t   %d\t%d\t",n[i].seq,n[i].len,n[i].flag);
    for(j=0;j<n[i].len;j++)
        printf("%c",n[i].data[j]);
    printf("\n");
}
num=count;

```

```

printf("\n ENter 1/0 to continue :");
scanf("%d",&ch);
if(ch==0)
    exit(0);
}
void trans(){
    int ch;
    count=0;
    printf("\nBegins : \n");
    while(count<num){
        i=rand()%num;
        if(n[i].flag==0)
            m[count++]=n[i];
        n[i].flag=1;
    }
    printf("\nThe order of frames received at receiving terminal :\n");
    printf("\nFSeq No. FLen  Flag  FData\n");
    for(i=0;i<count;i++){
        printf(" %d\t %d\t%d\t",m[i].seq,m[i].len,m[i].flag);
        for(j=0;j<m[i].len;j++)
            printf("%c",m[i].data[j]);
        printf("\n");
    }
    printf("\n Total NO of frames: %d",count);
    printf("\nENter 1/0 to continue: ");
    scanf("%d",&ch);
    if(ch==0)
        exit(0);
}
void sort(){
    for(i=0;i<count;i++)

```

```

for(j=i+1;j<count;j++)

    if(m[i].seq>m[j].seq){

        temp=m[i];

        m[i]=m[j];

        m[j]=temp;

    }

printf("\nSorted frames at receiving terminal :\n");

printf("\nFseq No.  FLen  Flag  Fdata\n");

for(i=0;i<count;i++){

    printf("%d\t %d\t%d\t",m[i].seq,m[i].len,m[i].flag);

    for(j=0;j<m[i].len;j++)

        printf("%c",m[i].data[j]);

    printf("\n");

}

}

```

OUTPUT:

```

root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# ./a.out
Enter the data:abcdefghijjkkllmmnnoopppqrrssttuuvv
Segmented Frames and transmitting order at Sender:
Fseq No.  FLen  Flag  FData
1          4  0      abcd
2          7  0      efghijj
3          8  0      kkllmmnn
4          6  0      ooppqq
5          4  0      rrss
6          6  0      ttuuvv

Enter 1/0 to continue :1
Begins :
The order of frames received at receiving terminal :
FSeq No.  FLen  Flag  FData
5          4  0      rrss
1          4  0      abcd
4          6  0      ooppqq
2          7  0      efghijj
3          8  0      kkllmmnn
6          6  0      ttuuvv

Total NO of frames: 6
ENTER 1/0 to continue: 1
Sorted frames at receiving terminal :
Fseq No.  FLen  Flag  Fdata
1          4  0      abcd
2          7  0      efghijj
3          8  0      kkllmmnn
4          6  0      ooppqq
5          4  0      rrss
6          6  0      ttuuvv

```

EXPERIMENT 3

AIM To write a C program for distance vector algorithm to find suitable path for transition.

SOURCE CODE:

```
#include<stdio.h>

struct node {
    unsigned dist[20];
    unsigned from[20];
} rt[20];

int main() {
    int costmat[20][20];
    int nodes, i, j, k, count=0;
    printf("Enter the number of nodes: ");
    scanf("%d", &nodes);
    printf("\nEnter the cost matrix:\n");
    for(i=0; i<nodes; i++) {
        for(j=0; j<nodes; j++) {
            scanf("%d", &costmat[i][j]);
            costmat[i][i]=0;
            rt[i].dist[j] = costmat[i][j];
            rt[i].from[j] = j;
        }
    }
    do {
        count = 0;
        for(i=0; i<nodes; i++) {
            for(j=0; j<nodes; j++) {
                for(k=0; k<nodes; k++) {
                    if(rt[i].dist[j] > costmat[i][k] + rt[k].dist[j]) {
                        rt[i].dist[j] = rt[i].dist[k] + rt[k].dist[j];
                        rt[i].from[j] = k;
                        count++;
                    }
                }
            }
        }
    } while(count > 0);
}
```



```

        }
    }
}

} while(count != 0);

for(i=0; i<nodes; i++) {
    printf("\n\nFor router %d:\n", i+1);
    for(j=0; j<nodes; j++) {
        printf("\tNode %d via %d Distance %d\n", j+1, rt[i].from[j]+1, rt[i].dist[j]);
    }
}

printf("\n");
return 0;
}

```

OUTPUT:

```

PS C:\Users\Miranda\Desktop\practice\cnlab> gcc dv.c
PS C:\Users\Miranda\Desktop\practice\cnlab> ./a.exe
Enter the number of nodes: 4

Enter the cost matrix:
999 5 8 3
5 999 2 999
8 2 999 6
3 999 6 999

For router 1:
Node 1 via 1 Distance 0
Node 2 via 2 Distance 5
Node 3 via 2 Distance 7
Node 4 via 4 Distance 3

For router 2:
Node 1 via 1 Distance 5
Node 2 via 2 Distance 0
Node 3 via 3 Distance 2
Node 4 via 1 Distance 8

For router 3:
Node 1 via 2 Distance 7
Node 2 via 2 Distance 2
Node 3 via 3 Distance 0
Node 4 via 4 Distance 6

For router 4:
Node 1 via 1 Distance 3
Node 2 via 1 Distance 8
Node 3 via 3 Distance 6
Node 4 via 4 Distance 0

```

EXPERIMENT 4

AIM: To write a C program for demonstrating ARP and RARP protocol.

SOURCE CODE:

//Server Side

```
#include<stdio.h>

#include<sys/types.h>

#include<sys/shm.h>

#include<string.h>

int main(){

    int shmid,a ,i;

    char *ptr,*shmptr;

    shmid=shmget(3000,10,IPC_CREAT | 0666);

    shmptr=shmat(shmid,NULL,0);

    ptr=shmptr;

    for(i=0;i<3;i++){

        puts("ENter the mac: \n");

        scanf("%s",ptr);

        a=strlen(ptr);

        printf("string length: %d",a);

        ptr[a]=' ';

        puts("ENter the IP: ");

        ptr=ptr+a+1;

        scanf("%s",ptr);

        ptr[a]='\n';

        ptr=ptr+a+1;

    }

    ptr[strlen(ptr)]='\0';

    printf("\n ARP table at sever side is =\n%s",shmptr);

    shmdt(shmptr);

}
```

//Client side

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<sys/shm.h>
#include<stdlib.h>

int main(){
    int shmid,a;
    char *ptr,*shmptr;
    char ptr2[51],ip[12],mac[26];
    shmid=shmget(3000,10,0666);
    shmptr=shmat(shmid,NULL,0);
    puts("the arp table is");
    printf("%s",shmptr);
    printf("\n1.ARP \n2.RARP\n 3.EXIT\n");
    scanf("%d",&a);
    switch(a){
        case 1:
            puts("enter ip address");
            scanf("%s",ip);
            ptr=strstr(shmptr,ip);
            ptr-=8;
            sscanf(ptr,"%s%s",ptr2);
            printf("mac addr is %s",ptr2);
            break;
        case 2:
            puts("enter mac addr");
            scanf("%s",mac);
            ptr=strstr(shmptr,mac);
            sscanf(ptr,"%s%s",ptr2);
```

```

        break;

        case 3:

            exit(1);

    }

}

```

OUTPUT:

```

root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# cc arpServer.c
root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# ./a.out
Enter the mac:

a.b.c.d
string length: 7Enter the IP:
1.2.3.4
Enter the mac:

b.c.d.e
string length: 7Enter the IP:
2.3.4.5
Enter the mac:

c.d.e.f
string length: 7Enter the IP:
3.4.5.6

  ARP table at sever side is =
a.b.c.d 1.2.3.4
b.c.d.e 2.3.4.5
c.d.e.f 3.4.5.6
root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# cc arpClient.c
root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# ./a.out
the arp table is
a.b.c.d 1.2.3.4
b.c.d.e 2.3.4.5
c.d.e.f 3.4.5.6

1.ARP
2.RARP
3.EXIT
1
enter ip address
2.3.4.5
mac addr is b.c.d.eroot@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab#

```

EXPERIMENT 5

AIM: To write a client-server program using TCP/IP sockets to make client sending the file name and the server to send back the contents of the requested file if present.

SOURCE CODE:

// TCP/IP_server program

```
#include <unistd.h>
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
```

```
#include <netinet/in.h>
```

```
#include <arpa/inet.h>
```

```
#include <fcntl.h>
```

```
#include <string.h>
```

```
#define SERV_TCP_PORT 6880 // Corrected the definition of SERV_TCP_PORT
```

```
#define SERV_HOST_ADDR "127.0.0.1"
```

```
int main() {
```

```
    int sockfd, newsockfd, clien;
```

```
    struct sockaddr_in cli_addr, serv_addr;
```

```
    char filename[25], buf[1000]; // Corrected the declaration of filename
```

```
    int n, m = 0;
```

```
    int fd;
```

```
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
```

```
        printf("server: can't open stream socket\n");
```

```
    else
```

```
        printf("server: stream socket opened successfully\n");
```

```
    serv_addr.sin_family = AF_INET;
```

```

serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);

serv_addr.sin_port = htons(SERV_TCP_PORT); // Corrected the usage of SERV_TCP_PORT

if ((bind(sockfd, (struct sockaddr*) &serv_addr, sizeof(serv_addr))) < 0)
    printf("server: can't bind local address\n");
else
    printf("server: bound to local address\n");

listen(sockfd, 5); // Added backlog size for listen

printf("\nSERVER: waiting for client...\n");

for (;;) {
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr*) &cli_addr, &clilen);

    if (newsockfd < 0)
        printf("server: accept failed\n");
    else {
        printf("server: accepted\n");
        n = read(newsockfd, filename, 25);
        filename[n] = '\0';
        printf("\nSERVER: %s is found and ready to transfer\n", filename);

        fd = open(filename, O_RDONLY);
        if (fd < 0) {
            perror("open");
            close(newsockfd);
            continue; // Continue to the next iteration of the loop
        }
    }
}

```

```

while ((n = read(fd, buf, 1000)) > 0) {
    if (write(newsockfd, buf, n) < 0) {
        perror("write");
        break; // Break out of the loop on write error
    }
}

if (n < 0)
    perror("read");

printf("\nTransfer complete\n");
close(fd);
close(newsockfd);
}
}

return 0;
}

```

// TCP/IP client side program

```

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

#define SERV_TCP_PORT 6880

```

```
#define SERV_HOST_ADDR "127.0.0.1"
```

```
int main() {
```

```
    int sockfd;
```

```
    struct sockaddr_in serv_addr;
```

```
    char filename[100], buf[1000];
```

```
    int n;
```

```
    serv_addr.sin_family = AF_INET;
```

```
    serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
```

```
    serv_addr.sin_port = htons(SERV_TCP_PORT);
```

```
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
```

```
        printf("Client: can't open stream socket\n");
```

```
        exit(1); // Exiting program if socket creation fails
```

```
    } else {
```

```
        printf("Client: Stream socket opened successfully\n");
```

```
    }
```

```
    if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
```

```
        printf("Client: can't connect to server\n");
```

```
        exit(1); // Exiting program if connection fails
```

```
    } else {
```

```
        printf("Client: connected to server successfully\n");
```

```
    }
```

```
    printf("\nEnter the filename to be displayed: ");
```

```
    scanf("%s", filename);
```

```
    write(sockfd, filename, strlen(filename));
```

```
    printf("\nFilename transferred to server\n");
```



```

n = read(sockfd, buf, sizeof(buf)); // Changed buffer size to sizeof(buf)

if (n < 0) {
    printf("\nError reading from socket\n");
    exit(1); // Exiting program if read error occurs
}

printf("\nClient: Displaying file content of %s\n", filename);
printf("%.s", n, buf); // Printing only the received bytes
close(sockfd);
exit(0);
}

```

OUTPUT:

```

root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# cc tcpserver.c
root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# ./a.out
server: stream socket opened successfully
server: bound to local address

SERVER: waiting for client...
server: accepted

SERVER: text.txt is found and ready to transfer

Transfer complete

```

```

root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# cc tcpclient.c
root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# ./a.out
Client: Stream socket opened successfully
Client: connected to server successfully

Enter the filename to be displayed: text.txt

Filename transferred to server

Client: Displaying file content of text.txt
hello world, good day.
root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# _

```

EXPERIMENT 6

AIM: To write a C program to develop a DNS client server to resolve the given hostname.

SOURCE CODE:

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<sys/shm.h>

#include<sys/types.h>

#include<netdb.h>

#include<errno.h>

#include<netinet/in.h>

#include<arpa/inet.h>

int main(){

    char hostname[100];

    printf("enter the host name:");

    fgets(hostname,sizeof(hostname),stdin);

    hostname[strcspn(hostname,"\n")]=0;

    struct hostent *hen;

    hen=gethostbyname(hostname);

    if(hen==NULL){

        fprintf(stderr,"host not found\n");

        return 1;

    }

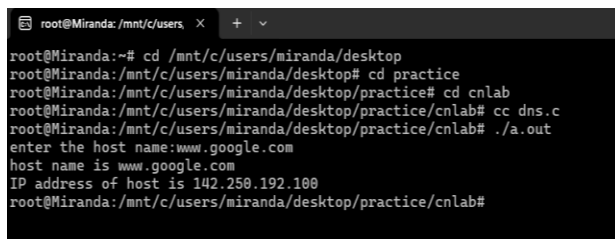
    printf("host name is %s\n",hen->h_name);

    printf("IP address of host is %s\n",inet_ntoa(*(struct in_addr*)hen->h_addr));

    return 0;

}
```

OUTPUT:



```
root@Miranda: /mnt/c/users, X + v
root@Miranda:~# cd /mnt/c/users/miranda/desktop
root@Miranda:/mnt/c/users/miranda/desktop# cd practice
root@Miranda:/mnt/c/users/miranda/desktop/practice# cd cnlab
root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# cc dns.c
root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab# ./a.out
enter the host name:www.google.com
host name is www.google.com
IP address of host is 142.250.192.100
root@Miranda:/mnt/c/users/miranda/desktop/practice/cnlab#
```

EXPERIMENT 7

AIM: To write a C program for client server communication using message queues or FIFO as IPC channels that client sends the file name and the server to send back the contents of the requested file if present.

SOURCE CODE:

//Server side

```
#include<stdio.h>

#include<fcntl.h>

#include<stdlib.h>

#include<string.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<unistd.h>

int main(){

    char filename[100],buf[300],bufl[300];

    int num,num2,n,filesize,f1,fd,fd2;

    mknod("fifo1",S_IFIFO|0666,0);

    mknod("fifo2",S_IFIFO|0666,0);

    printf("\nServer online\n");

    fd=open("fifo1",O_RDONLY);

    printf("Client online!Waiting for request...\n\n");

    while(1){

        num=read(fd,filename,100);

        filename[num]='\0';

        f1=open(filename,O_RDONLY);

        printf("\n Server : %s is found!\n transferring the contents\n",filename);

        filesize=lseek(f1,0,2);

        printf("\nFile size is %d\n",filesize);

        lseek(f1,0,0);

        n=read(f1,bufl,filesize);

        buf[n]='\0';
```

```

        fd2=open("fifo2",O_WRONLY);

        write(fd2,buf1,strlen(buf1));

        printf("\n SERVER:Transfer completed\n");

        exit(1);

    }

    unlink("fifo1");

    unlink("fifo2");
}

//Client side

#include<stdio.h>

#include<fcntl.h>

#include<string.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<unistd.h>

int main(){

    char filename[100],buf[300];

    int num,num2,fl,fd,fd2;

    mknod("fifo1",S_IFIFO | 0666,0);

    mknod("fifo2",S_IFIFO | 0666,0);

    fd=open("fifo1",O_WRONLY);

    printf("Client Online! \n CLIENT enter the path...\n\n");

    scanf("%s",filename);

    write(fd,filename,strlen(filename));

    printf("\n waiting for reply...\n");

    fd2=open("fifo2",O_RDONLY);

    num2=read(fd2,buf,300);

    buf[num2]='\0';

    printf("\n File received..The contents are...\n");

```

```
fputs(buf,stdout);  
unlink("fifo1");  
unlink("fifo2");  
exit(1);  
}
```

OUTPUT:

```
Server online  
Client online!Waiting for request...  
  
Server : data.txt is found!  
transferring the contents  
  
File size is 23
```

```
Client Online!  
CLIENT enter the path...  
  
data.txt  
  
waiting for reply...  
  
File received..The contents are...  
Hello hello World!
```

EXPERIMENT 8

AIM: To write a C program for simple RSA algorithm to encrypt and decrypt the data.

SOURCE CODE:

```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<string.h>

int gcd(long m,long n){
    while(n!=0){
        long r=m%n;
        m=n;
        n=r;
    }
    return m;
}

int rsa(char *message){
    int p=0,q=0,n=0,phi=0,e=0,d=0;
    long nummes[100]={0};
    long encrypted[100]={0},decrypted[100]={0};
    long i=0,j=0,nofelem=0;
    printf("\nEnter the values of p and q: \n");
    //step1:
    scanf("%d %d",&p,&q);
    //step2:
    n=p*q;
    //step3:
    phi=(p-1)*(q-1);
    //step4:find e s.t 1<e<phi,e and phi must be co primes
    for(i=2;i<phi;i++)
        if(gcd(i,phi)==1)
            break;
```

```

e=i;
//step5: find d, d*e*mod(phi)=1
for(i=2;i<phi;i++)
    if((e*i-1)%phi==0)
        break;
d=i;
//step6: find encrypted message
//C=M^e*mod(n)
//Message which is in characters,convert them into nums
for(i=0;i<strlen(message);i++)
    nummes[i]=message[i]-96;
nofelem=strlen(message);

//Start encryption
for(i=0;i<nofelem;i++){
    encrypted[i]=1;//initialization
    for(j=0;j<e;j++)
        encrypted[i]=(encrypted[i]*nummes[i])%n;
}
printf("The encrypted message is: \n");
for ( i = 0; i < nofelem; i++)
{
    printf("%ld",encrypted[i]);
    printf("%c",(char)(encrypted[i]+96));
}
for(i=0;i<nofelem;i++){
    decrypted[i]=1;
    for(j=0;j<d;j++){
        decrypted[i]=(decrypted[i]*encrypted[i])%n;
    }
}

```

```

    printf("\nThe decrypted message is: \n");
    for ( i = 0; i < nofelem; i++)
    {
        printf("%c",(char)(decrypted[i]+96));
    }
    return 0;
}

int main(){
    char *msg;

    printf("Enter the message to be encrypted: \n");
    scanf("%s",msg);

    rsa(msg);

    return 0;
}

```

OUTPUT:

```

PS C:\Users\Miranda\Desktop\practice\cnlab> gcc rsa.c
PS C:\Users\Miranda\Desktop\practice\cnlab> ./a.exe
Enter the message to be encrypted:
hello

Enter the values of p and q:
5
7
The encrypted message is:
8h10j17q17q15o
The decrypted message is:
hello

```


EXPERIMENT 9

AIM: To write a C program for Hamming code generation for error detecting and correction.

SOURCE CODE:

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int mm[30];

int main()
{
    int b[10],n,k=0,i,m,s[10],j=0,flag=0,l,len;
    int ll=1,kk=0,jj=0;
    int a[4][8]={
        {0,0,0,0,0,0,0,0},
        {0,1,0,1,1,1,0,0},
        {0,1,1,0,1,0,1,0},
        {0,0,1,1,1,0,0,1}
    };

    char mess[40];
    system("clear");
    printf("Enter the message: \n");
    scanf("%s", &mess);
    len=strlen(mess);
    for(i=0;i<len;i++)
    {
        k=mess[i];
        k%=16;
        while (k!= 0)
        {
            mm[jj++]=k%2;
```

```

        k = k / 2 ;

    }
    while(jj%4!=0)
        mm[jj++]=0;
}

k = 0;
while(kk<jj)
{
    do
    {
        b[l++]=mm[kk++];
    }while(kk%4!=0);
    l=1;

    b [5]=( b[1] + b[3] +b[4])%2;
    b [6]=( b[1] + b[2] +b[4])%2;
    b[7]=(b[2]+b[3]+b[4])%2;
    printf("\ncodeword of %c\n", mess [k]);
    for(i=1;i<=7;i++)
        printf("%d",b[i]);
    printf("\n");
    printf("\nEnter the codeword received by receiver\n");

    for(i=1;i<=7;i++)
        scanf("%d",&b[i]);
    b[0]=0;
    for(m=0;m<=3;m++)
        s[m]=0;

```

```

m=0;
for(i=0;i<=3;i++)
{
    for(j=0;j<=7;j++)
    {
        s[m]=s[m]+(a[i][j]*b[j]);
        s[m]=s[m]%2;

    }
    m++;

}
for(i=1;i<=3;i++)
    printf("%d",s[i]);
printf("\n");
for(i=1;i<=7;i++)
{
    for(j=1;j<=3;j++)
    {
        if(s[j]==a[j][i])
            flag=1;
        else{
            flag=0;
            break;

        }
    }
}
if(flag)
{
    printf("error is at %d\n",i);
    if(b[i])

```

```

        b[i]=0;
    else
        b[i]=1;

    }
}

printf("corrected codeword for %c\n",mess[k]);

for(i=1;i<=7;i++)

    printf("%d",b[i]);

printf("\n");

k++;

}

}

```

OUTPUT:

```

PS C:\Users\Miranda\Desktop\practice\cnlab> gcc hamming.c
PS C:\Users\Miranda\Desktop\practice\cnlab> ./a.exe
'clear' is not recognized as an internal or external command,
operable program or batch file.
Enter the message:
Hi

codeword of H
0001111

Enter the codeword received by receiver
0 0 0 1 1 1 1
000
corrected codeword for H
0001111

codeword of i
1001001

Enter the codeword received by receiver
1 0 0 0 0 0 1
111
error is at 4
corrected codeword for i
1001001

```

EXPERIMENT 10

AIM: To write a C program for congestion control using leaky Bucket algorithm

SOURCE CODE:

```
#include<strings.h>

#include<stdio.h>

int min(int x, int y)
{
    if (x<y)
        return x;
    else
        return y;
}

int main()
{
    int drop= 0,mini, nsec, cap, count= 0,i,inp[25],process;
    printf("Enter The Bucket Size\n");
    scanf ("%d", &cap);

    printf("Enter The Operation Rate\n");
    scanf("%d", &process);
    printf("Enter The No. Of Seconds You Want To Stimulate\n");
    scanf("%d", &nsec);
    for(i=0;i<nsec;i++)
    {
        printf("Enter The Size Of The Packet Entering At %d sec\n", i+1);
        scanf("%d",&inp[i]);
    }
    printf("\nSecond | Packet Recieved | Packet Sent | Packet Left | Packet Dropped |\n");
    printf("----- \n");
    for(i=0;i<nsec;i++)
    {
```

```

count+=inp[i];
if (count>cap)
{
drop=count-cap;
count=cap;
}
printf("%d",i+1);
printf("\t%d", inp[i]);
mini=min (count, process);
printf("\t\t%d", mini);
count=count-mini;
printf("\t\t%d",count);
printf("\t\t%d\n", drop);
drop=0;
}
for(;count !=0;i++)
{
    if(count>cap)
    {
        drop=count-cap;
        count=cap;
    }
printf("%d",i+1);
printf("\t0");
mini=min(count, process);
printf("\t\t%d", mini);
count= count- mini;
printf("\t\t%d", count);
printf("\t\t%d\n", drop);
}
}

```

OUTPUT:

```
PS C:\Users\Miranda\Desktop\practice\cnlab> gcc leaky.c
```

```
PS C:\Users\Miranda\Desktop\practice\cnlab> ./a.exe
```

```
Enter The Bucket Size
```

```
5
```

```
Enter The Operation Rate
```

```
2
```

```
Enter The No. Of Seconds You Want To Stimulate
```

```
3
```

```
Enter The Size Of The Packet Entering At 1 sec
```

```
5
```

```
Enter The Size Of The Packet Entering At 2 sec
```

```
4
```

```
Enter The Size Of The Packet Entering At 3 sec
```

```
3
```

```
Second| Packet Recieved| Packet Sent | Packet Left | Packet Dropped|
```

```
-----|-----|-----|-----|-----
```

```
1      5          2          3          0
```

```
2      4          2          3          2
```

```
3      3          2          3          1
```

```
4      0          2          1          0
```

```
5      0          1          0          0
```