Networking Lab
(PCC-CS-692)

Assignment for Hamming Code

1. **Write a C code to encode a user provided dataword using Hamming Code procedure.**

Test case 1:

Input:
Dataword: 1100101

Output:

No of redundant bits: 4
The codeword is: 11000101100

Test case 2:

Input:
Dataword: 1011010

Output:

No of redundant bits: 4
The codeword is: 10101010000

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

int main()
    {
        // *************START  CODE FOR hamming SENDER ************//
    char data[100];
    int data1[100],data2[100];
    int dl,r,i=0,j=0,k=0,z,c;

    printf("\n Enter the dataword: "); //taking input in string
    scanf("%s",data);
    dl=strlen(data);        //length of the input string

    while(1)        //finding number of parity bits
    {
        if(pow(2,i)>=dl+i+1)
            break;
        i++;
    }
    r=i;    //storing number of parity bits into r variable
    printf("\n No of redundant bits: %d \n",r);

    for(i=0;i<dl;i++)       //conversion of string data into integer
    {
```

```
                data1[i]=data[i]-48;    //data1 array is used to store only integer data
        }

        for(i=0;i<r;i++)            //initialising parity bits' positions with some value (say 999) in data2 array
                {
                        z=pow(2,i);
                        data2[z]=999;  //data2 array is used to store data+parity bits
                }

        for(i=dl+r;i>=1;i--)     //this loop is used to place the data bits and parity bits at fixed positions
        {
                if(data2[i]!=999)
                        {
                        data2[i]=data1[j];        //if it's not a parity bit, store the data bit in reverse order
                        j++;
                        }
        }

        for(i=0;i<r;i++)            //outer loop is used to find the values for each parity bit
        {
                z=pow(2,i);                 //finding position of each parity bit
                c=0;                        //initializing counter c

                for(j=z;j<=dl+r;j=z+k)          //inner loop is used to add data bits related to each parity bit
                {
                        for(k=j;k<z+j;k++)     //this loop is for part by part parity calculation
                        {
                                if(k<=dl+r)
                                {
                                        if(data2[k]!=999)        //if k is not a parity bit
                                        {
                                                c=c+data2[k];  //add the value of that position with counter c
                                        }
                                }
                        }
                }
                data2[z]=c%2; //parity bit value
        }

        printf("\n The codeword is: ");
        j=0;

        for(i=dl+r;i>=1;i--)
                {
                printf("%d",data2[i]);
                }
// *************END CODE FOR hamming SENDER ************//

return(0);
}
```

2. **Write a C code to decode a user provided codeword using Hamming Code procedure.**
[Input will be the generated codeword. If there is no error Output will be “Actual data received”. If there is any error in the codeword, output will be the position of the error and the original codeword]

Test case 1:

Input:
Codeword: 10101010000

Output:

```
Actual data received
```

Test case 2:

Input:
Codeword: 10101000000

Output:

```
Wrong data received

Error at position 5

Corrected codeword is: 10101010000
```

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

int main()
 {
//************** Hamming client code start *************
        char data[100];
        int data1[100],data2[100];
        int dl,r,i=0,j=0,k=0,z,c,l;
        printf("\n Enter the codeword: "); //taking input in string
        scanf("%s",data);
        dl=strlen(data); //length of the codeword

        while(1)        //finding number of parity bits
        {
                if(pow(2,i)>=dl+1)
                        break;
                i++;
        }
        r=i;    //storing number of parity bits into r variable
        j=dl-1; //last position of the character array

        for(i=1;i<=dl;i++)
        {
                data1[i]=data[j]-48;    //converting character array into integer array in reverse order
        j--;
    }

    l=1;            //l variable is used to store parity values in data2[]
    int count=0;    //count variable is used to check whether all the parity values are 0 or not

    for(i=0;i<r;i++)   //outer loop is used to find the values for each parity bit
    {
        z=pow(2,i);             //finding position of each parity bit
        c=0;                    //initializing counter c

        for(j=z;j<=dl;j=z+k)    //inner loop is used to add bits related to each parity position
        {
            for(k=j;k<z+j;k++)      //this loop is for part by part parity calculation
            {
                if(k<=dl)
                    c=c+data1[k];   //add values with variable c
            }
        }
        data2[l]=c%2;   //store the parity values in the lth location (starting from 1) of data2[]
        count=count+data2[l];   //parity value will be added to counter
        l++;    //l will be incremented to store next parity value in data2[]
    }

    if(count==0)    //if counter=0, no error
        {
            printf("\n Actual data received \n");
        }
    else    //if counter!=0, error exist
        {
            printf("\n Wrong data received \n");
            j=0;

            for(i=r;i>=1;i--)   //this loop will convert wrong binary bit position into decimal value
            {
                if(data2[i]==1)
                j=j+pow(2,(i-1));
            }
            printf("\n Error at position %d",j);

            if(data1[j]==0)   //correct the error at that position
                data1[j]=1;
            else
                data1[j]=0;
            printf("\n Corrected codeword is: ");
            for(i=dl;i>=1;i--)
                printf("%d ",data1[i]);
            printf("\n");
        }
```

```
return(0);
}
```