

**Bansilal Ramnath Agarwal Charitable Trust’s**

Vishwakarma Institute of Technology, Pune-37

*(Anautonomous Institute of Savitribai Phule Pune University)*

**Department of Computer Engineering**

|  |  |
| --- | --- |
| **Division** | **A** |
| **Batch** | **B2** |
| **GR-no** | **12110353** |
| **Roll no** | **10** |
| **Name** | **Aneesh Hemant Dighe** |

***CRC Sender Side***

package computernetworks;

import java.util.Scanner;

public class sendCRC {

public static void appendCRC(char[] code, final char[] divisor) {

int codeLength = code.length;

int divisorLength = divisor.length;

char[] temp = new char[codeLength + divisorLength - 1];

System.*arraycopy*(code, 0, temp, 0, codeLength);

for (int i = 0; i < divisorLength - 1; i++) {

temp[codeLength + i] = '0'; // Append zeros for padding

}

for (int i = 0; i < codeLength; i++) {

if (temp[i] == '1') {

for (int j = 0; j < divisorLength; j++) {

temp[i + j] = (temp[i + j] == divisor[j]) ? '0' : '1';

}

}

}

/\*for(int i=0;i<temp.length;i++)

{

System.out.print(temp[i]);

}

System.out.println();

\*/

//System.arraycopy(temp, codeLength, code, 0, temp.length -codeLength); // Append the CRC bits

for (int i = 0; i < temp.length-codeLength-1; i++) {

code[i] = temp[codeLength + i];

}

for(int i=0;i<code.length;i++)

{

System.***out***.print(code[i]);

}

System.***out***.println();

}

public static void main(String[] args) {

String code1;

String divisor1;

Scanner sc=new Scanner(System.***in***);

System.***out***.println("Enter code:");

code1=sc.nextLine();

System.***out***.println("Enter divisor:");

divisor1=sc.nextLine();

char[] code = code1.toCharArray();

final char[] divisor = divisor1.toCharArray();

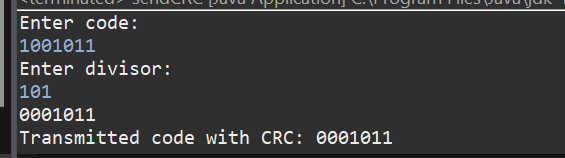
*appendCRC*(code, divisor);

System.***out***.println("Transmitted code with CRC: " + new String(code));

}

}

***OUTPUT:***



***Receiver Side-CRC***

package computernetworks;

import java.util.\*;

public class CRC {

public static boolean checkCRC(String code, String divisor) {

int codeLength = code.length();

int divisorLength = divisor.length();

char[] temp = code.toCharArray();

for (int i = 0; i <= codeLength - divisorLength; i++) {

if (temp[i] == '1') {

for (int j = 0; j < divisorLength; j++) {

temp[i + j] = (temp[i + j] == divisor.charAt(j)) ? '0' : '1';

}

}

}

for(int i=codeLength-divisorLength;i<temp.length;i++)

{

System.***out***.print(temp[i]);

}

System.***out***.println("");

for (int i = codeLength - divisorLength + 1; i < codeLength; i++) {

if (temp[i] == '1') {

return false; // Remainder is non-zero, error detected

}

}

return true; // Remainder is zero, data is correct

}

public static void main(String[] args) {

String code;

String divisor;

Scanner sc=new Scanner(System.***in***);

System.***out***.println("Enter code:");

code=sc.nextLine();

System.***out***.println("Enter divisor:");

divisor=sc.nextLine();

if (*checkCRC*(code, divisor)) {

System.***out***.println("CRC check passed! Data is valid.");

}

else

{

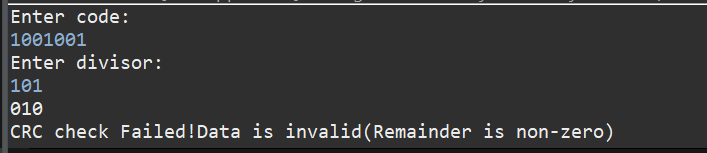
System.***out***.println("CRC check Failed!Data is invalid(Remainder is non-zero)");

}

}

}

***OUTPUT:***



***Hamming Code***

package computernetworks;

import java.util.\*;

class Hamming\_Code

{

public static void main(String[] args)

{

System.***out***.println("HAMMING CODE\n");

System.***out***.println("For SENDER:\n");

System.***out***.print("Enter number of bits of data word: ");

Scanner s1 = new Scanner(System.***in***);

int d = s1.nextInt();

int dataWord[] = new int[d];

int r = *calcNoOfRedBits*(d);

System.***out***.println("Number of Redundant bits are: "+r);

System.***out***.println("Enter Data word:");

for(int i=0;i<d;i++)

dataWord[i] = s1.nextInt();

int senderCodeWord[] = new int[d+r];

*calcSenderCodeWord*(senderCodeWord,dataWord,r,d);

System.***out***.println("Sender Code Word:");

*codeWord*(senderCodeWord,d,r);

for(int i=0;i<(d+r);i++)

System.***out***.print(senderCodeWord[i]+" ");

System.***out***.println();

int recCodeWord[] = new int[d+r];

System.***out***.println("\nFor RECEIVER:\n");

System.***out***.println("Enter Receieved Code Word: ");

for(int i=0;i<(d+r);i++)

recCodeWord[i] = s1.nextInt();

int redBits[] = new int[r];

boolean err = *detectError*(recCodeWord,redBits,d,r);

if(err)

{

System.***out***.println("\nNo error is transmission.");

System.***out***.println("\nExtracted Data word is:");

for(int i=0;i<d;i++)

System.***out***.print(dataWord[i]+" ");

System.***out***.println();

}

else

{

System.***out***.println("\nError found in data transmission.\n----------------------------------");

System.***out***.println();

*correctError*(recCodeWord,redBits,dataWord,d,r);

System.***out***.println("Code Word after Error correction is:");

for(int i=0;i<(d+r);i++)

System.***out***.print(recCodeWord[i]+" ");

System.***out***.println();

System.***out***.println("\nExtracted Data word is:");

for(int i=0;i<d;i++)

System.***out***.print(dataWord[i]+" ");

System.***out***.println();

}

}

public static int calcNoOfRedBits(int d)

{

int i,r=1;

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

{

if(Math.*pow*(2,i)>=(d+i+1))

{

r=i;

break;

}

else

continue;

}

return r;

}

public static void calcSenderCodeWord(int senderCodeWord[],int dataWord[], int r, int d)

{

int i, j, k=d-1, ind=1;

boolean flag = false;

for(i=(d+r-1);i>=0;i--)

{

flag=false;

for(j=0;j<ind;j++)

{

if((int)Math.*pow*(2,j)==ind)

{

flag=true;

break;

}

}

if(flag)

{

senderCodeWord[i]=0;

}

else

{

senderCodeWord[i]=dataWord[k];

k--;

}

ind++;

}

}

public static void codeWord(int senderCodeWord[],int d, int r)

{

System.***out***.println("\nRedundant Bits:");

int i,ind=1,j,k,xor=0,count;

boolean flag=false;

for(i=(d+r-1);i>=0;i--)

{

flag=false;

xor=0;

for(j=0;j<ind;j++)

{

if((int)Math.*pow*(2,j)==ind)

{

flag=true;

break;

}

}

if(flag)

{

k=d+r-ind;

count=ind;

while(k>=0)

{

xor=xor^(senderCodeWord[k]);

k--;

count--;

if(count==0)

{

k=k-ind;

count=ind;

}

}

System.***out***.println("R"+(i-(d+r))+" = "+xor);

senderCodeWord[i]=xor;

}

ind++;

}

System.***out***.println();

}

public static boolean detectError(int recCodeWord[], int redBits[],int d, int r)

{

*codeWord*(recCodeWord,d,r);

int i,j,ind=1,rind=r-1;

boolean flag=false;

for(i=(d+r-1);i>=0;i--)

{

flag=false;

for(j=0;j<ind;j++)

{

if((int)Math.*pow*(2,j)==ind)

{

flag=true;

break;

}

}

if(flag)

{

redBits[rind] = recCodeWord[i];

rind--;

}

ind++;

}

for(i=0;i<r;i++)

{

if(redBits[i]==1)

{

return false;

}

}

return true;

}

public static void correctError(int recCodeWord[],int redBits[], int dataWord[], int d, int r)

{

String binary = "";

for(int i=0;i<r;i++)

{

binary=binary+redBits[i];

}

int decimal = Integer.*parseInt*(binary,2);

System.***out***.println("Error detected at bit position "+decimal);

int index = d+r-decimal;

if(recCodeWord[index]==0)

{

recCodeWord[index]=1;

}

else

{

recCodeWord[index]=0;

}

int i,j,ind=1,dind=d-1;

boolean flag=false;

for(i=(d+r-1);i>=0;i--)

{

flag=false;

for(j=0;j<ind;j++)

{

if((int)Math.*pow*(2,j)==ind)

{

flag=true;

break;

}

}

if(flag)

{

continue;

}

else

{

dataWord[dind]=recCodeWord[i];

dind--;

}

ind++;

}

}

}

***OUTPUT:***

HAMMING CODE

For SENDER:

Enter number of bits of data word: 4

Number of Redundant bits are: 3

Enter Data word:

1

0

0

1

Sender Code Word:

Redundant Bits:

R-1 = 0

R-2 = 0

R-4 = 1

1 0 0 1 1 0 0

For RECEIVER:

Enter Receieved Code Word:

1

0

0

1

0

0

0

Redundant Bits:

R-1 = 1

R-2 = 1

R-4 = 0

Error found in data transmission.

----------------------------------

Error detected at bit position 3

Code Word after Error correction is:

1 0 0 0 1 1 1

Extracted Data word is:

1 0 0 1