Data Commun. & Computer Networks Lab

DATE: 26-03-21

Name: RAJUL DUBEY

SAPID: 500069424

ROLLno: R171218081

**Task->** Create a Cyclic Redundancy Check (CRC) code  
A **cyclic redundancy check** (**CRC**) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short **check** value attached, based on the remainder of a polynomial division of their contents.

**Transmitted Code: 100100**

**Divisor: 1101**

**C Language Code:**

#include<stdio.h>

#define LENGTH 6

#define LENGTHDIV 4

int **Xor1**(int x,int y) //Xor funtion

{

if(x==y)

{

return 0;

}

else

{

return 1;

}

}

void **Divisor**(int ar2[],int divar[],int ar3[]) //Divisor Function

{

int lenar2= (LENGTH+(LENGTHDIV-1));

int lendiv=LENGTHDIV;

int i,j,k,elementcount=lendiv;

printf("\nValue of leanar2: %d",lenar2); //debug

printf("\nValue of lendiv: %d\n",lendiv); //debug

//int ar3[lendiv]; //intialized another array ar3 to hold the remider of divisons

for(j=0;j<lendiv;j++) //copying first 4 element of ar2 in ar3

{

ar3[j]=ar2[j];

}

for(i=0;i<lenar2-lendiv;i++) //this loop will run till the length of ar2 - length of divison array

{

for(j=0;j<lendiv;j++) //in this for loop we checking for XOR condition through XOR funtion

{

ar3[j]=Xor1(ar3[j],divar[j]);

}

printf("\nReminder array(ar3) before conversion"); //Printing before removing the zeros from the starting of reminder array

//debug

for(k=0;k<lendiv;k++)

{

printf(" %d",ar3[k]);

}

//end

while(ar3[0]==0 && elementcount<=lenar2-1) // here we shifting the elements to remove starting 0's

{ //if there's a zero in starting

printf(" <Bring %d>",ar2[elementcount]); //Priting the element which we taking from ar2 after division

for(k=0;k<lendiv-1;k++) //this for loop will run till (length of divison array)-1

{

ar3[k]=ar3[k+1]; //assing +1 index for removing zero at first index

}

ar3[lendiv-1]=ar2[elementcount]; //assigning last index of reminder array to another bring from ar2[]

elementcount+=1; //increasing the count of element which are already brought in the reminder array

}

printf("\nReminder array(ar3) after conversion "); //After shiting the 0's and bringing new elements from ar2 in reminder array

//debug

for(k=0;k<lendiv;k++)

{

printf(" %d",ar3[k]);

}

//printf(" VAlue extracted from ar2",)

//end

}

}

int **main()**

{ int i,j;

int ar1[LENGTH]={1,0,0,1,0,0}; //array which holding transmitted code;

int ar2[LENGTH+(LENGTHDIV-1)]; //Array which will hold code after includeing 0's

int divAr[LENGTHDIV]={1,1,0,1}; //Array which will hold Divisor

int ar3[LENGTHDIV];

for(i=0;i<LENGTH+(LENGTHDIV-1);i++) //initializing ar2 with 0's

{

ar2[i]=0;

}

for(i=0;i<LENGTH;i++) //now putting elements of ar1 in ar2 till the length of ar1 after it all will be zeros

{

ar2[i]=ar1[i];

}

printf("Merged array ar2: ");

for(i=0;i<LENGTH+(LENGTHDIV-1);i++) //Printing ar2 after merge

{

printf(" %d",ar2[i]);

}

Divisor(ar2,divAr,ar3); //funtion for Divison operations

////

printf("\nPrinting the reminder: ");

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

{

printf(" %d",\*(ar3+i));

}

printf("\n\n\*\*Merging reminder with transmitted code in ar2\*\*\n\n");

for(j=1,i=LENGTH;i<LENGTH+LENGTHDIV-1;i++,j++) //here j initialized as 1 to ignore first 0 of ar3;

{

ar2[i]=ar3[j];

}

printf("Converted Merged array ar2: (CRC): ");

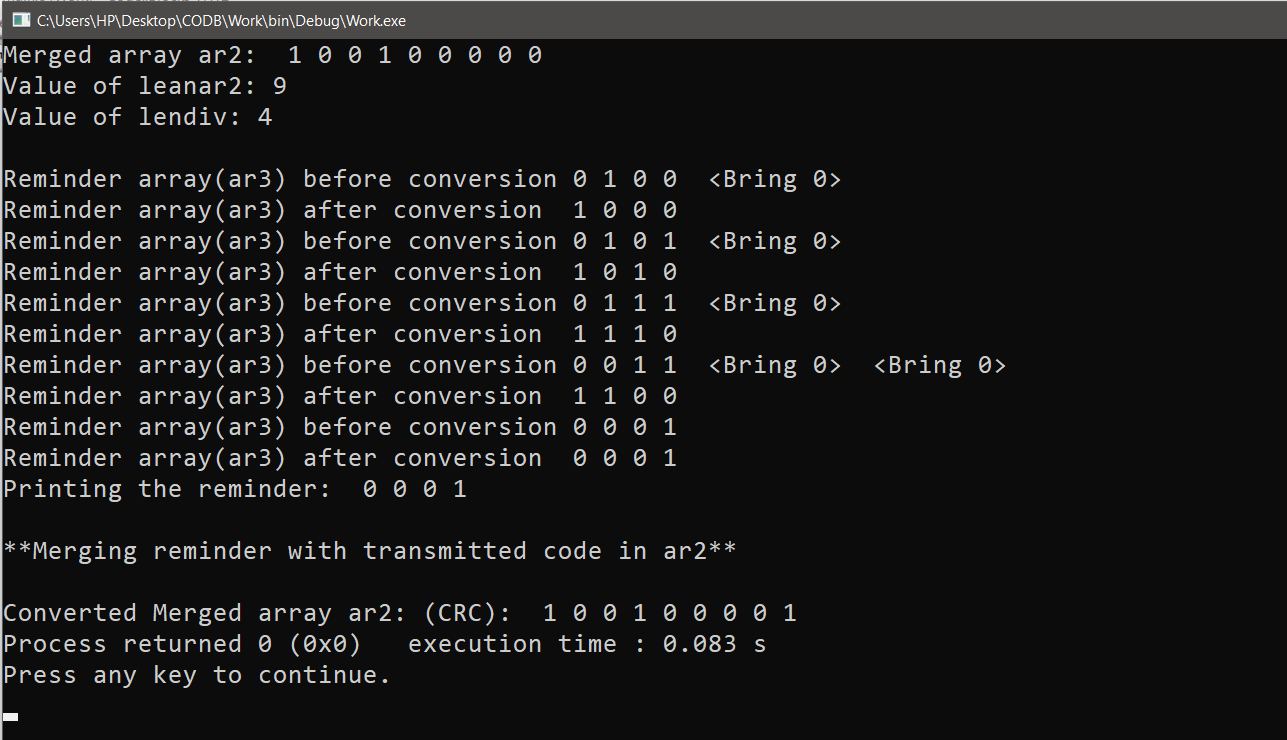
for(i=0;i<LENGTH+(LENGTHDIV-1);i++) //Printing ar2 after merge

{

printf(" %d",ar2[i]);

}

}

**OUTPUT:**