**G. H. RAISONI COLLEGE OF ENGG., NAGPUR**

**(An Autonomous Institute)**

**Department of Computer Science & Engg.**



**Date: 31-08-2021**

**Practical Subject: Data Communication and Networking.**

**Session: 2021-22**

**Student Details:**

| **Roll Number** | 01 |
| --- | --- |
| **Name** | Anand Suralkar |
| **Semester** | 9th |
| **Section** | A |
| **Batch** | CSE |

**Practical Details: Practical Number-2;**

| Practical Aim | Implement Cyclic  Redundancy Check code for a given  data |
| --- | --- |
| Theory & Syntax | Cyclic Redundancy Check (CRC)  is an error detection technique  Used to detect errors in data/digital data  that has been transmitted on a communications link.  used in digital networks and storage devices to  detect accidental changes to raw data.  As a type of checksum, the CRC produces a fixed-length  data set based on the build of a file or larger data set.  In terms of its use, CRC is a hash function that detects  accidental changes to raw computer data commonly  used in digital telecommunications networks and  storage devices such as hard disk drives.  This technique was invented by W. Wesley  Peterson in 1961  Cyclic redundancy checks are quite simple to  implement in hardware, and can be easily analyzed  mathematically.  CRC is based on binary division and is also called  “polynomial code checksum.”  Blocks of data entering these systems get a  short check value attached, based on the  remainder of a polynomial division of their  contents.  On retrieval, the calculation is repeated and, in  the event the check values do not match,  corrective action can be taken against data  corruption.  In the cyclic redundancy check, a fixed number of  check bits, often called a checksum, are appended  to the message that needs to be transmitted.  The data receivers receive the data, and inspect  the check bits for any errors.  Mathematically, data receivers evaluate the check  value that is attached by finding the remainder of  the polynomial division of the contents  transmitted.  If it seems that an error has occurred, a negative  acknowledgement is transmitted asking for data  retransmission.  applied to storage devices like hard disks. In this case, check  bits are allocated to each block in the hard disk.  When the computer reads a corrupt or incomplete  file,  a cyclic redundancy error gets triggered.  The CRC can come from another storage device or  from CD/DVDs.  The common reasons for errors include  system crashes, incomplete or corrupt files, or files with lots  of bugs.  CRC polynomial designs depend on the length of  the block that is supposed to be protected.  Error protection features can also determine the  CRC design.  “redundant” because it adds to the size of the data set  without adding new information, and  “cyclical” because it works on a system of cyclical  implementations.  CRC uses Generator Polynomial which is available  on both sender and receiver side.  Sender Side (Generation of Encoded Data from Data  and Generator Polynomial (or Key)):  The binary data is first augmented by adding k-1 zeros  in the end of the data  Use modulo-2 binary division to divide binary data by  the key and store remainder of division.  Append the remainder at the end of the data to form  the encoded data and send the same.  Modulo 2 Division:  The process of modulo-2 binary division is the same as the  familiar division process we use for decimal numbers.  Just that instead of subtraction, we use XOR here.  In each step, a copy of the divisor (or data) is XORed with  the k bits of the dividend (or key).  The result of the XOR operation (remainder) is (n-1) bits,  which is used for the next step after 1 extra bit is pulled  down to make it n bits long.  When there are no bits left to pull down, we have a result.  The (n-1)-bit remainder which is appended at the sender  side.  Receiver Side (Check if there are errors  introduced in transmission)  Perform modulo-2 division again and if the  remainder is 0, then there are no errors.  In this, we focus only on finding the remainder i.e. check word and the code word. |
| Program | #include <iostream>  #include <math.h>  #include <cstring>  using namespace std;  char exor(char a,char b)  {  if(a==b)  return '0';  else  return '1';  }  void crc(char data[], char key[])  {  int datalen = strlen(data);  int keylen = strlen(key);  for(int i=0;i<keylen-1;i++)  data[datalen+i]='0';  data[datalen+keylen-1]='\0';  int codelen = datalen+keylen-1;  char temp[20],rem[20];  for(int i=0;i<keylen;i++)  rem[i]=data[i];  for(int j=keylen;j<=codelen;j++)  {  for(int i=0;i<keylen;i++)  temp[i]=rem[i];  if(rem[0]=='0')  {  for(int i=0;i<keylen-1;i++)  rem[i]=temp[i+1];  }  else  {  for(int i=0;i<keylen-1;i++)  rem[i]=exor(temp[i+1],key[i+1]);    }  if(j!=codelen)  rem[keylen-1]=data[j];  else  rem[keylen-1]='\0';  }    for(int i=0;i<keylen-1;i++)  data[datalen+i]=rem[i];  data[codelen]='\0';  cout<<"CRC="<<rem<<"\nDataword="<<data;  }  int main()  {  char key[20],data[20];  cout<<"Enter the data:";  cin>>data;  cout<<"Enter the key:";  cin>>key;  crc(data,key);  return 0;  } |
| Output |  |
| Conclusion | Implemented and learned concept of Cyclic  Redundancy Check code for a given  data |