**RAMAIAH**

**Institute of**

**Technology**

**Certificate**

*This report is submitted for the evaluation of Practical Assignment component for the subject "DATA COMMUNICATION" with the subject code CS1544 during the term January to May 2017.*

Submitted by

**1MS15CS074 M V S VISWANADH**

**1MS15CS090 PRABHULING**

**1MS15CS091 PRASHANT KRISHNAN V**

Signature of Faculty

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

www.msrit.edu

**ACKNOWLEDGEMENT**

I express my sincere gratitude to Prof. Mamtha Jadav V, Dept. of Computer Science and Engineering, MSRIT, for her stimulating guidance, continuous encouragement and supervision throughout the course of present work.

**Signature(s) of Students**

M V S VISWANADH PRABHULING PRASHANT KRISHNAN V

(1MS15CS074) (1MS15CS090) (1MS15CS091)

# CERTIFICATE

I hereby certify that the work which is being presented in the B.E. Practical Assignment component for the subject "DATA COMMUNICATION" for the project report entitled “GENERATOR POLYNOMIAL ”**,** in partial fulfillment of the requirements for the award of the **Bachelor of Engineering in Computer Science & Engineering** and submitted to the Department of Computer Science & Engineering of MS Ramaiah Institute of Technology an authentic record of my own work carried out during Practical Assignment under the supervision of **Mamtha Jadav V, Assistant Professor, CSE Department**.

**Signature of Student (S)**

M V S VISWANADH PRABHULING PRASHANT KRISHNAN V

(1MS15CS074) (1MS15CS090) (1MS15CS091)

This is to certify that the above statement made by the student(s) is correct to the best of my knowledge.

**Signature of Supervisor**

**Mamatha Jadhav V.**

**TABLE OF CONTENTS**

Page No.

Acknowledgement 1

Certificate 2

Introduction 4

Code 5-12

Snapshots 13-14

Conclusion 15

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**INTRODUCTION**

The project aims at identifying whether the given polynomial is a good polynomial or not, mainly to identify the single bit errors, 2 isolated errors, odd number errors and also burst errors.

**Single bit errors:**

If the generator has more than one term and the coefficient of x0 is 1, all single errors can be caught.

**2 isolated errors:**

If a generator cannot divide x^t + 1 (t between 0 and n – 1), then all isolated double errors can be detected.

**Odd numbered errors:**

A generator that contains a factor of x + 1 can detect all odd-numbered errors.

**Burst errors:**

All burst errors with L ≤ r will be detected.

All burst errors with L = r + 1 will be detected with probability 1 – (1/2)r–1. All burst errors with L > r + 1 will be detected with probability 1 – (1/2)r.

**Characteristics of a good polynomial:**

A good polynomial generator needs to have the following characteristics:

1. It should have at least two terms.

2. The coefficient of the term x0 should be 1.

3. It should not divide xt + 1, for t between 2 and n − 1.

4. It should have the factor x + 1.

**Polynomials:**

A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0

and 1. The power of each term shows the position of the bit; the coefficient

shows the value of the bit.

**Degree of a Polynomial:**

The degree of a polynomial is the highest power in the polynomial. For

example, the degree of the polynomial x^6 + x + 1 is 6. Note that the degree of

a polynomial is 1 less that the number of bits in the pattern. The bit pattern in

this case has 7 bits.

**CODE**

**<html>**

**<head>**

**<script type="text/javascript">**

**function burst(n)**

**{**

**document.getElementById("bur").innerHTML = "3) Burst Errors : ";**

**var prob=0;**

**document.getElementById("bur1").innerHTML="a. This generator can detect all burst errors with a length less than or equal to " + (n-1) + " bits.";**

**prob=1 - Math.pow(0.5,n-2);**

**document.getElementById("bur2").innerHTML="b. The probability of detecting burst errors of length " + n + " is " + prob ;**

**prob=1-(Math.pow(0.5,n-1));**

**document.getElementById("bur3").innerHTML="c. The probability of detecting burst errors of length more than " + (n+1) + " is " + prob;**

**}**

**function sinerr(n,b)**

**{**

**var s=b.charAt(n-1);**

**document.getElementById("single1").innerHTML = "1) Single bit errors";**

**if(s==1)**

**{**

**document.getElementById("single").innerHTML = "Any single-bit error can be caught.";**

**}**

**else if(n==2)**

**{**

**document.getElementById("single").innerHTML = "Only single-bit error in positions 1 is caught.";**

**}**

**else**

**{**

**document.getElementById("single").innerHTML = "All single-bit errors in positions 1 to " + (n-1) + " are caught.";**

**}**

**}**

**function iso(n,b)**

**{**

**document.getElementById("isol1").innerHTML="2) Two isolated errors :";**

**a=b.charAt(0);**

**c=b.charAt(1);**

**if(n==2)**

**{**

**if( a==1 && c ==1 )**

**document.getElementById("isol").innerHTML="Any two errors next to each other cannot be detected.";**

**else**

**document.getElementById("isol").innerHTML="This generator cannot detect two errors that are " + (n-1) +" positions apart.";**

**}**

**else**

**{**

**document.getElementById("isol").innerHTML="This polynomial cannot divide any error of type x^t+1 if t is less than " + Math.pow(2,n-1) + " . This means that a codeword with two isolated errors that are next to each other or upto " + Math.pow(2,n-1) + " bits apart can be detected by this generator.";**

**}**

**}**

**function odd(n,b)**

**{**

**document.getElementById("oodd").innerHTML = "4) Odd-numbered errors : ";**

**var i=0,sum=0,q=[],quo=0;**

**var z=b;**

**var m=b.split("");**

**var r=m.reverse();**

**var y=r.join("");**

**var f=Array.from(y);**

**var g=Array.from(b);**

**if(n>2)**

**{**

**for(i=n-1;i>=0;i--)**

**sum=sum+ parseInt(f[i]);**

**document.getElementById("odds").innerHTML="Sum = " + sum;**

**if(sum%2==0)**

**{**

**for(i=0;i<n-1;i++)**

**{**

**if(g[i]==1)**

**{**

**q[i]=1;**

**if(g[i+1]==1)**

**g[i+1]=0;**

**else**

**g[i+1]=1;**

**}**

**else**

**q[i]=0;**

**quo=(quo\*10)+ parseInt(q[i]);**

**}**

**document.getElementById("odds4").innerHTML=b + " can catch all odd-numbered errors since it can be written as a product of 11 and " + quo;**

**}**

**else**

**document.getElementById("odds4").innerHTML = b + " cannot catch odd-numbered errors";**

**}**

**else**

**document.getElementById("odds4").innerHTML =b + " cannot catch odd-numbered errors ";**

**}**

**function check(n,t)**

**{**

**var i;**

**for(i=0;i<n;i++)**

**if(t[i]==2 || t[i]==3 ||t[i]==4 ||t[i]==5 ||t[i]==6 ||t[i]==7 ||t[i]==8 ||t[i]==9)**

**{**

**document.getElementById("aaa").innerHTML="";**

**alert("INVALID INPUT!! The input is not a binary number! ");**

**location.reload(true);**

**}**

**}**

**function master()**

**{**

**var b=document.getElementById("poly").value;**

**var t=Array.from(b);**

**var n=b.length;**

**check(n,t);**

**if(n==1 || n==0)**

**{**

**document.write("Length is lesser than 2 <br/> No single bit error can be caught")**

**}**

**else**

**{**

**sinerr(n,b);**

**iso(n,b);**

**burst(n);**

**odd(n,b)**

**}**

**}**

**</script>**

**<title> GENERATOR POLYNOMIAL </title>**

**<style type="text/css">**

**body{**

**background-color: #848c7b;**

**}**

**</style>**

**<body>**

**<center><strong><h1 style="color:red;font-family:serif; font-size: : 250%;">GENERATOR POLYNOMIAL</h1></strong></center>**

**<div style="align: center">**

**<center><div style="align : center;font-family: Arial;color: #422a01; "><h2>Enter the generator polynomial </h2><input type="number" id="poly" /></div>**

**<input type="Submit" value="Submit" onclick="master()" /></div></center>**

**<div style="font-family: sans-serif;">**

**<h3 id="bbb"></h3>**

**<h3 id="aaa"> </h3>**

**<h2 id="single1"></h2>**

**<h3 id="single"> </h3>**

**<h2 id="isol1"></h2>**

**<h3 id="isol"> </h3>**

**<h2 id="bur"></h2>**

**<h3 id="bur1"> </h3>**

**<h3 id="bur2"> </h3>**

**<h3 id="bur3"> </h3>**

**<h2 id="oodd"></h2>**

**<h3 id="odds"> </h3>**

**<h3 id="odds4"> </h3></div>**

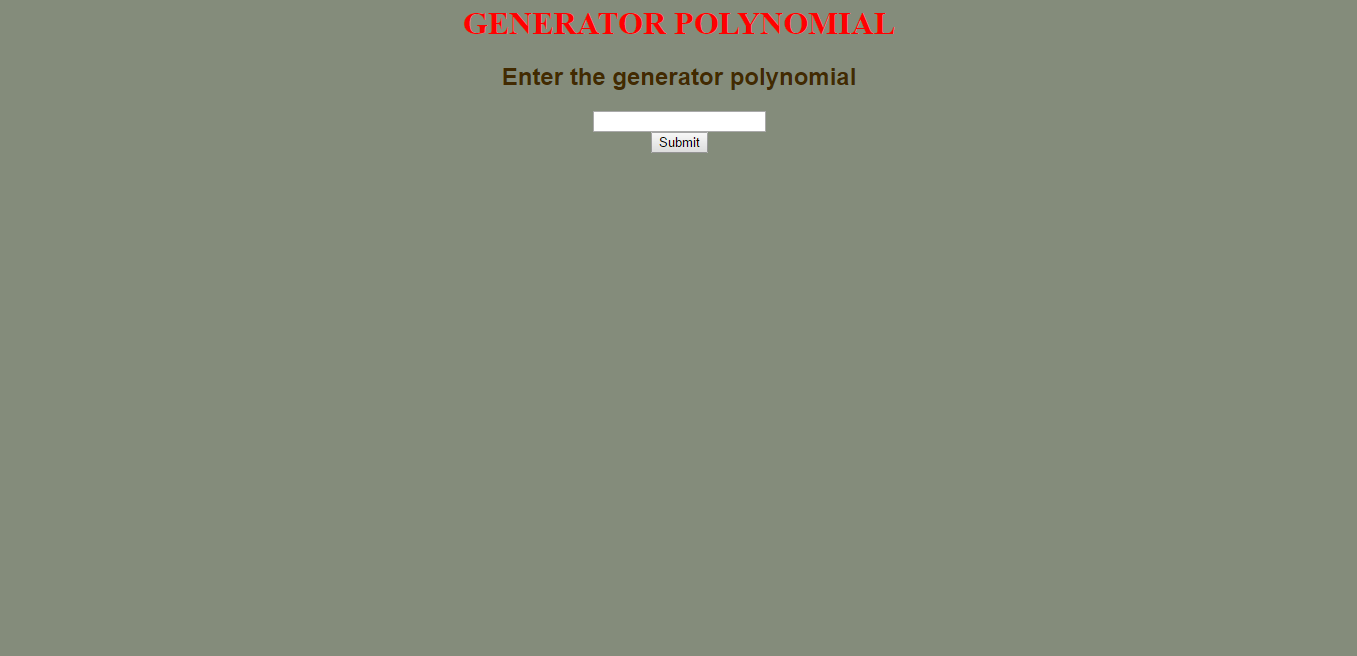
**</div>**

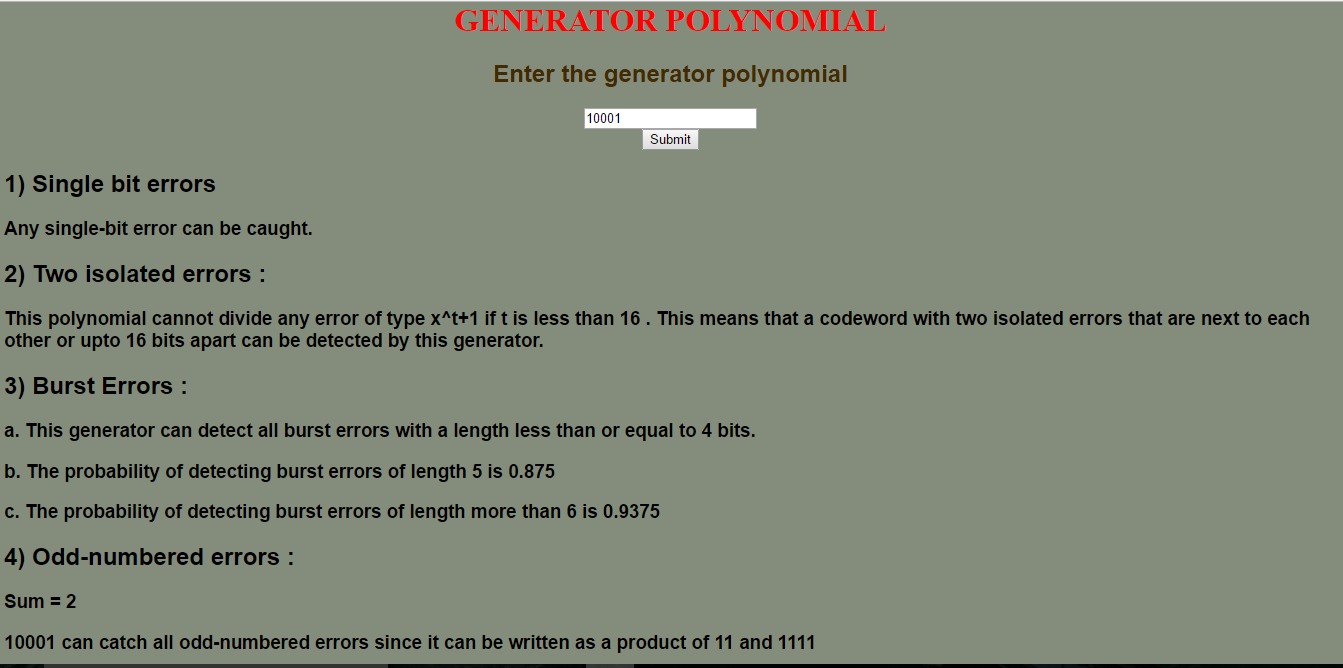
**</head>**

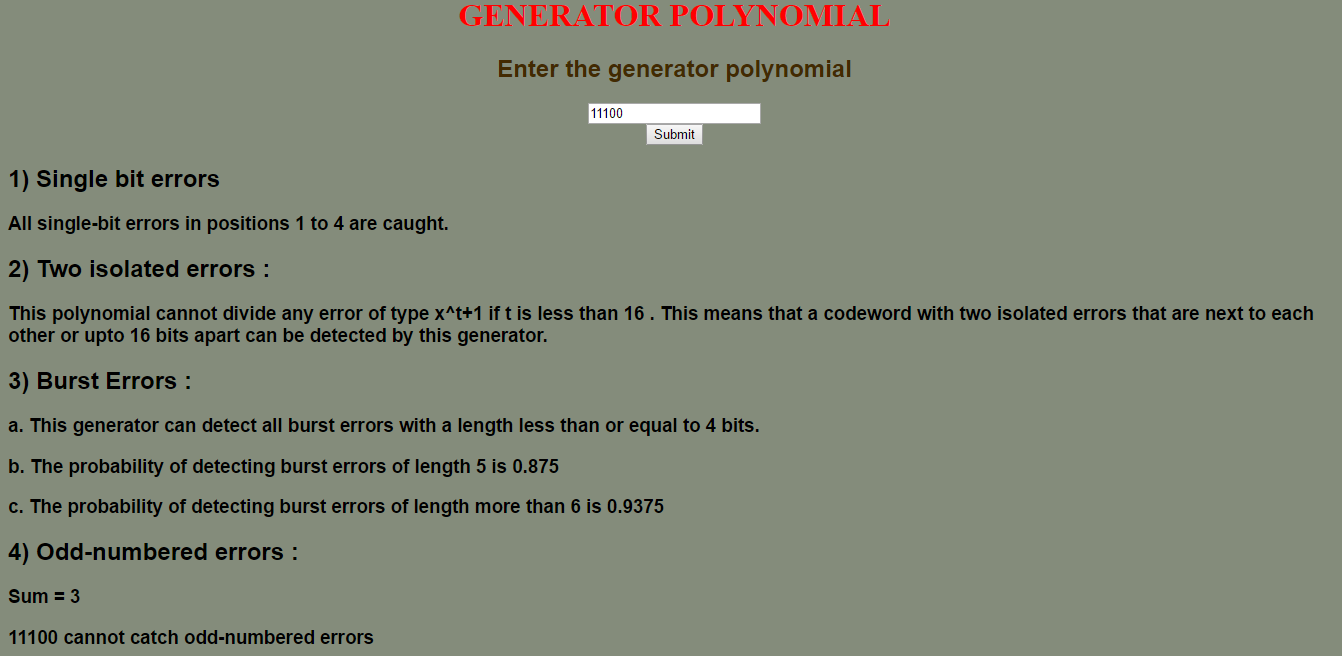
**</body>**

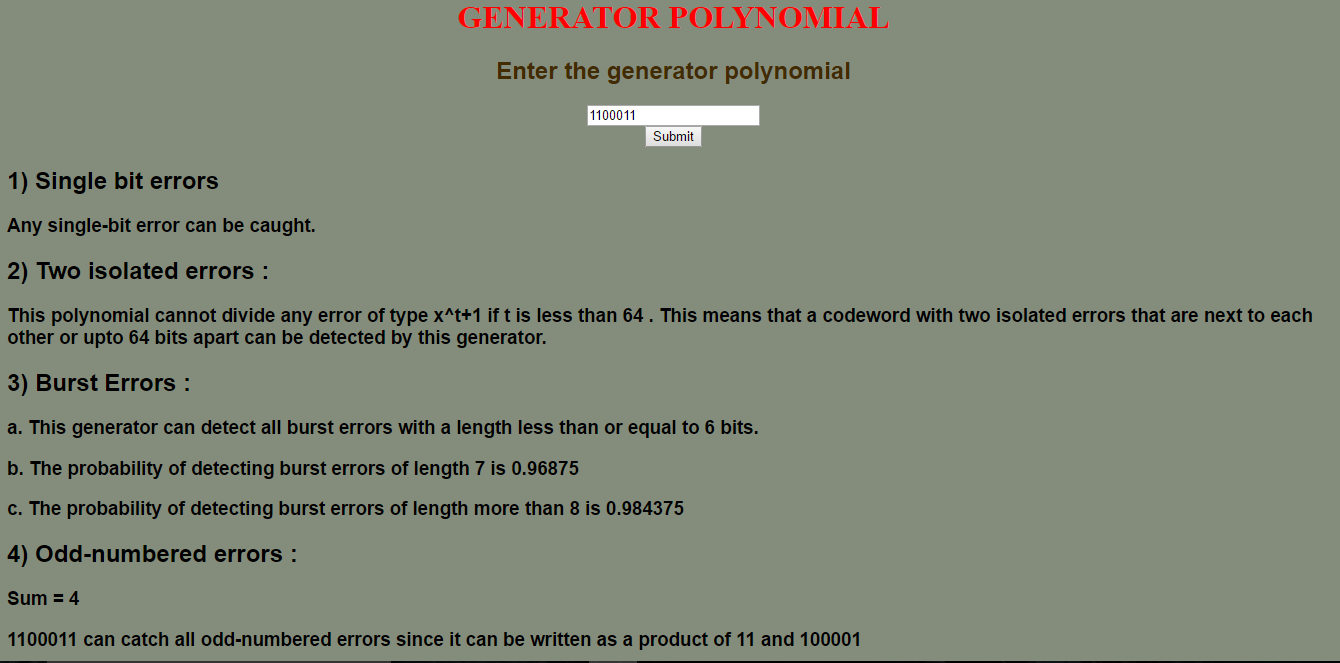
**</html>**

**SNAPSHOTS OF THE OUTPUT**









**CONCLUSION:**

This project has helped us in understanding about the errors in detail and how to identify the different types of errors when the data is represented in the form of a generator polynomial .We have learnt about which kind of polynomial can identify single errors, odd numbered errors, etc. We have understood the magnitude of importance of error checking and the role CRC plays in data transfer in the real world and believe that this project is somewhat of a small prototype of what is actually used by industries. Error checking becomes important because of the amount of data that is sent today and the frequency of that big data. Through the course of the project, our skills in webpage development have also been enhanced.