Experiment 1: Error Detection using Parity

Aim: To apply Parity check rules for error detection

Objective: After carrying out this experiment, students will be able to:

- Apply 1D and 2D parity rules for error detection
- Analyse the difference between 1D and 2D parity and their limitations

Problem statement: You are required to write separate programs to demonstrate the use of 1D and 2D parity. Take the input bit streams (max five) of 7 bit each from the user. Your programs should calculate the parity and display the input and output bit streams.

Analysis: While analysing your program, you are required to address the following points:

- Why can this method not be used to correct errors?
- How are 1D and 2D parity different?
- What are the limitations of this method of error detection?

MARKS DISTRIBUTION

Component	Maximum Marks	Marks Obtained
Preparation of Document	7	
Results	7	
Viva	6	
Total	**20**	

Submitted by: K Srikanth

Register Number: 17ETCS002124

1. Algorithm/Flowchart

i. Start

ii. Declaration of variables

iii. Switch case (case 1) **< 1D Parity >**

iv. Enter the input (input-1)

v. Function call (oneD_Parity)

a. Initialise count <type int>

1. Loop and count the number of 1's in the given input (input-1)

b.Switch case (case 1) <Even Encoded Parity>

1. Divide count with "2" if the reminder is "0" then concatenate "0" with input if there are odd number of '1's '.

2. If the conditions doesn't satisfy then concatenate "1" with input.

c. Switch case (case 2) <Odd Encoded Parity>

1. Divide count with "2" if the reminder is other then "0" concatenate "0" with input of there are even number of '1's' make it odd.

2. If the conditions doesn't satisfy then concatenate "1" with input

vi.Switch case (case 2) **< 2D Parity>**

vii. Enter the number of rows (**input-2**)

viii. Enter the input row wise (*if **input-2** = 3 then 3 inputs)

ix. Function call (twoD_Parity)

a. Initialise count <type int>

1. Loop and count the number of 1's in the given input (input-1)

b.Switch case (case 1) <Row Even Encoded Parity>

1. Divide count with "2" if the reminder is "0" then concatenate "0" with input if there are odd number of '1's '.

2. If the conditions doesn't satisfy then concatenate "1" with input.

c. Switch case (case 2) <Row Odd Encoded Parity>

1. Divide count with "2" if the reminder is other then "0" concatenate "0" with input of there are even number of '1's' make it odd.

2. If the conditions doesn't satisfy then concatenate "1" with input

d. Loop

1. Loop through rows and columns and count the number of 1's.

e. Switch case (case 1) <Column Even Encoded Parity>

1. Divide count with "2" if the reminder is "0" then concatenate "0" with input if there are odd number of '1's '.

2. If the conditions doesn't satisfy then concatenate "1" with input.

f. Switch case (case 2) <Column Odd Encoded Parity>

1. Divide count with "2" if the reminder is other then "0" concatenate "0" with input of there are even number of '1's' make it odd.

2. If the conditions doesn't satisfy then concatenate "1" with input.

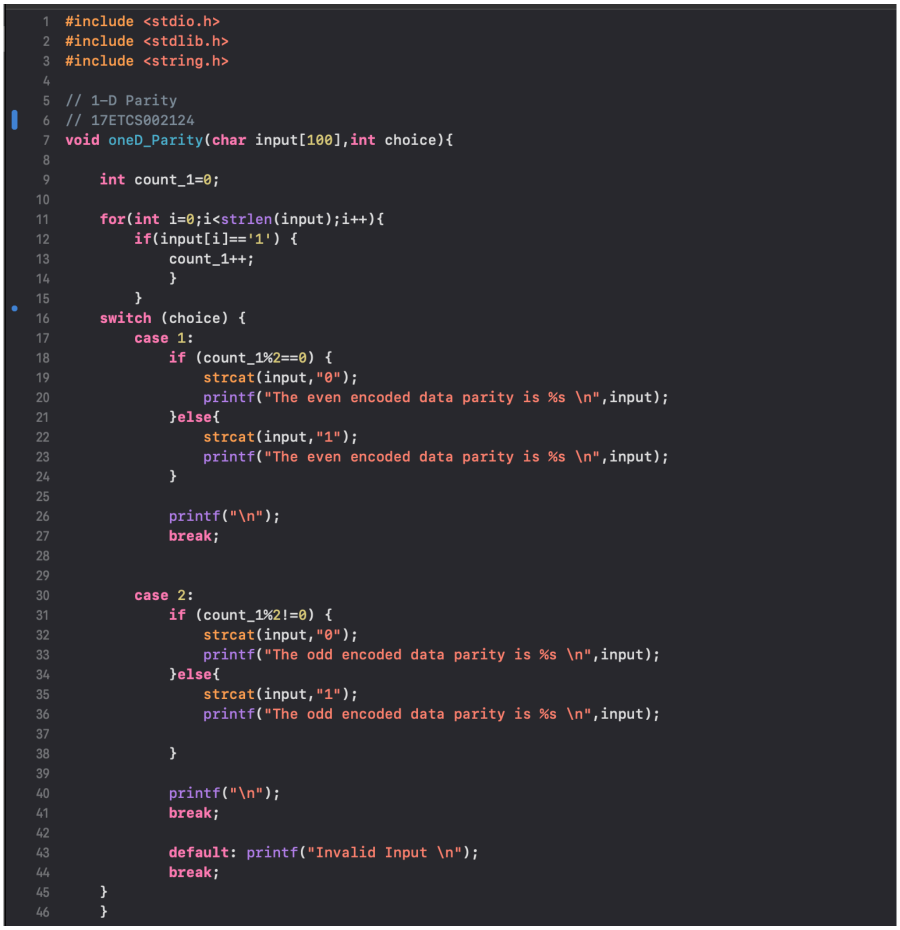
x. Switch case (case 3) **< Exit >**

xi. Stop

2. Program

Code

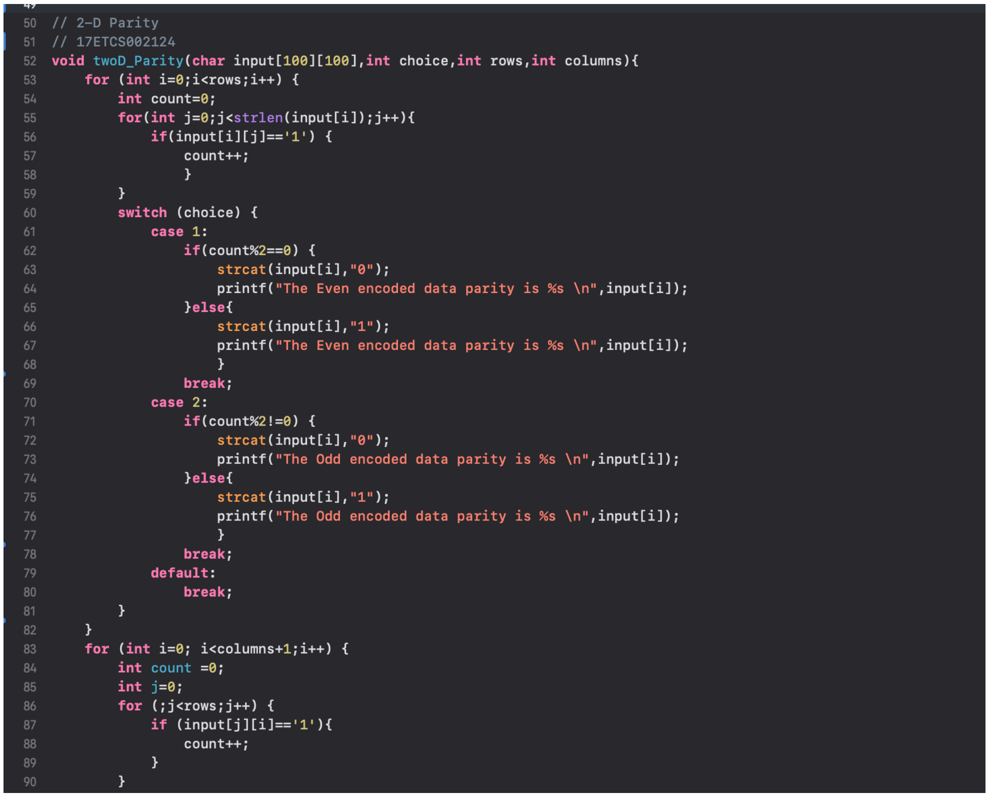
1 - D Parity (Even & Odd)



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// 1-D Parity
// 17ETCS002124
void oneD_Parity(char input[100],int choice){

    int count_1=0;

    for(int i=0;i<strlen(input);i++){
        if(input[i]=='1') {
            count_1++;
            }
        }
    switch (choice) {
        case 1:
            if (count_1%2==0) {
                strcat(input,"0");
                printf("The even encoded data parity is %s \n",input);
            }else{
                strcat(input,"1");
                printf("The even encoded data parity is %s \n",input);
            }

            printf("\n");
            break;


        case 2:
            if (count_1%2!=0) {
                strcat(input,"0");
                printf("The odd encoded data parity is %s \n",input);
            }else{
                strcat(input,"1");
                printf("The odd encoded data parity is %s \n",input);

            }

            printf("\n");
            break;

            default: printf("Invalid Input \n");
            break;
    }
    }
```

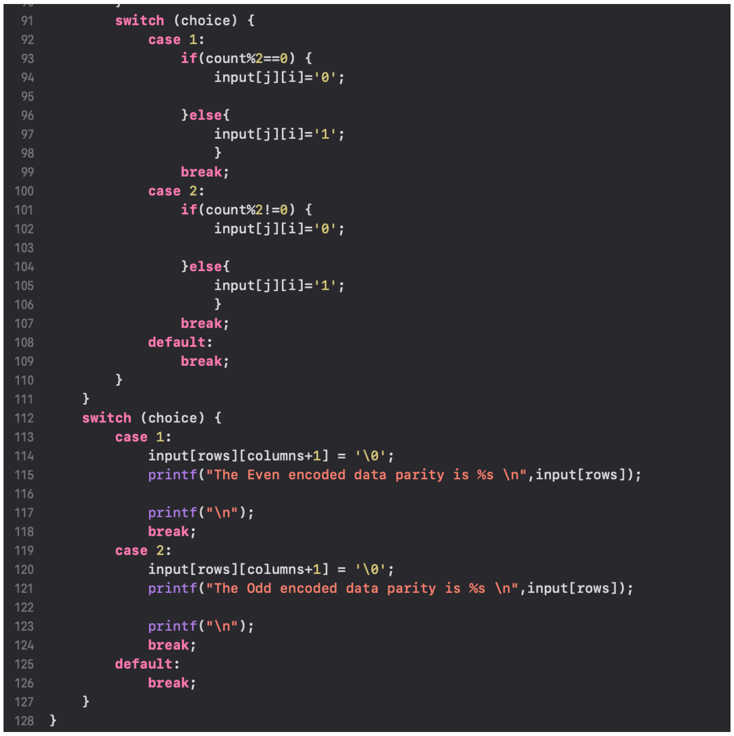
Image 1.1   C Program for 1-D Parity (Even & Odd)

2 - D Parity (Even & Odd)



```c
// 2-D Parity
// 17ETCS002124
void twoD_Parity(char input[100][100],int choice,int rows,int columns){
    for (int i=0;i<rows;i++) {
        int count=0;
        for(int j=0;j<strlen(input[i]);j++){
            if(input[i][j]=='1') {
                count++;
                }
        }
        switch (choice) {
            case 1:
                if(count%2==0) {
                    strcat(input[i],"0");
                    printf("The Even encoded data parity is %s \n",input[i]);
                }else{
                    strcat(input[i],"1");
                    printf("The Even encoded data parity is %s \n",input[i]);
                    }
                break;
            case 2:
                if(count%2!=0) {
                    strcat(input[i],"0");
                    printf("The Odd encoded data parity is %s \n",input[i]);
                }else{
                    strcat(input[i],"1");
                    printf("The Odd encoded data parity is %s \n",input[i]);
                    }
                break;
            default:
                break;
        }
    }
    for (int i=0; i<columns+1;i++) {
        int count =0;
        int j=0;
        for (;j<rows;j++) {
            if (input[j][i]=='1'){
                count++;
            }
        }
```

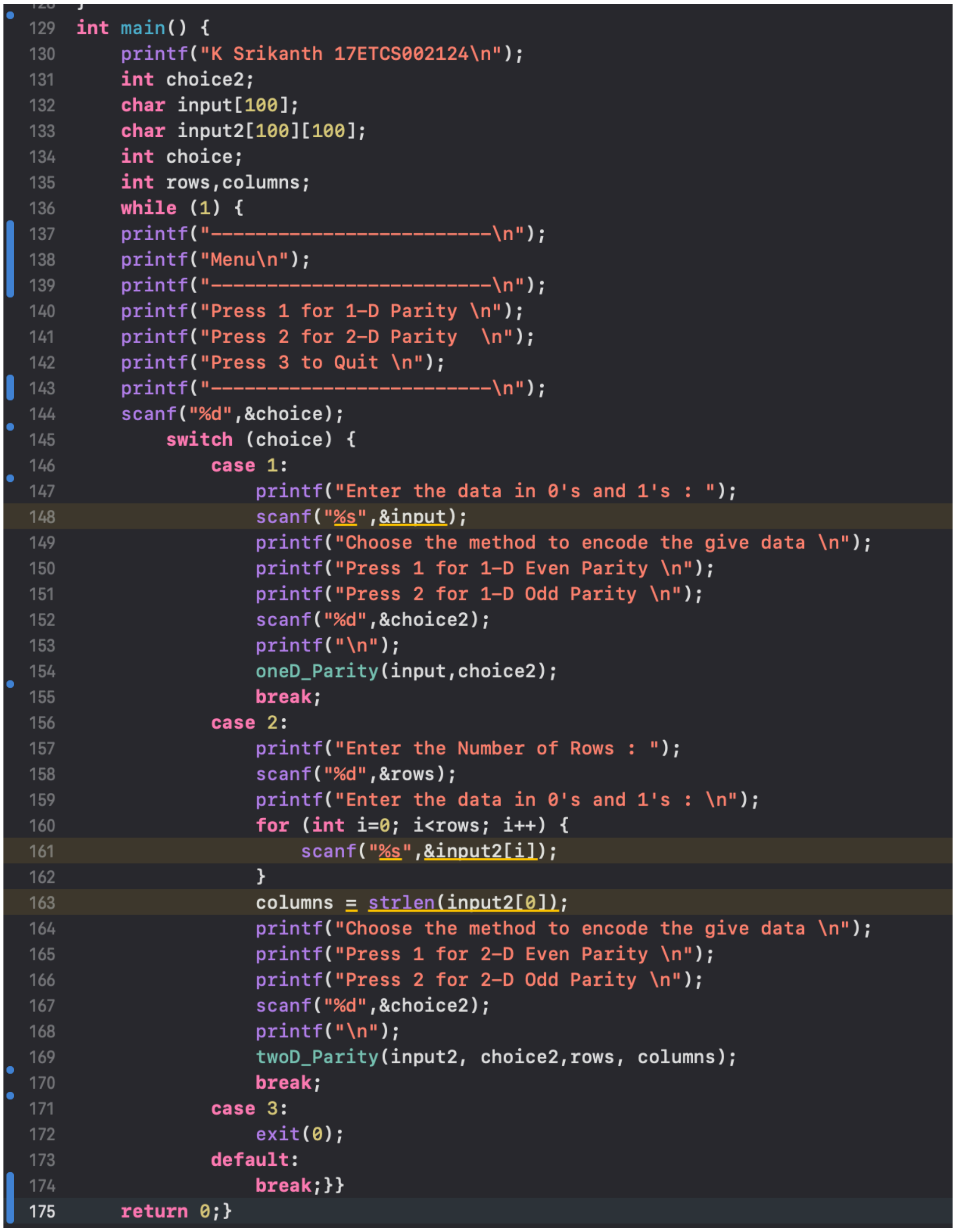
Image 1.2 C Program for 2-D Parity (Even & Odd)



```c
        switch (choice) {
            case 1:
                if(count%2==0) {
                    input[j][i]='0';

                }else{
                    input[j][i]='1';
                    }
                break;
            case 2:
                if(count%2!=0) {
                    input[j][i]='0';

                }else{
                    input[j][i]='1';
                    }
                break;
            default:
                break;
        }
    }
    switch (choice) {
        case 1:
            input[rows][columns+1] = '\0';
            printf("The Even encoded data parity is %s \n",input[rows]);

            printf("\n");
            break;
        case 2:
            input[rows][columns+1] = '\0';
            printf("The Odd encoded data parity is %s \n",input[rows]);

            printf("\n");
            break;
        default:
            break;
    }
}
```

Image 1.3 C Program for 2-D Parity (Even & Odd)

Menu

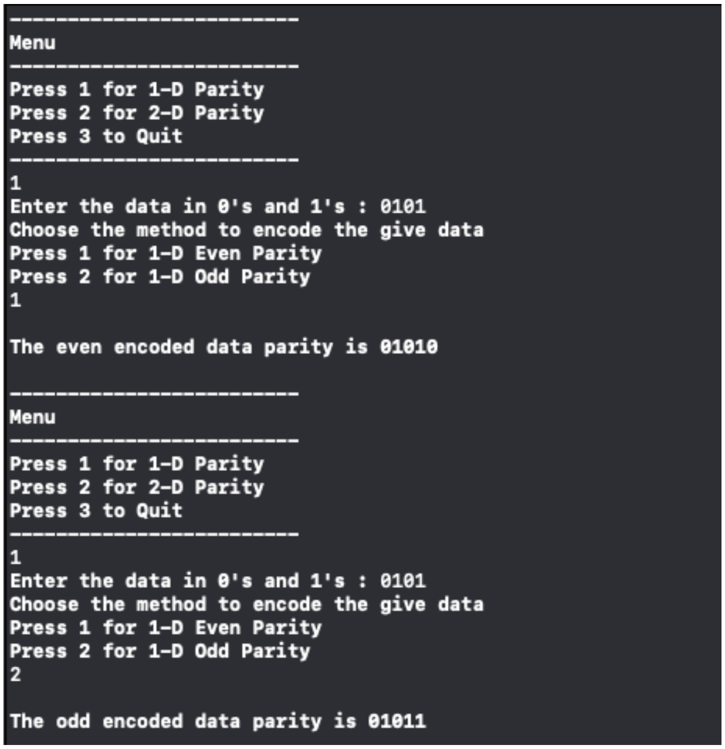


```
int main() {
    printf("K Srikanth 17ETCS002124\n");
    int choice2;
    char input[100];
    char input2[100][100];
    int choice;
    int rows,columns;
    while (1) {
    printf("--------------------------\n");
    printf("Menu\n");
    printf("--------------------------\n");
    printf("Press 1 for 1-D Parity \n");
    printf("Press 2 for 2-D Parity  \n");
    printf("Press 3 to Quit \n");
    printf("--------------------------\n");
    scanf("%d",&choice);
        switch (choice) {
            case 1:
                printf("Enter the data in 0's and 1's : ");
                scanf("%s",&input);
                printf("Choose the method to encode the give data \n");
                printf("Press 1 for 1-D Even Parity \n");
                printf("Press 2 for 1-D Odd Parity \n");
                scanf("%d",&choice2);
                printf("\n");
                oneD_Parity(input,choice2);
                break;
            case 2:
                printf("Enter the Number of Rows : ");
                scanf("%d",&rows);
                printf("Enter the data in 0's and 1's : \n");
                for (int i=0; i<rows; i++) {
                    scanf("%s",&input2[i]);
                }
                columns = strlen(input2[0]);
                printf("Choose the method to encode the give data \n");
                printf("Press 1 for 2-D Even Parity \n");
                printf("Press 2 for 2-D Odd Parity \n");
                scanf("%d",&choice2);
                printf("\n");
                twoD_Parity(input2, choice2,rows, columns);
                break;
            case 3:
                exit(0);
            default:
                break;}}
    return 0;}
```

Image 1.4   C Program Menu for 1-D and 2-D Parity

3. Results

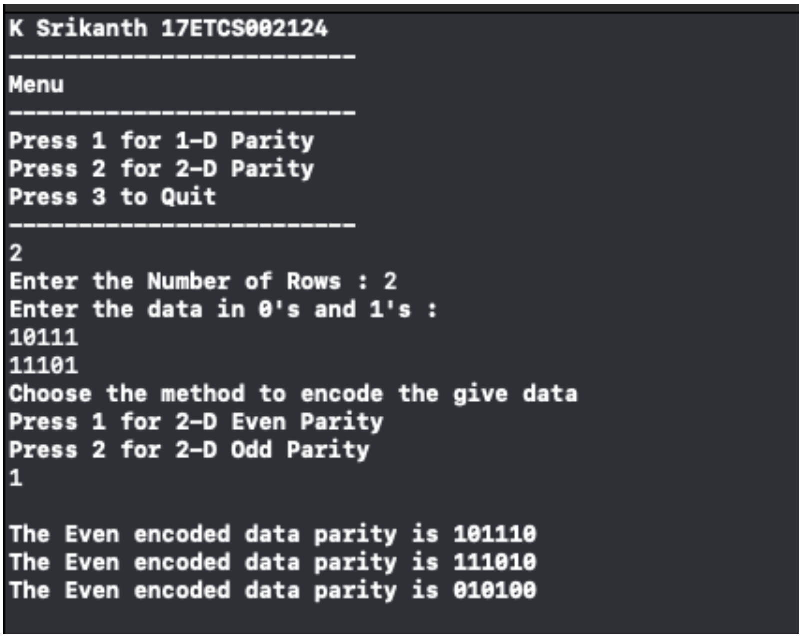
1- D Parity Result



```
-------------------------
Menu
-------------------------
Press 1 for 1-D Parity
Press 2 for 2-D Parity
Press 3 to Quit
-------------------------
1
Enter the data in 0's and 1's : 0101
Choose the method to encode the give data
Press 1 for 1-D Even Parity
Press 2 for 1-D Odd Parity
1

The even encoded data parity is 01010

-------------------------
Menu
-------------------------
Press 1 for 1-D Parity
Press 2 for 2-D Parity
Press 3 to Quit
-------------------------
1
Enter the data in 0's and 1's : 0101
Choose the method to encode the give data
Press 1 for 1-D Even Parity
Press 2 for 1-D Odd Parity
2

The odd encoded data parity is 01011
```

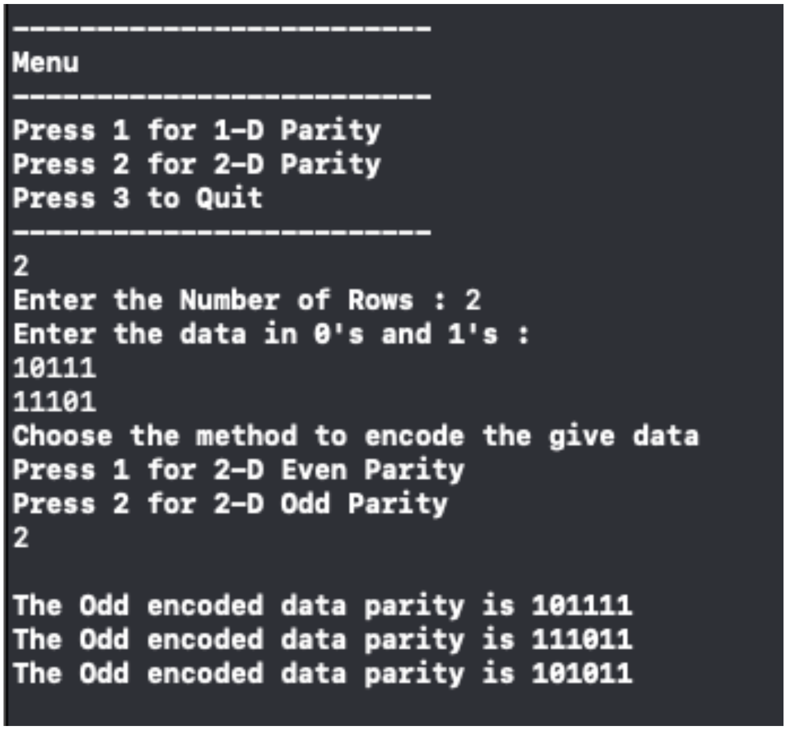
Image 1.5 C Program Output for 1-D Parity (Even & Odd)

2-D Parity Result



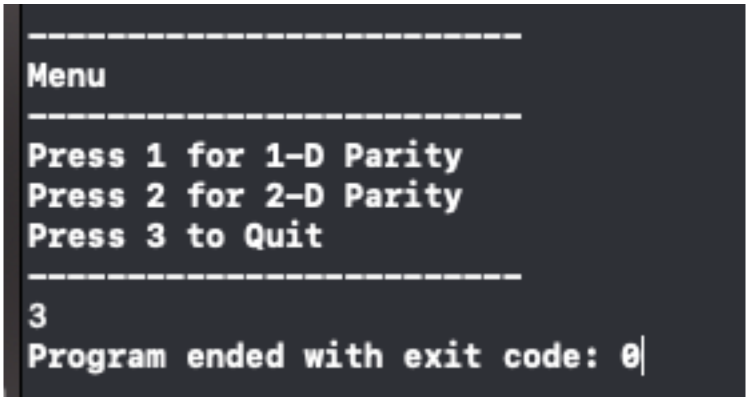
```
K Srikanth 17ETCS002124
-------------------------
Menu
-------------------------
Press 1 for 1-D Parity
Press 2 for 2-D Parity
Press 3 to Quit
-------------------------
2
Enter the Number of Rows : 2
Enter the data in 0's and 1's :
10111
11101
Choose the method to encode the give data
Press 1 for 2-D Even Parity
Press 2 for 2-D Odd Parity
1

The Even encoded data parity is 101110
The Even encoded data parity is 111010
The Even encoded data parity is 010100
```

Image 1.6 C Program Output for 2-D Parity (Even)



```
-------------------------
Menu
-------------------------
Press 1 for 1-D Parity
Press 2 for 2-D Parity
Press 3 to Quit
-------------------------
2
Enter the Number of Rows : 2
Enter the data in 0's and 1's :
10111
11101
Choose the method to encode the give data
Press 1 for 2-D Even Parity
Press 2 for 2-D Odd Parity
2

The Odd encoded data parity is 101111
The Odd encoded data parity is 111011
The Odd encoded data parity is 101011
```

Image 1.7 C Program Output for 2-D Parity (Odd)



```
-------------------------
Menu
-------------------------
Press 1 for 1-D Parity
Press 2 for 2-D Parity
Press 3 to Quit
-------------------------
3
Program ended with exit code: 0
```

Image 1.8 C Program Output Menu and Exited

4. Analysis and Discussions

- **Why can this method not be used to correct errors?**

We can detect single errors with a **parity bit**. The parity bit is computed as the exclusive-OR (even parity) or exclusive-NOR (odd parity) of all of the other bits in the word. Thus, the resulting word with a parity bit will always have an even (for even parity) or odd (for odd parity) number of 1 bits in it. If a single bit is flipped in transmission or storage, the received data will have the wrong parity, so we will know something bad has happened.

- **How are 1D and 2D parity different?**

Parity checking can be **one-dimensional or two dimensional**. In a single parity-check code, an extra bit is added to every data. It can also detect multiple errors only if the total number of errors in a data unit is odd.Now, if an error occurs, we detect a parity error in both a row and a column. This allows us to localise the bit which is in error, using far fewer bits. Since it is only a bit which is in error, we can simply flip the bit to correct the error. then, as the length of the rows increases the redundancy becomes small.

- **What are the limitations of this method of error detection?**

So the limitations would be that we can't tell which bit was corrupted or if it was just the parity bit that was corrupted. Double errors go undetected, triple errors get detected, quadruple errors don't, etc. Random garbage has a 50% probability of being accepted as valid. So basically this is the issue of using 1-D or 2-D parity to correct errors.

5. Conclusion

One of the methods used for bit error detection is parity checking. There are two kinds of parity checking: 1. Even parity (When the number of ones is even). 2. Odd parity (when the number of ones in the data is odd).

5. Comments

a. Limitations of the experiment

The limitation for parity bit is only guaranteed to detect an odd number of bit errors, meaning that if there are 2 errors in a byte the parity detection will think the byte has no error.

b. Limitations of the results obtained

When send the data you can’t know whether the data is correct or not cause there might be chance that data gets manipulated.

c. Learning

Learned about how to solve even and odd parity for 1-D and 2-D input datas