Exercise : 6

Hamming Code

Aim: Write a program to implement error detection and correction using hamming code concept make a test run to input data stream and verify error correction feature.

```
#include <stdio.h>
#include <string.h>
#include <math.h>

Void charToBinary (char ch, int Binary [],
                   int index) {
    for (int i = 7; i >= 0; i--) {
        binary [(*index)++] = (ch >> i) & 1;
    }
}

Void calculateparityBits (int hamming code [],
                          int n, int r) {
    for (int i=0; i<r; i++) {
        int partypos = (int) pow (2, i);
        int parity = 0;
        for (int j = partypos; j <= n; j = (2 *
                                            Partypos)) {
            for (int k = j; k < j + partpos && k <= n; k++) {
                Parity = hamcode [k];
            }
        }
        hamcode [Parity pos] = Parity;
    }
}
```

```
int generatehamcode (int dbit [], int m, int hamcode [] ){
    int r=0; int n=m;
    while (m + r + 1 > pow(2, r)){
        r++;
    }
    n = m + r;
    for (int i=1 ; j=0; k=0; i<=n; i++){
    if (i == (int)power (2,k){
    hamcode [i] = 0;
    k++;
    } else {
        hamcode [i] = dbits [i++];
    }
}
Cal paritybits (hamcode n,r);
return n;
}
int detand and error (int hamcode[] , int n, int r){
    int error pos = 0;
    for (int i=0; i<r; i++){
    int paritypos = (int) pow(2,i);
    int parity = 0;
    for (int j = paritypos; j<=n; j+= (2 * paritypos)){
        for ( int k = j; k<j + paritypos && k<=n; k++){
            Parity = hamcode [k];
        }
    }
    if (parity != 0){
            error pos += paritypos;
    }
}
return error pos;
}
```

```
void binary to char(int binary[], int length, char
                                        output[]){
    int index = 0;
    for (int i = 0; i < length; i += 8) {
     char ch = 0;
     for (int j = 0; j < 8; j++) {
       ch |= (binary[i+j] << (7-j));
    }
    output[index++] = ch;
    }
     output[index] = '\0'; }
int main() {
    char input string[32];
    int binary[256];
    int data bits[256];
    int hamming code[312];
   Printf("enter the string (upto 99 characters): ");
   Scanf("%s", input string);
   int index = 0;
   for (int i = 0; i < strlen(input string); i++) {
     Char to Binary(input string[i], binary, & index); }
   for (int i = 0; i < index; i++) {
       data bits[i] = binary[i];
    }
   int n = generate hamming code(data bits, index,
                                        hamming code);
   Printf("encoded hamming code: ");
   for (int i = 1; i <= n; i++) {
       Printf("%d", hamming code[i]);
    }
       printf("%c", '\n');
```

```
Printf ("enter the position to stimulate error (0 for no error):");
int error pos;
Scanf ("%d", & error pos);

if (error pos > 0 && error pos < = n) {
hamming code [error pos] = ! hamming code [error pos];

Printf (" hamming code with error: ");
for (int i = 1; i < = n ; i++) {
Printf ("%d", hamming code [i]);
}
Print (" \n");
}
int detected errorpos = detect and correct error (
                                            hamming code, n, log2(n+
if (detected error pos = = 0) {
Pf (" no error detected \n");
}
else {
Printf ("error detected at position: %d \n", detected error pos);
int orginal bit + = ! hamming code [detected error pos];
hamming code [detected error pos] = orginal bit ;
Printf (" corrected hamming code : ");
for (int i = 1; i < = n ; i++) {
Printf ("%d", hamming code [i]);
}
Printf (" \n");
Printf (" corrected bit at position %d : %d \n",
                                        detected error pos,
                                            orginal bit );
}
```

```
int corrected Databits[256];
    int j=0; k=0;
for (int i=1; i<=n; i++){
  if (i != (int pow(2,k))){
    Corrected Databits[i+1] = hamming code[i];
  }else{
     k++;
  }
}
char corrected string[32];
  binary to char (corrected Databits, j, corrected string);
  Printf ("corrected string: %s \n", corrected string);
  return 0;
}
```

Enter String: aaaa

Encoded hamming code: 10001100000101110000101100001010100001

Enter position to stimulate error: 2

Error!

Hamming Code with error: 11001100000101110000101010100001

Error detect position: 2

Corrected hamming code: 10001100000101110000101000100001

Corrected bit at position 2: 0

Corrected string: aaaa

Result:

Thus, the program is successfully executed and verified.

12/9/24