Exp.no: 06

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.

Error Correction at data link layer:

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is transmitted from the Sender to the receiver. It is a technique developed by R.W. Hamming for error correction.

Create a Sender program with below features:

1] Input to Sender file should be a text of any length program should convert the text to binary
2] Apply hamming code concept on the binary data and add redundant bits to it.
3] Save this output in a file called channel,

Create a receiver program with below features:

1] Receiver program should read the input from channel file.
2] Apply hamming code on the binary data to check for errors
3] If there is an error display the position of the error.
4] Else remove the redundant error bits and Convert the binary data to ascii and display the output.

>>>

Student observation:

```
#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 Calparitybits(int hamcode[], int n, int r){
    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];
            }
        }
        hamcode [paritypos] = parity;
    }
}

int generatehamcode (int dbits[], int m, int hamcode[]){
    int r=0; int n=m;
    while (n+r+1 > pow(2, r)){
        r++;
    }
    n=m+r;
    for(int i=1, j=0, k=0; i<=n; i++){
        if (i== (int)pow (2,k)) {
            hamcode [i] =0;
            k++;
```

>>>

```
    } else {
        hamcode[i] = dbits[j++];
    }
  }
  calparitybits(hamcode, n, r);
  return n;
}

int detandcorerror(int hamcode[], int n, int r){
    int errorpos = 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) {
           errorpos += paritypos;
      }
    }
    return errorpos;
}

void bintochar(int binary[], int length, ~~int~~ 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++] = ~~ch~~ '\0';
}
```

```
int main(){
 char inputString [32];
 Int binary [256];
 int dbits [256];
 Int hamcode [512];
 printf ("Enter the input string:");
 Scanf ("%s", inputString);
 int index=0;
 for(int i=0; i< strlen(inputString); i++){
    CharToBinary (inputString[i], binary, &index);
 }
 for (int i=0; i< index; i++){
    databits[i] = binary [i];
 }
 int n= generatehamcode (databits, index, hamcode);
 printf ("Generated Hamming Code:");
 for (int i = 1; i<=n; i++){
    printf ("%d", Hamcode [i]);
 }
 printf ("\n");
 printf ("Enter the position to simulate error");
 int errorpos;
 Scanf ("%d", &errorpos);

if (errorpos >0 && errorpos <=n){
   hamcode [errorpos] = !hamcode [errorpos];
   printf (" Hamming code with error:");
   for (int i=1; i<=n; i++){
      printf ("%d", ham~~ming~~ code[i]);
   }
   printf ("\n");
}
```

```
int deterrorpos = detandcorerror(hamcode, n, log2(n
if (deterrorpos == 0) {
    printf("No error detected.\n");
}
else {
  printf("error detected at position: %d\n", deterrorpos
  int orgbit = !hamcode[deterrorpos];
  hamcode[deterrorpos] = orgbit;
  printf("corrected Hamming code:");
  for (int i = 1; i <= n; i++) {
      printf("%d", hamcode[i]);
  }
  printf("\n");
  printf("corrected bit at position %d: %d\n",
          deterrorpos, orgbit);
}

  int correctedDatabits[32];
  int j = 0, k = 0;
  for ( int i = 1; i <= n; i++) {
     if (i != (int)pow(2, k)) {
         correctedDatabits[j++] = hamcode[i];
  }
  else {
       k++;
  }
}

char correctedString[32];
binaryTochar(correctedDataBits, j, correctedString);
printf("corrected String: %s\n", correctedString);
return 0;
}
```

output:

```
Enter the input String: hey

Generated Hamming code: 0001110110000011100101011110

Enter the position to stimulate error: 3

Hamming Code with error: 00111101100001110010101111001

Error detected at position: 3

Corrected Code: 0001111011000001110010101111001

Corrected bit position 3: 0

Corrected String: hey
```

Result:

Thus the program to implement HAMMING CODE error detection and Correction is execut and output is verified.