DONE | Page No.
Date 21/9/2024

Practical - 6

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.

Hamming code is a set of error-correction code 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.

Code:

```
#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 hammingCode[], 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 ^= hammingCode[k];
            }
        }
    }
}
```

```
hamming code [parity pos] = parity;
}
}
int generated hamming code (int dataBits[],
int m, int hamming code [])
{
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) pow(2,k)) {
hamming code [i] = 0;
k++;
}
else {
hamming code [i] = databits [j++];
}
}
calculate parity bits (hamming code, n, r);
return n;
}
int detect and correct error (int
hamming code [], int n, int r) {
int error pos=0;
for (int i=0; i<r; i++) {
int parity pos = (int) pow(2,i);
int parity=0;
for (int j = parity pos; j<=n; j+=(2* parity pos)) {
for (int k=j; k<j+ parity pos && k<=n; k++)
parity ^= hamming code [k];
}
if (parity != 0) {
error pos += parity pos;
```

```
}
{ 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; i<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 [512];
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 ("\n");
```

```
printf("enter the position to stimulate
        error (0 for no error): ");
int errorpos;
scanf("%d", &errorpos);
if (errorpos > 0 && errorpos <= n) {
hamming code[errorpos] = !hamming
        code[errorpos];
printf("hamming code with error: ");
for (int i = 1; i <= n; i++) {
    printf("%d", hamming code[i]);
}
printf("\n");
}
int detected Error pos = detect and
        correct error (hamming
        code, n, log2(n+1));
if (detected error pos == 0) {
    pf("no error detected.\n");
}
else {
 pf("error detected at position: %d\n",
        detected error pos);
int original Bit = !hamming code [detected
        error pos];
hamming code [detected error pos] =
        original bit;
printf("corrected hamming code: ");
for (int i = 1; i <= n; i++) {
    pf("%d", hamming code [i]);
}
pf("\n");
pf("corrected bit at position: %d:
        %d\n", detected Error pos,
original 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 [j++] = hammingcode [i];
    } else {
        k++;
    }
}
Char corrected string [32];
binary To char (corrected DataBits, i,
                corrected string);
printf ("corrected string : %s \n",
                correctedstring);
return 0;
}
```

Enter String : aaaa
encoded hamming code: 10001100000101111
0000101100001010100001
enter position to ~~flip~~ simulate error : 2
error !
Haming code with error 11001100000101111
0000101100001010100001
error detected position : 2
corrected haming code : 10001100000101
0000101100001010100001
corrected bit at position 2 : 0
corrected String : aaaa

Result:
Thus the program is successfully executed and output is verified.

11/9/12