Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester V

16 September 2020

UCS1511 - Networks Laboratory

Exercise 8: Hamming Code

Objective:

Construct **Hamming Code** for a given binary data.

Code:

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

int power(int num, int exp){
    int pdt = 1;
    while(exp--){
        pdt *= num;
    }
    return pdt;
```

```
}

void strrev(char* s){
    int len = strlen(s);
    int i = 0;
    int j = len-1;
    while(i<j){
        char tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
        i++;
        j--;
    }
}

int checkBinary(char *code){
    int check = 1;
    for(int i = 0; code[i] && check; i++){
        if(code[i] != '1' && code[i] != '0')
            check = 0;
    }
    return check;
}

char* conv_to_bin(int number){
    char *bin = (char*)calloc(100, sizeof(char));
    int n = number;
    int pos=0;
    while(n>0){
        bin[pos++] = ('0'+(n%2));
        n /= 2;
    }
    bin[pos] = '\0';
    strrev(bin);
    return bin;
}

int check_position(int number, int position){
    char *bin=(char*)calloc(100, sizeof(char));
    strcpy(bin, conv_to_bin(number));
    int len = strlen(bin);
    return (bin[len - position]=='1')? 1 : 0;
}

int main(){
```

```
56      char *input = (char*)calloc(100, sizeof(char));
57      char *ecode = (char*)calloc(100, sizeof(char));
58      char *ocode = (char*)calloc(100, sizeof(char));
59
60      printf("\n Enter input data: ");scanf(" %[^\n]", input);
61      while(!checkBinary(input)){
62          printf("\nNot a binary stream. Please re-enter.\n");
63          scanf(" %[^\n]", input);
64      }
65
66      int ip_len = strlen(input);
67      int red_bits = 0;
68      for(int i = 0; i<100; i++){
69          int lhs = power(2.0, i);
70          int rhs = ip_len + i + 1;
71          if( lhs >= rhs){
72              red_bits = i;
73              break;
74          }
75      }
76      char *ip = (char*)calloc(100, sizeof(char));
77      strcpy(ip, input);
78      strrev(ip);
79      int code_len = ip_len + red_bits;
80      //Assign data bits
81      int ip_ctr = 0;
82      for(int i = 0;i<code_len;i++){
83          int ham_bit = 0;
84          for(int j = 0; j < code_len && !ham_bit; j++){
85              if((i+1) == power(2, j))
86                  ham_bit = 1;
87          }
88          if(ham_bit){
89              ecode[i] = '0';
90              ocode[i] = '0';
91
92          }
93          else{
94              ecode[i] = ip[ip_ctr];
95              ocode[i] = ip[ip_ctr];
96              ip_ctr++;
97          }
98      }
99
100     //Hamming code
```

```
    int pos = 0; //Position to check in binary value
    for(int i = 0;i<code_len;i++){
        int ham_bit = 0;
        for(int j = 0; j < code_len && !ham_bit; j++){
            if((i+1) == power(2, j)){
                ham_bit = 1;
                pos += 1;
            }
        }
        if(ham_bit){
            int ctr = 0;
            for(int j = 0;j<code_len;j++){
                int check_pos = check_position(j+1, pos);
                if(ecode[j] == '1'&&check_pos){
                    ctr++;
                }
            }
            ecode[i] = ctr%2? '1':'0';
            ocode[i] = ctr%2? '0':'1';
        }
    }

    //Reversing code
    strrev(ecode); strrev(ocode);
    printf("\nInput data: %s", input);
    printf("\nEven parity code: %s", ecode);
    printf("\nOdd parity code: %s\n", ocode);
}
```

Output:

```
 Enter input data: 1011001

Input data: 1011001
Even parity code: 10101001110
Odd parity code: 10111000101
```
