PRACTICAL NO 6

Name: Saloni Vishwakarma

Batch: C1

Roll no: 13

Aim: Write a c program for error detection and correction in 7-bit hamming code and variable length hamming code.

7-bit hamming code:

Code:

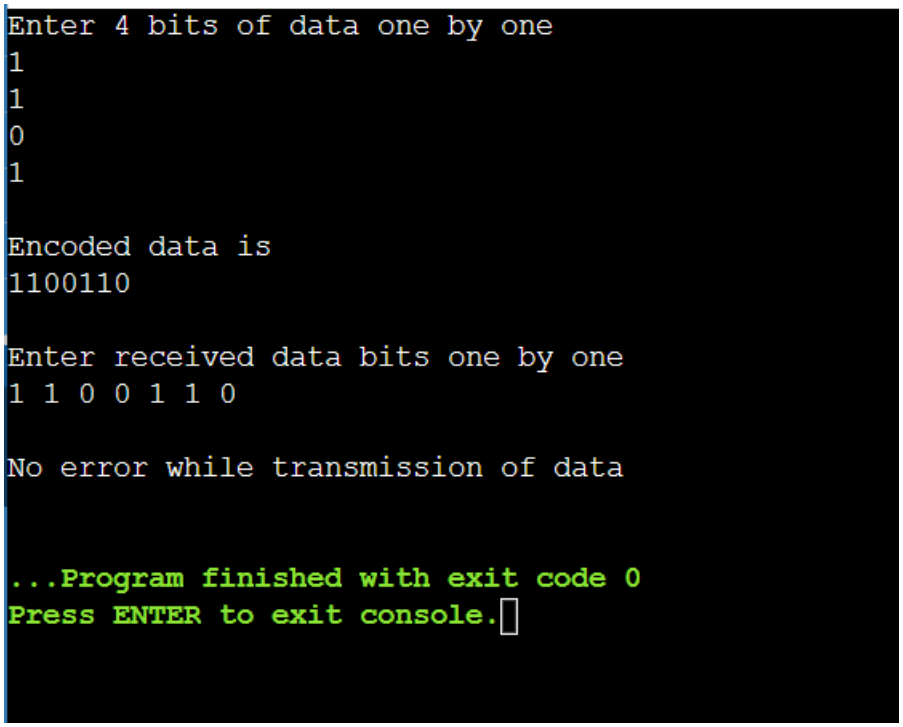
```
#include <stdio.h>
int main() {
   int data[10];
   int datacheck[10],c,c1,c2,c3,i;
   printf("Enter 4 bits of data one by one\n");
   scanf("%d",&data[0]);
   scanf("%d",&data[1]);
   scanf("%d",&data[2]);
   scanf("%d",&data[4]);
   //Calculation of parity
   data[6]=data[0]^data[2]^data[4];
   data[5]=data[0]^data[1]^data[4];
   data[3]=data[0]^data[1]^data[2];

   printf("\nEncoded data is\n");
   for(i=0;i<7;i++)
      printf("%d",data[i]);
   printf("\n\nEnter received data bits one by one\n");
   for(i=0;i<7;i++)
      scanf("%d",&datacheck[i]);
```

```
c1=datacheck[6]^datacheck[4]^datacheck[2]^datacheck[0];
c2=datacheck[5]^datacheck[4]^datacheck[1]^datacheck[0];
c3=datacheck[3]^datacheck[2]^datacheck[1]^datacheck[0];
   c=c3*4+c2*2+c1 ;
   if(c==0) {
      printf("\nNo error while transmission of data\n");
   }
   else {
      printf("\nFound error on position %d",c);
      printf("\nData in : ");
      for(i=0;i<7;i++)
         printf("%d",data[i]);

      printf("\nData out : ");
      for(i=0;i<7;i++)
         printf("%d",datacheck[i]);
      printf("\nCorrect message is\n");
      //if errorneous bit is 0 we complement it else vice versa
      if(datacheck[7-c]==0)
         datacheck[7-c]=1;
      else
         datacheck[7-c]=0;
      for (i=0;i<7;i++) {
         printf("%d",datacheck[i]);
      }
   }
}
```

Output:



Variable length hamming code:

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
void main()
{
   int maxp=6;
   int a[50],temp[70],temp2[70];
   int t,i,j,k,nd,n,nh,sum=0,pos=0;
   printf("Enter Length of Data String: ");
   scanf("%d",&nd);
   printf("Enter Data String: ");
   for(i=0;i<nd;i++)
   {
      scanf("%d",&a[i]);
```

```
}
printf("-----------------------------------\n",nd);
for(i=0,j=0;i<nd;i++)
{
   for(k=0;k<maxp;k++)
   {
      t=pow(2,k)-1;
      if(j==t)
      {
         temp[j]=0;
         j++;
      }
   }
   temp[j]=a[i];
   j++;
}
nh=j;
printf("Length of Hamming code: %d bits\n",nh);
n=nh-nd;
printf("Number of Parity Bits: %d \n",n);
int b[n];
int m=n-1;
for(k=0;k<n;k++)
{
   t=pow(2,k)-1;
   for(i=t;i<nh;)
   {
      for(j=0;j<=t;j++)
      {
         sum=sum+temp[i];
```

```
            i++;
            if(i>=nh)
               break;
         }
         if(i>=nh)
            break;
         for(j=0;j<=t;j++)
         {
            i++;
            if(i>=nh)
               break;
         }
         if(i>=nh)
            break;
      }
      temp[t]=sum%2;
      sum=0;
      printf("P%d: %d\n",t+1,temp[t]);
   }
   printf("\nHamming code: Sender side:   ");
   for(i=0;i<nh;i++)
   {
      printf("%d ",temp[i]);
   }
   printf("\nHamming code: Receiver side: ");
   for(i=0;i<nh;i++)
   {
      scanf("%d",&temp2[i]);
   }
   sum=0;
```

```
for(k=0;k<n;k++)
{
   t=pow(2,k)-1;

   for(i=t;i<nh;)
   {
      for(j=0;j<=t;j++)
      {
         sum=sum+temp2[i];
         i++;
         if(i>=nh)
            break;
      }
      if(i>=nh)
         break;
      for(j=0;j<=t;j++)
      {
         i++;
         if(i>=nh)
            break;
      }
      if(i>=nh)
         break;
   }
   b[m]=sum%2;
   sum=0;
   printf("P%d: %d\n",t+1,b[m]);
   m--;
}
for(m=0;m<n;m++)
```

```
    {
        pos=pos+b[n-m-1]*pow(2,m);
    }
    printf("Position of Error: %d\n",pos);
    if(temp2[pos-1]==0)
        temp2[pos-1]=1;
    else
        temp2[pos-1]=0;
    printf("\nHamming code: Receiver side: Error Corrected:  ");
    for(i=0;i<nh;i++)
    {
        printf("%d ",temp2[i]);
    }
    printf("\n---------------------------------\n",nd);
}
```

Output:

```
main.c: In function 'main':
main.c:28:9: warning: too many arguments for format [-Wformat-extra-args]
main.c:246:9: warning: too many arguments for format [-Wformat-extra-args]
Enter Length of Data String: 4
Enter Data String: 1 0 0 1
---------------------------------
Length of Hamming code: 7 bits
Number of Parity Bits: 3
P1: 0
P2: 0
P4: 1

Hamming code: Sender side: 0 0 1 1 0 0 1
Hamming code: Receiver side: 0 0 1 1 0 0 0
P1: 1
P2: 1
P4: 1
Position of Error: 7

Hamming code: Receiver side: Error Corrected: 0 0 1 1 0 0 1
---------------------------------


...Program finished with exit code 0
Press ENTER to exit console.
```