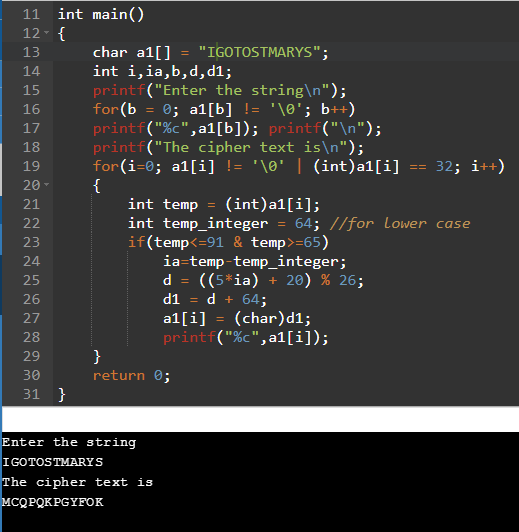
Program for encrypting text “IGOTOSTMARYS” with the formula (5p+20)%26 and finding the enciphered text (cipher text) output where p is position of the character in the alphabetical set



n-redundancy Encoder & Decoder

#include <stdio.h>

#include<stdlib.h>

int main()

{

long long a,b;

int count=0,n,i,j,j1,z=0,\*a1,\*a2,b1;

printf("enter a number");

scanf("%lld",&a);

b=a;

while(b > 0)

{

b = b/10;

count = count + 1;

}

b=a;

printf("count=%d",count);

printf("number is %lld\n",a);

printf("enter the redundant number\n");

scanf("%d",&n);

a1 = (int\*)malloc(sizeof(int) \* (count\*n));

j=count\*n;

j1=j;

//printf("%d\n",j);

for(i=0; i<(count\*n); i++)

{

a1[i]=0;

}

printf("The encoded value for the number is\n");

while(b > 0)

{

b1 = b%10;

b = b/10;//printf("%d\n",b1);

for(int x=1; x<=n; x++)

{

a1[j-1]=b1;

j--;

printf("a1[%d] = %d\n",j,a1[j]);

}

}

printf("The decoded value for the number is\n");

a2 = (int\*)malloc(sizeof(int) \* count);

for(int i=0, x=0;i<j1; i += n)

{

a2[z]=a1[i];

printf("a2[%d] = %d\n",z,a2[z]);

z++;

}

//for(i=0; i<count; i++)

//{

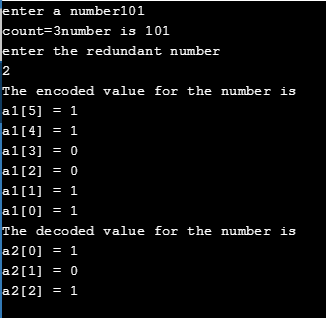
// printf("%d\n",a2[i]);

//}

return 0;

}

Output:



Random message generation and checking whether the error detection happened or not

#include <stdio.h>

#include<stdlib.h>

int main()

{

int random\_number=80;

int random\_data[6], message[3];

for(int j=0;j<6;j++)

{

if(j==0)

random\_data[j]=random\_number/64%2;

else if(j==1)

random\_data[j]=random\_number/32%2;

else if(j==2)

random\_data[j]=random\_number/16%2;

else if(j==3)

random\_data[j]=random\_number/8%2;

else if(j==4)

random\_data[j]=random\_number/4%2;

else

random\_data[j]=random\_number/2%2;

}

printf("The random message is:");

for(int j=0;j<6;j++){

printf("%d",random\_data[j]);}

printf("\n");

message[0]=random\_data[3] ^ random\_data[1];

message[1]=random\_data[5] ^ random\_data[2];

message[2]=random\_data[4] ^ random\_data[0];

if(message[0] != random\_data[0] && message[1] != random\_data[1] && message != random\_data[2])

{

printf("Error is detected in the message\n");

}

else

{

printf("Error is not detected in the message\n");

}

//printf("Hello World");

return 0;

}

Output:





n-bit redundancy encoder & decoder with noise inputs

#include<stdio.h>

#include<stdlib.h>

#define MAX\_COUNT 256

#define MAX\_REDUNDANCY\_VALUE 256

static void encode\_fn(const char \*message, int message\_len, int redundancy\_value, char \*encode) {

   int x;

   int k = 0;

   int j = 0;

   //while loop for encoding message data with n-bit redundancy value

   while (k < message\_len){

       for (x = 0; x < redundancy\_value; x++)

       {

           encode[j] = message[k];

           //printf("encode[%d] = %c\n",j,encode[j]);

       printf("%d,%d,%d\n",x, j, k);

           j++;

       }

       k++;

   }

   printf("encode: %s\n", encode);

}

int main()

{

   //initalizing value of redundancy for encoding (redundancy\_value), length of the message data //(count), and random noise length (random)

   int count = 0;

   int count1 = 0;

   int redundancy\_value = 0;

   int random = 0;

   int random1 = 0;

   int j = 0;

   int k = 0;

   int x;

   int i;

   char c;

   //initializing message data (message), encoded data (encode), noise input, encoded data with //noise, and decoded data

   printf("Enter the length of the message: ");

   //scanf("%d", &count);

   count = 3;

   printf("The length of the message = %d\n", count);

   printf("Enter the message: \n");

   char message[MAX\_COUNT] = {0};

   message[0] = '1';

   message[1] = '0';

   message[2] = '0';

   //for loop to accept the message data input from the user (message)

   for(i = 0; i < count; i++){

       //scanf("%c", &c);

       //message[i] = c;

       printf("message[%d] = %c\n", i, message[i]);

   }

   redundancy\_value = 2;

   printf("Enter the redundancy value: %d\n", redundancy\_value);

   //scanf("%d",&redundancy\_value);

   count1 = count;

   char encode[256] = {0};

   encode\_fn(message, count, redundancy\_value, encode);

   //for loop for printing the encoded data

   for (x = 0; x < (count \* redundancy\_value); x++){

       printf("encode[%d] = %c\n", x, encode[x]);

   }

   random = 2;

   printf("Enter the length of the random noise: %d\n", random);

   //scanf("%d\n",&random);

   printf("Enter the noise value\n");

   char noise[MAX\_COUNT] = {0};

   char encoded\_noise[256] = {0};

   char decoded\_data[256] = {0};

   //Accepting the user noise input with the random noise length (random)

   noise[0] = '1';

   noise[1] = '0';

   //for(int z=0; z<random; z++){

   //    scanf("%c",&noise[z]);

   //}

   printf("The data sent with noise is: %s\n", noise);

   //strcat(encoded\_noise,encode,noise);

   random1 = random;

   //Adding the noise value with the encoded data

   for(i = 0; i < ((count \* redundancy\_value) + random); i++){

       if (i < (count \* redundancy\_value)) {

           encoded\_noise[i] = encode[i];

       }

       else {

           encoded\_noise[i] = noise[--random1];

       }

   }

   printf("encoded\_data: %s\n", encoded\_noise);

   //for(int i=0; i<(count\*redundancy\_value\*random); i++){

   //    printf("encoded\_noise[%d] = %c",i,encoded\_noise[i]);

   //}

   printf("The decoded data sent with noise is\n");

   printf("redundancy\_value: %d\n", redundancy\_value);

   //for loop to decode the encoded noise data recovering the original message with noise data

   for (i = 0, x = 0; i < (count \* redundancy\_value); i += redundancy\_value)

   {

       printf("i: %d, x: %d\n", i, x);

       decoded\_data[x++]=encoded\_noise[i];

   }

   char decoded\_noise[256] = {0};

   for (j = random-1; j >= 0; j--, i++) {

       decoded\_noise[j] = encoded\_noise[i];

   }

   printf("decoded\_data: %s\n", decoded\_data);

   printf("decoded noise: %s\n", decoded\_noise);

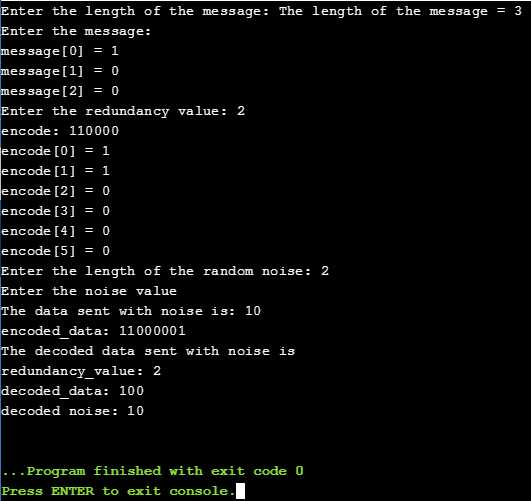
   //length = strlen(a);

   //printf("%d",length);

   return 0;

}

Output:



Program for cyclic redundancy check to check for the given message transmitted with errors or not

#include<stdio.h>

#include<string.h>

#define Number strlen(generator)

char original\_data[28],checksum[28],generator[]="10001000000100001";

int count,error\_value,c;

void computation\_xor(){

   for(c = 1;c < Number; c++)

   checksum[c] = (( checksum[c] == generator[c])?'0':'1');

}

void cyclic\_redundancy\_check(){

   for(e=0;e<Number;e++)

       checksum[error\_value]=original\_data[error\_value];

   do{

       if(checksum[0]=='1')

           computation\_xor();

       for(c=0;c<Number-1;c++)

           checksum[c]=checksum[c+1];

       checksum[c]=original\_word[error\_value++];

   }while(e<=count+Number-1);

}

int main()

{

   printf("\nEnter data : ");

   scanf("%s",original\_data);

   printf("\n----------------------------------------");

   printf("\nGenerating polynomial : %s",generator;

   count=strlen(original\_data);

   for(error\_value=count;error\_value<count+Number-1;error\_value++)

       original\_data[error\_value]='0';

   printf("\n----------------------------------------");

   printf("\nModified data is : %s",original\_data);

   printf("\n----------------------------------------");

   cyclic\_redundancy\_check();

   printf("\nChecksum is : %s",checksum);

   for(error\_value=count;error\_value<count+Number-1;error\_value++)

       original\_data[error\_value]=checksum[error\_value-count];

   printf("\n----------------------------------------");

   printf("\nFinal codeword is : %s",original\_data);

   printf("\n----------------------------------------");

   printf("\nTest error detection 0(yes) 1(no)? : ");

   scanf("%d",&error\_value);

   if(e==0)

   {

       do{

           printf("\nEnter the position where error is to be inserted : ");

           scanf("%d",&error\_value);

       }while(error\_value==0 || error\_value>count+Number-1);

       original\_data[error\_value-1]=(original\_data[error\_value-1]=='0')?'1':'0';

       printf("\n----------------------------------------");

       printf("\nErroneous data : %s\n",original\_data);

   }

   Cyclic\_redundancy\_check();

   for(error\_value=0;(error\_value<Number-1) && (checksum[error\_value]!='1');e++);

       if(e<N-1)

           printf("\nError detected\n\n");

       else

           printf("\nNo error detected\n\n");

           printf("\n----------------------------------------\n");

       return 0;

}

Output:

