```c
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char S[30];
//***********************************
int longueur(char S[]){
    int i=0;
    while(S[i]!='\0')
    i++;
    return (i);

    }
//************************************
// question1
int ChaineChiffre(){
    int i=0;
    while(S[i]!='\0')
    if(S[i]<'0' || S[i]>'9')
    return(0);
    else
    i++;
    return 1;

    }
//************************************
// question 2
void supprimer_zeros(){
     int i,j;
     i=0;
     while (S[i]!='\0' && S[i]=='0')
     i++;
     if (S[i]!='\0'){
     j=0;
     while(S[i]!='\0'){
     S[j]=S[i];
     i++;j++;
     }
     S[j]=S[i];
     }
}
//************************************
//question 3
void additionner(char S1[],char S2[],char SOM[]){
     int n1,n2,nsom,q,r,i,j,k;
     n1=longueur(S1);
     n2=longueur(S2);
     if(n1>n2)
     nsom=n1;
     else
     nsom=n2;
  //   nsom++;
     SOM[nsom]='\0';i=n1-1;j=n2-1;r=0;
     for(k=nsom-1;k>=0;k--)
     if (i>=0 && j>=0)
     {
        q=r+S1[i]+S2[j]-96;
        r= q
        q=q/10;
        i--;j--;
        SOM[k]=r+48;
        r=q;
        }
```

```
     else if(i<0){
          q=r+S2[j]-48;
          r=q%10;
          q=q/10;
          SOM[k]=r+48;
          j--;
          r=q;
          }
          else if(j<0){
               q=r+S1[i]-48;
               r=q%10;
               q=q/10;
               SOM[k]=r+48;
               i--;
               r=q;
               }
          else
          { SOM[k]=r+48;
            r=0;
              }
          }

//**************************** partie B***************
//la structure
typedef struct Liste{
        short int partie;
        struct Liste *suiv;
        }ListeNombres;
//*********************************
// question 4
ListeNombres * regrouperNombres1( char * S0){
             int i,j;short int q; ListeNombres * p,*l,*r;

   i=0;j=0;q=0;l=NULL;
   while (S0[i]!='\0'){
         if(j==4)
         {j=0;
         p=(ListeNombres *)malloc(sizeof(ListeNombres));
         p->partie=q;
         p->suiv=NULL;
         if (l==NULL){l=p;
         r=l;}
         else
            {r->suiv=p;
            r=p;}
          q=0;
           }
         else
         {q=q*10+(S0[i]-48);
         j++;i++;
         }

       }
if (q!=0){
    p=(ListeNombres *)malloc(sizeof(ListeNombres));
         p->partie=q;
         p->suiv=NULL;
         if (l==NULL){l=p;r=l;}
         else
            {r->suiv=p;
            r=p;}

   }
```

```
return l;
}
//***********************************************
// fonction pour afficher les elements de la liste Nombres
void afficherListe(ListeNombres *l){
     ListeNombres *q;
     q=l;
     while(q!=NULL){
 printf("%d-",q->partie);
 q=q->suiv;
     }
 }
//***************************************************
// Probleme II
//***************************************************
//: partie A
// phase 1
//***************************************************
#define L1 100
 #define L2 100
 char Claire[L1];
 char Clef[L2];
 char Chiffree[L2];
 char K[256];
 int s[256];
 //***********************************************
 // question 1 inittialisation du tabeau s à l'identité
 void identiteS(){
      int i;
      for (i=0;i<256;i++)
      s[i]=i;

      }
 //***********************************************
 //Question 2 initialisation du tableau K avec la clé
 void initialiserK(){
      int i;
      for (i=0;i<256;i++)
      K[i]=Clef[i % L2];

      }
  //***********************************************
 //Question 3 permutation s[i] et s[j]
 void permuterS(int i, int j){
      int c;
      c=s[i];
      s[i]=s[j];
      s[j]=c;
      }

 //***********************************************
 //Question 4 KSA()tion du tableau K avec la clé
 void KSA(){
      int i,j;
      identiteS();
      initialiserK();
      j=0;
      for(i=0;i<255;i++)
      {
            j=(j+s[i]+K[i]) % 256;
            permuterS(s[i],s[j]);

                        }
```

```
      }
 //************************************************
 //phase 2
 //*********************************************

 //question5
 void decomposer(int bit[],int nombre){
      int i,b;
      b=nombre;
      for (i=0;i<8;i++){
          bit[i]=b%2;
          b/=2;
          }
      }
 //*************************************
//question 6
//*************************************
int deuxPuissance(int n){
    if (n!=0)
    return 2*deuxPuissance(n-1);
    return 1;

    }
//************************************************
//question 7
int decimale(int bit[]){
    int nombre=0,i;
    for (i=0;i<8;i++)
    nombre+=bit[i]*deuxPuissance(i);
    return nombre;

    }
//************************************************
//question 8
int ouExclusif(int x,int y){
    int Nx[8],Ny[8],res[8],i;
    decomposer(Nx,x);
    decomposer(Ny,y);
    for (i=0;i<8;i++)
    if(Nx[i]+Ny[i]==1)
    res[i]=1;
    else
    res[i]=0;
    return decimale(res);
        }
//************************************************
//question 9
void PRGA(){
     int i,j,a;
     i=j=0;
     int octet
     for(a=0;a<L1;a++)
     {
              i=((i+1) %256)
              j=((j+Claire[i]) %256);
              permuterS( i,j)
              octet=s[(s[i]+s[j])%256)]  ;
              Chiffree[a]=ouExclusif(clef[a],octet);
     }

     }
//************************************************
```

```
//partie B
//***********************************************
//
char* RC4Chaine(char *Claire,int L,char *Clef){
 // supposée déclarée et définie

      }
//***********************************************
//question 10
void RC4Fichier(char * fich,char * clef,char * fichChiffre){
     FILE *f1,*f2;
     char * LL;
     LL=(char*)malloc(sizeof(char)*80);
     f1=fopen(fich,"r");
     f2=fopen(fichChiffre,"w");
     while(fgets(LL,80,f1)!=NULL)
     {
                                LL=RC4Chaine(LL,80,clef);
                                fputs(LL,f2);

     }
     fclose(f1);
     fclose(f2);
  }
//***************************************************

main(){
       int bit[8],i;
    char   S1[]="00000129782004977\0";
       char S2[]="5433298776754022999953\0";
       strcpy(S,"00000012456323987\0");supprimer_zeros( );
      ListeNombres* l;
       printf("%s\n%s\n",S1,S2);
       additionner(S1,S2,S);
       supprimer_zeros();
      // printf("%s\n",S);
      l= regrouperNombres1(S1);
       afficherListe(l);
       decomposer(bit,27);
       for (i=0;i<8;i++)
       printf("\n%d:",bit[i]);
    printf("%d\n",deuxPuissance(20));
    printf("%d\n",ouExclusif(5,6));
       getch();

       }
```