Une proposition d'une solution pour le CNC en informatique 2011:

```
#include<stdio.h>
#include<conio.h>
#include <stdlib.h>
#define N 7
#define N1 5
#define N2 4
int T[N];      //={2,3,23,14,25,7};//,8,7};//,8,90,54,6};
int T1[N1];    //={2,14,28,75,88};
int T2[N2];    //={1,6,14,28};//2,4,5,9,17,19};
int valide;
```

//Question 1 Vérification d'un Ensemble Tableau

```
int estEnsembleTailleN(int T[]){
   int i,j,e;
   e=1;
   for (i=0;i<N-1;i++)
      for(j=i+1;j<N;j++)
       if (T[i]==T[j] || T[j]<=0)
       // return 0;
       e=0;
       //return 1;
      if (e==0)
       valide=0;
      else
       valide=1;
       return e;
       }
```

//Question 2 Appartenance à un ensemble Tableau

```
 int  appartient(int T[],int x){
     int i;
     i=0;
     while (x!=T[i] && i<N)
          i++;
     if (i!=N)
          {
               printf("%d appartient à T",x);
               return 1;
          }
     else
        {
          printf("%d n'appartient pas à T",x);
```

```
          return 0;
        }
          }
```

//Question 3 Tri d'un Ensemble Tableau : algorithme tri par selection

```
void triASelection(int T[]){
    int i,j,k;
    for (i=0;i<N-1;i++)
       for(j=i+1;j<N;j++)
         if (T[i]>T[j])
         {
           k=T[i];
           T[i]=T[j];
           T[j]=k;
           }
    }
```

//question 3 1 tri d'un ensemble Tableau : algorithme tri a bulle

```
 void triABulle(int T[]){
    int i,j,k;
    for (i=N-1;i>=0;i--)
       for(j=0;j<i;j++)
         if (T[j]>T[j+1])
         {
           k=T[j];
           T[j]=T[j+1];
           T[j+1]=k;
           }
    }
```

//Question 4 Inclusion d'un ensemble dans un autre

```
int T1inclusdansT2() {
   int j=0;
  while (appartient(T2,T1[j]) && j<N1)
       j++;
  if(j==N1)
   return 1;
   else
   return 0;
   }
```

// Question 5 Union de deux ensembles tableaux triés

```
void union1 (int T[], int T1[],int T2[])
{
```

```
    int i,j,e;
    i=0;j=0;

    for (e=0;e<N;e++)
    if (i==N1)
    {
       T[e]=T2[j];
           j++;
     }
     else
     if (j==N2)
     {
      T[e]=T1[i];
                   i++; }
     else
       if(T1[i]<T2[j])
                  {
                   T[e]=T1[i];
                   i++;
                  }
       else {
           if(T1[i]==T2[j])  i++;   //pour ne pas avoir des répetitions
           T[e]=T2[j];
           j++;

           }

}
```

// Question 5 Union de deux ensembles tableaux triés (aurre methode)

```
void union11 (int T[], int T1[],int T2[])
{
    int i,j,e;
    i=0;j=0;e=0;
    while(i<N1 && j<N2){
        if(T1[i]<T2[j])
                  {
                   T[e]=T1[i];
                   i++;
                  }
       else {
           if(T1[i]==T2[j])  i++;   //pour ne pas avoir des répetitions
```

```
            T[e]=T2[j];
            j++;

            }
           e++;
}
while(i<N1) {T[e]=T1[i]; i++;e++;}
while(j<N2) {T[e]=T1[j]; j++;e++;}
}
```

//Partie B

```
typedef struct ens
      {int nombre;
      struct ens * suiv;
      }ensembleListe;
```

//Question 6

```
ensembleListe *p;
void inserer(int val){
   ensembleListe *q;
   ensembleListe *r;
   r=(ensembleListe*)malloc(sizeof(ensembleListe));
   r->nombre=val;
   r->suiv=NULL;
   q=p;
   while (q->suiv!=NULL && q->suiv->nombre<val)
    q=q->suiv;

   if(q->suiv==NULL)
               q->suiv=r;
   else
   if (q->suiv->nombre>val){
               r->suiv=q->suiv;
               q->suiv=r;
               }
}
```

//Probleme II

```
const int L=10;
const int NB=5;
```

//Question 1

```
int distanceH(char S1[],char S2[],int M){
```

```
   int i,nombre;
   nombre=0;
   for (i=0;i<M;i++)
   if (S1[i]!=S2[i]) nombre++;
   return nombre;
   }
```

//Question 2

```
int distanceH_langage(char langage[NB][L] ){
   int i,j,min,valeur;
  //  min=distanceH(langage[0][L],langage[1][L],L);
   for (i=0;i<NB-1;i++)
    for(j=i+1;j<NB;j++)
   {// valeur=distanceH(langage[i][L],langage[j][l],L);
    if (min>valeur) min=valeur;
    }
    return min;
}
```

//Question 3

//3-a

```
void binaire(char * bin , int NP){
    int P,R,j;
   // bin=(char*)malloc(9*sizeof(char));
    P=NP;j=7;
    while (P!=0){
        R=P % 2;
        *(bin+j)=R+48;
        P=P/2;j--;
        }
   while(j>=0) {*(bin+j)='0';j--;}
   *(bin+8)='\0';
   printf("%s\n",bin); //pour afficher la représentation binaire du nombre
    }
```

//Question 3

//3-b

```
   int distanceNombre(int A, int B) {
    char *CA,*CB;int i,distance;
     CA=(char*)malloc(9*sizeof(char));
     CB=(char*)malloc(9*sizeof(char));
    binaire(CA,A);
    binaire(CB,B);
```

```
      distance=0;i=7;
      while(*(CA+i)&& i>=0)
      {
       if(*(CA+i)!=*(CB+i)) distance++;
       i--;
      }
   return distance;

      }

// bloc principal pour tester les différentes fonctions
int main(void){
   int i;
   printf("la distance entre les deux nombre %d,%d est %d",34,54,distanceNombre(34,54));
   /*union11 (T,T1,T2);      //pour tester l'union
   for(i=0;i<N;i++)
   printf("%d : ",T[i]);*/
    /*if (T1inclusdansT2())   //pour tester l'inclusion
     printf("inclus") ;
     else
     printf("non inclus");*/
     getch();
    }
```

//fin