Examen du 17/11/2003

Corrigé

Exercice I

Après les commandes

```
a=1;
for(i=0;i<10;i++)
  tab[i]=a*i;
```

le contenu du tableau `tab` est `0 1 2 3 4 5 6 7 8 9`. La commande `a++;` n'affecte pas le tableau, en revanche la variable `a` contient desormais 2. Après les commandes

```
for(i=0;i<9;i++)
  tab[i]=tab[i+1];
```

le contenu du tableau `tab` est `1 2 3 4 5 6 7 8 9 9`. A la ligne suivante `b` pointe sur la deuxième case du tableau. La commande `*b=-a;-` est une affectation à ce qui est pointé par `b`, donc une affectation à la la deuxième case du tableau. La valeur affectée est `a` décrementée de 1, soit $2 - 1 = 1$. Le tableau `tab` devient `1 1 3 4 5 6 7 8 9 9`.

```
for(i=0;i<10;i++)
  printf("%d\n",tab[i]);
```

affiche les cases du tableau en colonne, en passant à la ligne après chaque case. On obtient

```
1
1
3
4
5
6
7
8
9
9
```

Exercice II

```
#include <stdio.h>
#include <assert.h>

#define N 1000
// N doit etre superieur ou egal a 6 et inferieur a 2^15

typedef short int* ETL;
```

```
void afficherETL (ETL n);

ETL int2ETL (int n)
     /* Converti un entier en ETL
ENTREE : entier n
SORTIE : ETL ayant de valeur n
     */
{
  ETL resultat;
  int i;

  resultat=(short int*)malloc(N*sizeof(short int));

  // Initialisation de l'ETL a 0
  for (i=0;i<N;i++)
    resultat[i]=0;

  i=0;
  while (n!=0) {
    resultat[i]=n%10;     // On affecte le chiffre de n le plus a droite
    n/=10; // On shift tous les chiffres d'un emplacement vers la droite
    i++;
  }

  return(resultat);
}


void afficherETL (ETL n)
     /* Affichage de l'ETL n
ENTREE : l'ETL n
SORTIE : aucune mais il y a affiche a l'ecran
REMARQUE : \n n'est pas envoye a stdout
     */
{
  int i, j;

  // Pour ne pas afficherETL les 0 inutiles on se positionne a la premiere
  // position non nulle
  i=N-1;
  while ((n[i]==0)&&i>=0)
    i--;

  if (i==-1)
    printf("0"); // n=0
  else
    {
      // n non nul : on affiche chaque chiffre
      for (j=i;j>=0;j--)
printf("%d",n[j]);
    }
}
```

```
short int egalETL (ETL n, ETL m)
     /* Comparaison de deux ETL
ENTREE : n et m, les deux ETL a comparer
SORTIE : 1 si n=m, 0 sinon
     */
{
  int i, egal;

  i=0;
  while ((n[i]==m[i])&&(i<N))
    i++;
  if (i==N-1)
    // Si les N-1 premiers chiffres sont egaux alors l'egalite depend du N-eme chiffre
    return(n[N-1]==m[N-1]);
  else
    // Sinon n et m sont differents
    return(0);
}


ETL sommeETL (ETL n, ETL m)
     /* Somme de deux ETL
ENTREE : n et m, les deux ETL a sommer
SORTIE : n+m
     */
{
  int i, somme;
  int c=0; // Variable servant a la retenue
  ETL resultat;

  // Creation de l'ETL resultat
  resultat=int2ETL(0);

  // On fait la somme comme a l'ecole elementaire
  for (i=0;i<N;i++)
    {
      somme=n[i]+m[i]+c;
      resultat[i]=somme%10;
      c=somme/10; // 1 ou 0 selon qu'il y ait une retenue ou pas
    }

  // S'il y a une retenue a la fin c'est qu'il y a eu depassement
  // Dans ce cas on genere une erreur
  assert(c==0);

  return(resultat);
}


void doubleETL (ETL * n_Ptr)
     /* Double un ETL
ENTREE : l'adresse d'un ETL n
SORTIE : aucune mais n est double
     */
```

```
{
  ETL m;

  m=sommeETL(*n_Ptr,*n_Ptr);
  free(*n_Ptr); // cet ETL n'est plus utilise
  *n_Ptr=m;
}


int main() {
  ETL x;
  int i;

  x=int2ETL(1);
  for (i=1;i<=2500;i++)
    doubleETL(&x);
  printf("2^2500=");
  afficherETL(x);
  printf("\n");
}
```