Programmation, TD 2: le langage C

7 octobre 2002

1. Que font les programmes suivants ?

```
main (int argc, char *argv[])
{
  int i;

  for (i=1; i<argc; i++) {
    puts (argv[i]);
  }
}
```

```
main (int argc, char *argv[])
{
  int i;

  for (i=1; i<=argc; i++) {
    puts (argv[i]);
  }
}
```

2. Pourquoi ceci ne met-il pas `i` à 200 ?

```
main ()
{
  char i;

  i = 100;
  i += 100;
  printf ("%d.\n", i); // affichage de i
}
```

3. Quel est le bug qui vous pend au nez dans le programme suivant ?

```
main (int argc, char *argv[])
{
  int i;

  for (i=1; i<argc; i=i++) {
    puts (argv[i]);
  }
}
```

4. Le programme suivant est censé effectuer un tri. Lisez-le, essayez de comprendre comment il fonctionne, et dites-moi pourquoi il ne peut pas fonctionner, et comment le corriger.

```
void merge (int *l1, int n1, int *l2, int n2, int *res)
{ // prend deux tableaux l1 de n1 entrees, l2 de n2 entrees,
  // tous les deux tries, en fait l'union triee dans res.
  while (n1!=0 && n2!=0) {
    if (*l1 < *l2)
      { *res++ = *l1++; n1--; }
    else { *res++ = *l2++; n2--; }
  }
  while (n1--!=0) *res++ = *l1++;
  while (n2--!=0) *res++ = *l2++;
}
```

```
void sort1 (int *l, int n, int *res)
{ // trie un tableau l de n entrees, le trie dans res.
  int k;

  if (n==0) return;
  if (n==1) { *res = *l; return; }
  k = n/2;
  sort1 (l, k, res);
  sort1 (l+k, n-k, res+k);
  merge (res, k, res+k, n-k, l);
}

void sort (int *l, int n)
{
  int *aux = (int *) malloc (n * sizeof (int));
  sort1 (l, n, aux);
  free (aux);
}
```

5. Écrire un programme prenant un argument dans `argv[1]`, le chiffrant par l'algorithme de César, c'est-à-dire en ajoutant 3 à chaque caractère cycliquement : `a` $\mapsto$ `d`, `b` $\mapsto$ `e`, ..., `z` $\mapsto$ `c`.

   Note : en C, `'a'+3` égale `'d'`...

6. On se donne une structure `temps` pour décrire un instant du temps par :

   ```
   struct temps {
     unsigned int min; // minutes
     unsigned int sec; // secondes
     unsigned int cent; // centiemes de secondes
   };
   ```

   Écrire une fonction `ajoute_temps (struct temps *t1, struct temps *t2)` qui ajoute le temps `t1` à la structure de temps `t2`.

7. Écrire un programme `main (int argc, char *argv[])` qui retourne la chaîne de caractères dans `argv[1]` et l'affiche. Par exemple, `abc` en entrée donnera `cba` en sortie.