Université de Bordeaux
Licence STS

ARCHITECTURE DES ORDINATEURS

TP : 04

Structures de données

Exercice 1 : Listes chaînées

En C, pour définir une liste chaînée d'entiers, on utilise typiquement une structure de données de la forme :

```
struct Cell {
  long          x;        /* Charge utile         */
  struct Cell * next;     /* Pointeur sur suivant */
};
```

avec pour convention que, dans le dernier maillon de la liste chaînée, le champ `next` est `NULL`.

Question 1

Définissez une liste chaînée contenant les entiers `1`, `2`, `3` et `4`, et une autre liste chaînée contenant les entiers `10`, `11` et `12`. On encodera la référence `NULL` par la valeur `0`.

Question 2

Écrivez le code C d'une fonction `long sum (struct Cell * list)` retournant la somme des éléments d'une liste chaînée dont l'adresse du premier maillon est le paramètre `list`. Écrivez le code Y86 correspondant.

Question 3

Écrivez une fonction `void concat (struct Cell * list1, struct Cell * list2)` qui fait pointer la dernière cellule de la liste `list1` vers la première cellule de la liste `list2` ; on *concatène* donc les deux listes. Écrivez le code Y86 correspondant (on supposera que la `list1` est non-vide).

Exercice 2 : Arbres

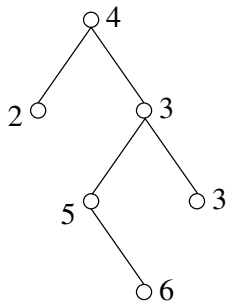
En C, pour définir un arbre, on utilise une structure de données de la forme :

```
struct Node {
  long          x;        /* Charge utile */
  struct Node * left;     /* Fils gauche  */
  struct Node * right;    /* Fils droit   */
};
```

avec pour convention que, si `left` ou `right` contient `NULL`, c'est qu'il n'y a pas de fils de ce côté-là.

Question 1

Définissez l'arbre suivant :



Question 2

Écrivez le code C d'une fonction récursive `long sum (struct Node * tree)` calculant la somme des éléments de l'arbre dont l'adresse de la racine est le paramètre `tree`. Écrivez le code Y86 correspondant.