Examen de TD n°1

La durée de l'examen est de 30 minutes, les notes de cours et de TD sont autorisées. La notation prendra en compte la présentation et la clarté des explications (programmes commentés, dessins explicatifs, ...). L'examen sera noté sur 10 points et le barème est donné uniquement à titre indicatif. Il pourra être modifié lors de la notation finale.

▷ **Question 1. (4 points)** Expliquer le fonctionnement de la pile lors d'un appel de fonction.

Réponse. Les explications ont été données en TD et en cours. On pouvait par exemple expliquer que les paramètres ainsi que les variables locales à la fonction étaient empilés, que lors de la sortie le pointeur de pile diminuait et qu'on n'avait plus accès aux variables locales à la fonction, que les modifications effectuées sur les paramètres n'étaient donc plus visibles une fois sorti de la fonction, ... On pouvait également donner un programme en exemple et faire de petits dessins de la pile, bref réexpliquer ce que l'on avait vu en TD. □

▷ **Question 2. (3 points)** Écrire une fonction C qui renvoie le plus grand commun diviseur de deux nombres. On rappelle que le pgcd de a et de b peut être calculé en utilisant les formules suivantes :

$$\begin{aligned} pgcd(a,b) &= 1 && \text{si } a \text{ ou } b \text{ est égal à } 1 \\ pgcd(a,b) &= a && \text{si } a = b \\ pgcd(a,b) &= pgcd(a-b,b) && \text{si } a > b \\ pgcd(a,b) &= pgcd(a,b-a) && \text{si } b > a \end{aligned}$$

Réponse.

```
1  int pgcd(int a, int b)
2  {
3    if(a==1) return 1;
4    if(b==1) return 1;
5    if(a==b) return a;
6    if(a<b) return pgcd(a,b-a);
7    else return pgcd(a-b,b);
8  }
```

□

▷ **Question 3. (3 points)** On rappelle les définitions suivantes :

```
1  typedef struct s_maillon *p_maillon_t;
2  typedef p_maillon_t       liste_t;
3  typedef liste_t          *p_liste_t;
4  typedef struct s_maillon {
5    int         valeur;
6    p_maillon_t suivant;
7  } maillon_t;
```

Écrire une fonction C qui prend une liste en argument et renvoie la valeur du plus grand élément de cette liste. Vous pourrez utiliser, si vous le désirez les fonctions suivantes :

```
int max(int a, int b) {
  if(a<b) return(b);
  else return(a);
}
liste_t nil() {
  return NULL;
}
int car(liste_t liste) {
  if(liste == nil()) {
    printf("Le car d'une liste vide, c'est pas terrible !!\n");
    exit(1);
  }
  return liste->valeur;
}
liste_t cdr(liste_t liste) {
  if(liste == nil()) {
    printf("Le cdr d'une liste vide... disons vide par convention !\n");
    return(nil());
  }
  return liste->suivant;
}
```

Réponse.

```
int max_liste(liste_t liste) {
  if(liste == nil()) {
    printf("La liste est vide! Il n'y a pas de plus petit élément\n");
    exit(1);
  }
  if(cdr(liste)==nil())
    return car(liste);
  else
    return max(car(liste),max_liste(cdr(liste)));
}
```

□