TD n°13 – Arbres

1 Arbres binaires

1. *Écrire en C une fonction calculant la hauteur d'un arbre binaire. La signature sera* `int hauteur(arbrebin* a)`.

```
int hauteur(arbrebin* a){
  if (a==NULL){return -1;}
  if (hauteur(a->g) > hauteur(a->d)){return hauteur(a->g)+1;}
  return hauteur(a->d)+1;
}
```

2. *Écrire en C une fonction* `int dernier(bintree* a)` *qui renvoie l'étiquette du noeud le plus à droite de l'arbre. (le noeud le plus à droite est le noeud qu'on atteint en allant que à droite)*

```
int dernier(arbrebin* a){
  assert !(a==NULL);
  while (a->d != NULL){
    a=a->d;
  }
  return a->etiquette;
}
```

3. *Écrire en C une fonction* `void echange_sous_arbre(bintree* a)` *qui échange le sous-arbre gauche de la racine et le sous-arbre droit de la racine.*

```
void echange_sous_arbre(arbrebin* a){
  arbrebin* sto = a->g;
  a->g = a->d;
  a->d = sto;
}
```

4. *Écrire en C une fonction* `void miroir(bintree* a)` *qui transforme l'arbre en entrée en son miroir (sa symétrie axiale verticale).*

```
void miroir(arbrebin* a){
  if(a==NULL){return;}
  miroir(a->g);
  miroir(a->d);
  echange_sous_arbre(a);
}
```

2 Arbres enracinés

5. *Écrire une fonction qui réalise un parcours en profondeur préfixe d'un arbre non-binaire, en C. Le traitement consistera en l'affichage des noeuds.*

```
void parcours_prefixe(arbrebin* a){
   if(a==NULL){return;}

   printf("%d", a->etiquette);
   parcours_prefixe(a->g);
   parcours_prefixe(a->d);
}
```

6. **Écrire une fonction qui réalise un parcours en largeur d'un arbre non-binaire, en C.**

```
void parcours_largeur(arbrebin* a){
   file* f = creer_file();
   enfile(a,f);
   while (!est_vide(f)){
     arbrebin b = defile(f);
     printf("%d", b->etiquette);
     if (b->g != NULL){enfile(b->g, f);}
     if (b->d != NULL){enfile(b->d, f);}
   }
}
```