Université de Batna 2 — Algorithmique et Structures de Données
Département d'informatique — 08 Avril 2018
Licence 2 : informatique académique

# Rattrapage

## Exo1 :

- Un nombre binaire est une suite de bits, où chaque bit peut prendre la valeur 0 ou 1.

Q1) Proposer une structure de donnée doublement chainée « **NombreBinaire1 »** qui permet de gérer un nombre binaire d'une manière **Bidirectionnel**.

Q2) Si un pointeur occupe un espace mémoire de 8 octets. Donner le résultat de la fonction :
**sizeof (NombreBinaire1) ;**

---

- Dans un nombre binaire, chaque bit possède un **Rang** qui commence par 0 en partant de la droite.

**EX :**

| **bit** | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Rang** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **Poids = $2^{rang}$** | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

Q3) Proposer une autre structure de donnée **simplement chainée** « **NombreBinaire2 »** qui permet de stocker : les **bits**, les **rangs**, et les **poids** d'un nombre binaire.

- Supposons que les **bits** sont déjà saisis par l'utilisateur, et que le bit de faible poids se trouve dans la tête de liste et le bit de fort poids dans la queue.

Q4) Ecrire une fonction « **CalculeRangPoid (NombreBinaire2* L)** » qui prend en paramètre une liste et qui permet de **calculer et stocker** le Rang et le Poids de chaque bit.

- Supposons maintenant que tous les champs sont remplis (les bits, les Rang et les Poids).

Q5) Ecrire une fonction « **int Convertir (NombreBinaire2* L)** » qui permet de calculer le nombre décimal représenté par la liste des bits.

## Exo2 :

Q1) Construire l'arbre binaire de recherche à partir des éléments suivants : 20,15,10,4,13,30,17,27,11,8,3, 22,29

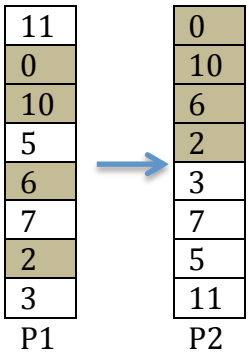
Q2) Donner le résultat du parcours : préfixé, infixé et postefixé de l'arbre binaire résultant de la question précédente.

Q3) Ecrire une fonction qui permet de fusionner deux arbres binaire simple avec un entier X.

Q4) Ecrire une fonction qui permet de trouver un entier X dans un arbre binaire de recherche.

## Exo3 :

Soit P1 une pile d'entiers.
En utilisant seulement des piles, écrire une fonction qui permet de déplacer les entiers de P1 dans une pile P2 de façon à avoir dans P2 **: tous les nombres pairs en dessus des nombres impairs**.



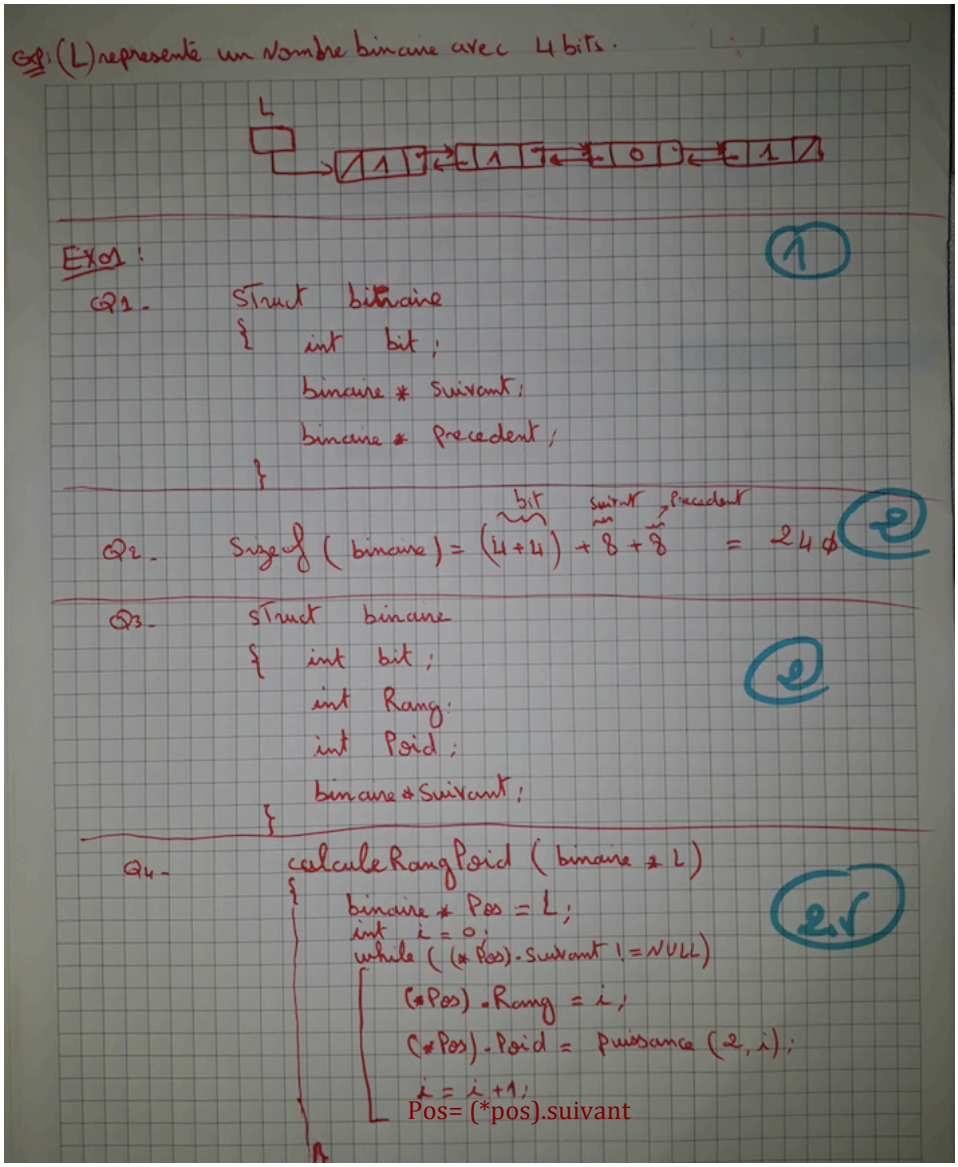

**NB :** *** Vous pouvez utiliser autant de piles temporaire que vous voulez.**
*** L'ordre des éléments n'est pas important.**

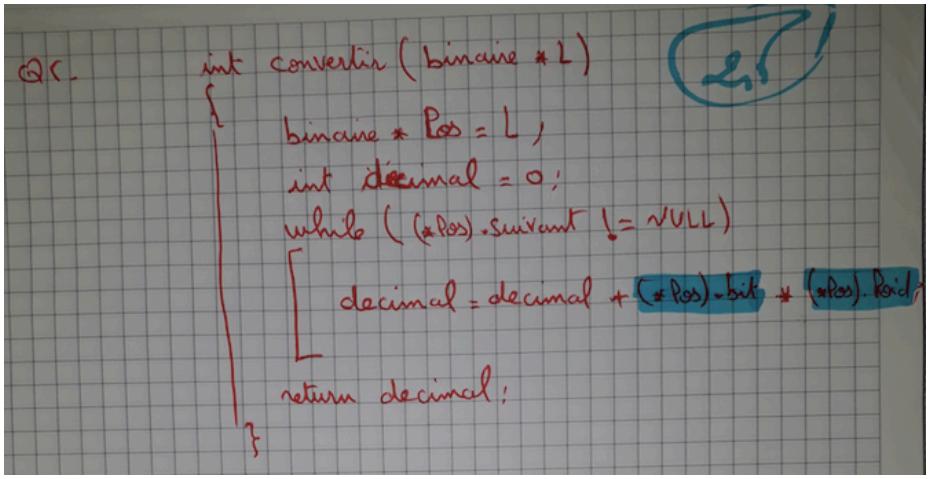
* Vous pouvez utiliser directement les fonctions vues dans le cours :

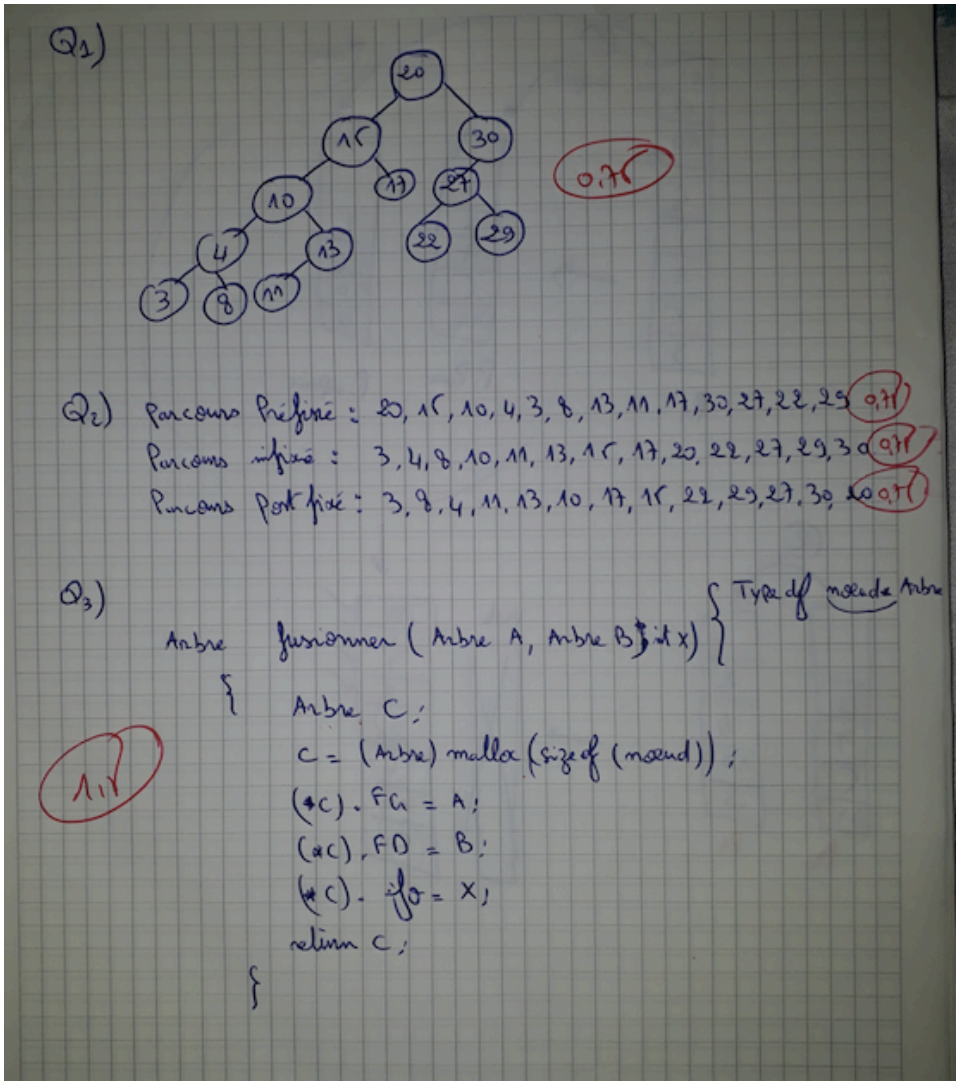
**Pile_vide (P1) =1** si la pile vide, 0 si la pile contient au moins un élément.
**X= Dépiler (P1).**
**Empiler (P1, X).**
**Etc...**

**Bon courage**

Exo1 :

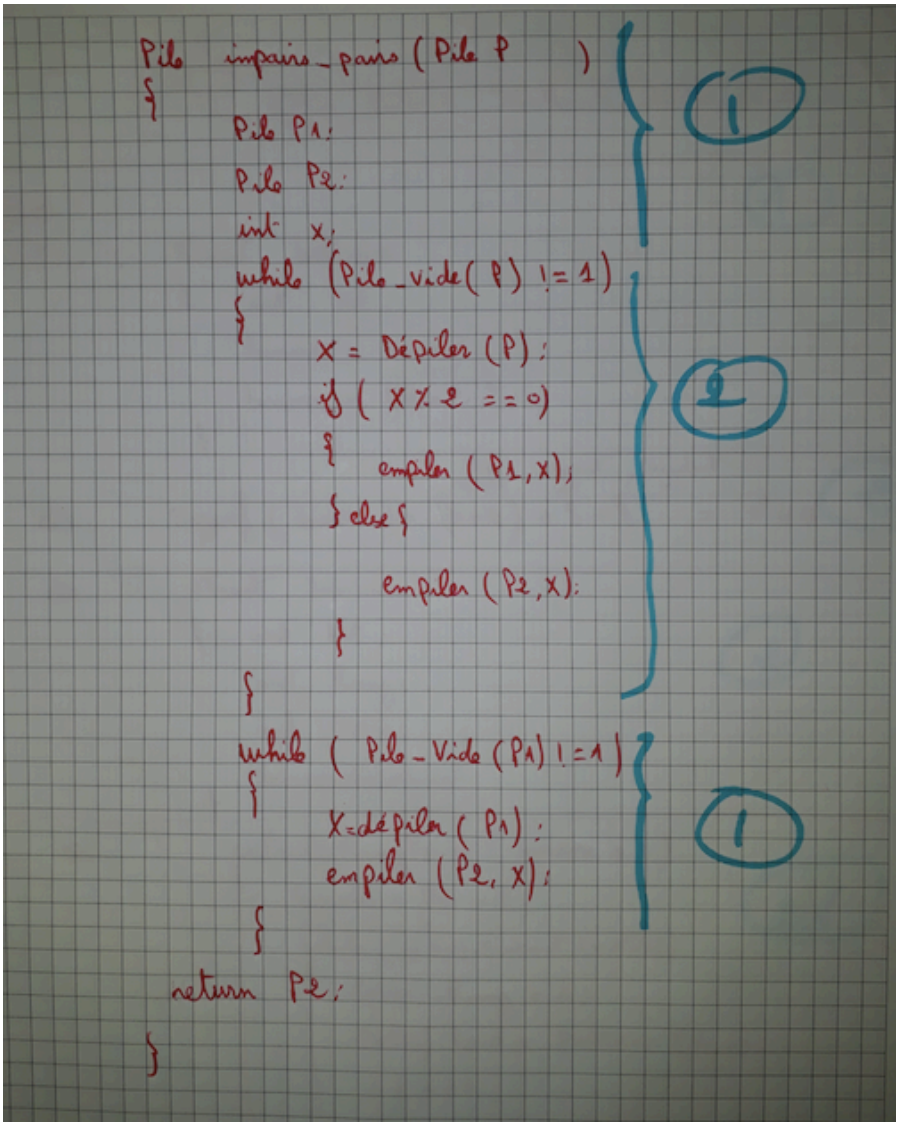

Q1: (L) représente un nombre binaire avec 4 bits.
L
1
1
0
1
Exo1 :
1
Q1-
struct binaire
{
int bit;
binaire * suivant;
binaire * precedent;
}
Q2-
bit
suivant
precedent
Sizeof ( binaire ) = (4+4) + 8 + 8 = 24 o
2
Q3-
struct binaire
{
int bit;
int Rang;
int Poid;
binaire * Suivant;
}
2
Q4-
calculeRangPoid ( binaire * L)
{
binaire * Pos = L;
int i = 0;
while ( (*Pos).Suivant != NULL)
{
(*Pos).Rang = i;
(*Pos).Poid = puissance (2, i);
i = i + 1;
Pos= (*pos).suivant
2,5


Q5-
int convertir ( binaire * L)
{
binaire * Pos = L;
int decimal = 0;
while ( (*Pos).suivant != NULL)
{
decimal = decimal + (*Pos).bit * (*Pos).Poid;
}
return decimal;
}
2,5




```
nœud *  recherche ( int x, nœud* R) {

                while (   (R != NULL)  &&  (x !=  (*R).info )   ) {

                        if ( x  <  (*R).info  ){
                                R = (*R).FG ;

                        }else{
                                R= (*R).FD ;
                        }
                }

                return R ;
        }
```

1,5



```
Pile impairs_pairs (Pile P )
{
    Pile P1;
    Pile P2;
    int x;
    while (Pile_vide( P) != 1)
    {
        x = Dépiler (P);
        if ( x % 2 == 0)
        {
            empiler ( P1, x);
        } else {
            empiler ( P2, x);
        }
    }
    while ( Pile_Vide (P1) != 1 )
    {
        X=dépiler ( P1);
        empiler (P2, X);
    }
return P2;
}
```

1

2

1