1 Arbre et pointeur

On représente un arbre grâce à la structure de données suivante :

```
struct sommet {
    int nb_fils;
    char lettre;
    sommet* premier_fils;
    sommet* frere_suivant;
};
```

et le pointeur vers sa racine :

```
sommet* racine;
```

Soit l'arbre défini par sept sommets "A","B",...,"G" tels que :

"A" a trois fils : "B","C","D" (dans cet ordre)

"B" a deux fils : "E","F" (dans cet ordre)

"C" a un fils : "G"

les sommets "D","E","F" et "G" n'ont pas de fils.

Par exemple, on remplit les champs de "B" et "C" par :

```
B.nb_fils = 2;
B.lettre = 'B';
B.premier_fils = &E;
B.frere_suivant = &C;

C.nb_fils = 1;
C.lettre = 'C';
C.premier_fils = &G;
C.frere_suivant = null;
```

On souhaite modifier l'arbre afin que le sommet "F" devienne sa nouvelle racine :

- en intervertissant des relations "père-fils", c.a.d. en changeant les pointeurs nécessaires pour que de 2 sommets qui étaient père et fils l'un de l'autre échangent leurs roles.

1.1 Combien de relations "père-fils" doit-on intervertir ?

2 car F est à 2 niveaux de profondeur.

1.2 Quel sera alors le nouveau frère de "E" ?
A ou B selon qui on permute en premier

1.3 Si maintenant on souhaite modifier directement les pointeurs des sommets nécessaires pour arriver au même résultat, quel est le nombre minimal de sommets à modifier ?
4 cad
```
Racine = &F;
A.premier_fils = null;
F.premier_fils = &B;
E.frere_suivant = &A;
```
Mais si on choisissait d'intervertir dans l'autre ordre il en faut 6!

1.4 Que fait le code suivant dans l'arbre initial ?
Il intervertit B et F mais C et D bougent pour rester frères de B au lieu de rester fils de A
En effet on a pas change B.frere_suivant

1.5 Combien faut-il modifier de pointeur(s) au minimum pour transformer l'arbre initial en arbre binaire sans perdre de noeuds
2 par exemple C.frere_suivant=null; et C->premier_fils = &D;

1.6 Combien de pointeur(s) faut-il au minimum modifier pour ajouter "H" en tant que feuille de l'arbre initial :
1 encore plus simple E.premier_fils = &H;

2 Liste chainée

On considère une liste chainée, avec des noeuds sous la forme :

```
struct Node {
    int val;
    struct Node* next;
};
```

On cherche à faire une fonction qui retourne la taille de la liste :

```
int taille(struct Node* head){
    int count = 0;
    struct Node* current = head;

    //TODO 1
        count ++;
        current = current->next;
    }

    return count;
}
```

2.1 Par quoi doit-on remplacer TODO 1 :

While (current!=null)

Voici maintenant une fonction mystère qui utilise la structure Node:

```
Node* fonction_mystere(Node * head, int v){
  Node *temp = malloc(sizeof(Node));
  temp->val = v;
  temp->next = head;
  return temp;
}
```

qui est appelée depuis la fonction main comme ceci :

```
void main(...){
  struct Node* head = head;
  head = fonction_mystere(head, 2);
  head = fonction_mystere(head, 3);
  head = fonction_mystere(head, 5);
  head = fonction_mystere(head, 7);
  head = fonction_mystere(head, 11);
  head = fonction_mystere(head, 13);
}
```

2.2 Que fait la fonction "fonction_mystere" :

Elle rajoute un élement au début de la liste

2.3 On souhaite maintenant trier la liste chainée à l'aide du tri fusion, quelle est sa complexité moyenne ?

O(n.log(n))

2.4 Quel tri a une Complexité moyenne égale ou inférieure au tri fusion ?

Tri rapide

3 Questions vues en cours

3.1 Que signifie Node *n1 = &n2; ?

que n1 et n2 pointent tous les deux vers la valeur de n2

3.2 Quelle conséquence est vraie pour la suite du code ? (s'il y a plusieurs bonnes rép choisissez n'importe laquelle)

toute modification de la valeur au bout de n2 changera la valeur au bout de n1 (et vice versa)

3.3 Quelle est la différence entre recherche linéaire et recherche par interpolation linéaire
la recherche par interpolation linéaire va plus vite en général

3.4 Quelle est le degré de l'arbre ci-dessus

3 (le noeud avec le plus de fils en a 3)

3.5 Dans quel ordre sont parcourus les noeuds lors d'un parcours en profondeur postfixe
BKLHEMNIFOPJGCDA

Car c'est la seule reponse

qui commence par B