INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

PROJET INFORMATIQUE n°7

Roxane LEDUC
Manon ROSE

Encadrant : M. Kotowicz

Département : Génie Mathématiques

Année scolaire : 2021-2022

Table des matières

INTRODUCTION

Ce projet consiste en l’analyse des opérations sur les listes d’entiers, tels que la saisie d’éléments, l’intersection de deux listes ou encore le tri des éléments d’une liste. Ayant des difficultés en informatique, nous avons choisi ce sujet dans le but d’améliorer nos compétences, notamment en langage C, ce qui nous sera utile pour le reste de notre formation. Ce sujet recoupe nos enseignements de C et d’algorithmique, c’est pourquoi, nous le trouvions très enrichissant.

IMPLÉMENTATION

N’ayant pas encore travaillé sur les fonctions en C, nous avons décidé d’utiliser un switch afin de permettre à l’utilisateur de choisir le cas qu’il souhaite traiter : le programme se déroule avec des « case » (ici exemple case 1 et case 2).

```
        printf("Que voulez-vous faire? 1. Afficher , 2. Cardinal, 3. Tri, 4. Union, 5.
Intersection, 6. Difference, 7. Ajouter 8. Supprimer (indice) 9. Supprimer (valeur) 10.
Extraire une sous-liste 11. Tri par insertion \n");
        scanf("%d", &choix);

        switch(choix){

        case 1:
                for (i=0;i<T1;i++) printf("tab1[%2d] = %d \n",i, tab1[i]);
                break;

        case 2:
                for (i=0;i<T1;i++) c++;
                printf("Le cardinal de la liste est de : %d \n", c);
                break;
```

Si l’utilisateur entre autre chose que les entiers de 1 à 11, le programme renvoie au cas « default » et lui demande de recommencer :

```
default :
        printf("Recommencez");
        break;
```

Au début de notre programme, on demande à l’utilisateur de rentrer des valeurs pour la liste 1. Puis on lui indique les choix qu’il peut faire avec sa liste dans le programme. Ces dernières sont de tailles définies. Par exemple, nous avons choisi la liste 1 de taille 6 et la liste 2 de taille 4, pour avoir deux tailles différentes.

Nous allons maintenant faire état des fonctionnalités de notre programme un peu plus en détails :

Case 1 :

L’objectif est ici, d’afficher la liste des éléments rentrés plus tôt.

Exemple :

```
Quelle valeur pour la case 0 du tableau 1:
 7
Quelle valeur pour la case 1 du tableau 1:
 4
Quelle valeur pour la case 2 du tableau 1:
 5
Quelle valeur pour la case 3 du tableau 1:
 3
Quelle valeur pour la case 4 du tableau 1:
 2
Quelle valeur pour la case 5 du tableau 1:
 9
Que voulez-vous faire? 1. Afficher , 2. Cardinal, 3. Tri a bulles, 4. Union, 5.
Intersection, 6. Difference symetrique, 7. Ajouter 8. Supprimer (indice) 9. Supp
rimer (valeur) 10. Extraire une sous-liste 11. Tri par insertion
1
Affichage du tableau:
tab1[ 0] = 7
tab1[ 1] = 4
tab1[ 2] = 5
tab1[ 3] = 3
tab1[ 4] = 2
tab1[ 5] = 9
```

Case 2 :

On va renvoyer le cardinal de cette liste, c'est à dire le nombre d'éléments la composant. Nous savons que le compteur est inutile puisque nous connaissons au préalable la taille du tableau, mais nous souhaitions traiter un cas plus général dans le cas où cette taille nous aurait été inconnue.

Exemple :

```
Quelle valeur pour la case 0 du tableau 1:
 7
Quelle valeur pour la case 1 du tableau 1:
 0
Quelle valeur pour la case 2 du tableau 1:
 3
Quelle valeur pour la case 3 du tableau 1:
 4
Quelle valeur pour la case 4 du tableau 1:
 5
Quelle valeur pour la case 5 du tableau 1:
 5
Que voulez-vous faire? 1. Afficher , 2. Cardinal, 3. Tri a bulles, 4. Union, 5.
Intersection, 6. Difference symetrique, 7. Ajouter 8. Supprimer (indice) 9. Supp
rimer (valeur) 10. Extraire une sous-liste 11. Tri par insertion
2
Le cardinal de la liste est de : 6
```

Case 3 :

Cette partie permet d'afficher les éléments de la liste 1 triés dans l'**ordre croissant** grâce à un tri à bulles. L'objectif est de comparer répétitivement les éléments consécutifs d'un tableau ou d'une liste et de les permuter lorsqu'ils sont mal triés. Après un premier parcours, le plus grand élément étant à sa position définitive, il n'a plus à être traité. Pour autant, le reste du tableau est encore en désordre. Il faut donc le parcourir à nouveau, en s'arrêtant à l'avant-dernier élément. Après ce deuxième parcours, les deux plus grands éléments sont à leur position définitive. Il faut donc répéter les parcours du tableau, jusqu'à ce que les deux plus petits éléments soient placés à leur position définitive. La complexité de cet algorithme est en $O(n^2)$.

Exemple :

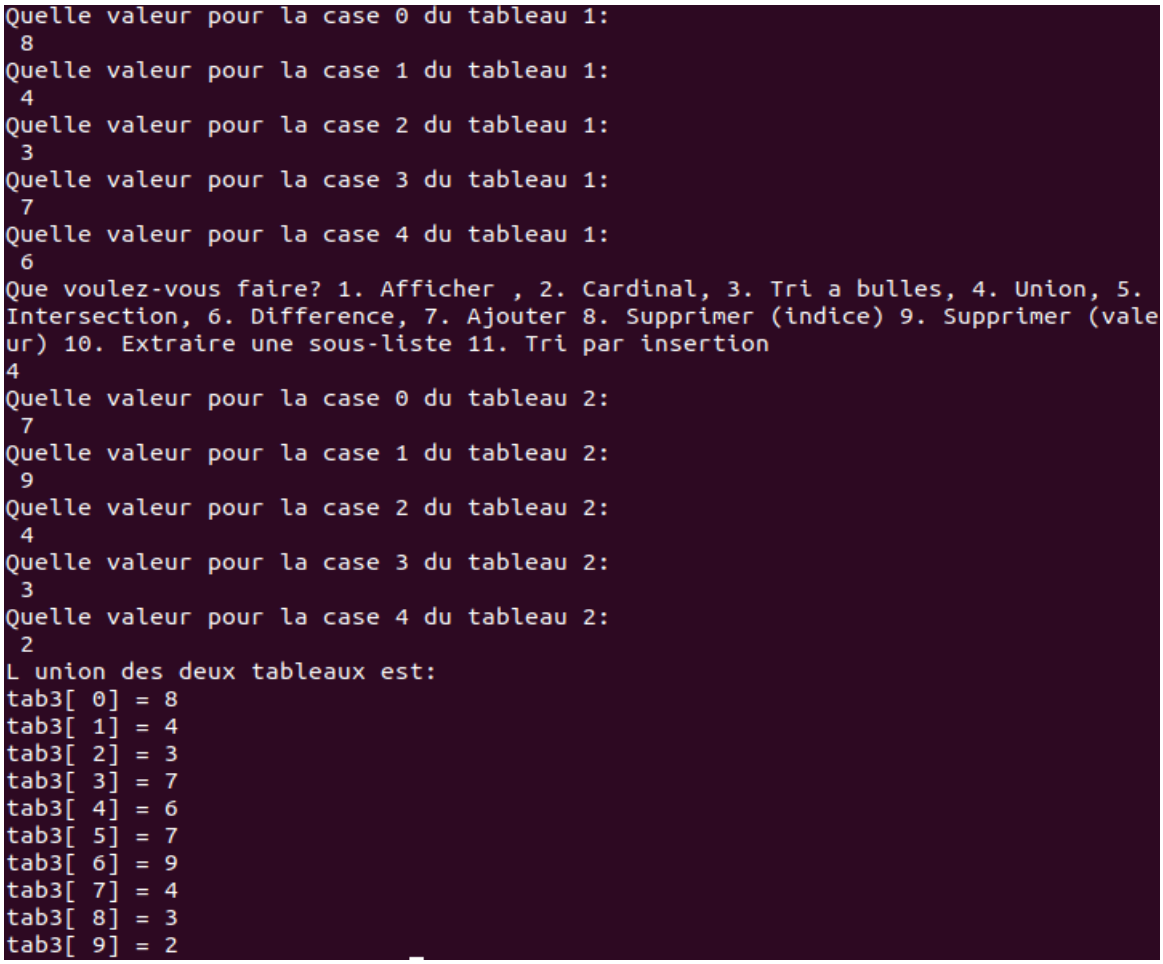
```
Quelle valeur pour la case 0 du tableau 1:
 7
Quelle valeur pour la case 1 du tableau 1:
 2
Quelle valeur pour la case 2 du tableau 1:
 3
Quelle valeur pour la case 3 du tableau 1:
 9
Quelle valeur pour la case 4 du tableau 1:
 5
Quelle valeur pour la case 5 du tableau 1:
 1
Que voulez-vous faire? 1. Afficher , 2. Cardinal, 3. Tri a bulles, 4. Union, 5.
Intersection, 6. Difference symetrique, 7. Ajouter 8. Supprimer (indice) 9. Supp
rimer (valeur) 10. Extraire une sous-liste 11. Tri par insertion
3
Le tableau trie est:
tab1[ 0] = 1
tab1[ 1] = 2
tab1[ 2] = 3
tab1[ 3] = 5
tab1[ 4] = 7
tab1[ 5] = 9
```

A partir du case 4, le programme demande à l'utilisateur de rentrer des valeurs pour la liste 2.

Case 4 :

Ici, sera affichée l'union de deux listes. Cette union consiste en une concaténation de deux listes distinctes ou non. Nous aurons donc la composée de tous les éléments des deux listes, même les éléments en double.

Exemple :



Case 5 :

Affichage de l’intersection de deux listes. L’intersection des deux listes consiste à renvoyer un tableau constitué des éléments que les deux listes ont en commun.

Exemple : Si elles possèdent toutes les deux un 8 et un 9, la liste finale ne contient qu’un des deux 8 et 9.

```
Quelle valeur pour la case 0 du tableau 1:
 7
Quelle valeur pour la case 1 du tableau 1:
 4
Quelle valeur pour la case 2 du tableau 1:
 8
Quelle valeur pour la case 3 du tableau 1:
 9
Quelle valeur pour la case 4 du tableau 1:
 2
Que voulez-vous faire? 1. Afficher , 2. Cardinal, 3. Tri a bulles, 4. Union, 5.
Intersection, 6. Difference, 7. Ajouter 8. Supprimer (indice) 9. Supprimer (vale
ur) 10. Extraire une sous-liste 11. Tri par insertion
5
Quelle valeur pour la case 0 du tableau 2:
 8
Quelle valeur pour la case 1 du tableau 2:
 3
Quelle valeur pour la case 2 du tableau 2:
 9
Quelle valeur pour la case 3 du tableau 2:
 5
Quelle valeur pour la case 4 du tableau 2:
 5
L intersection des deux tableaux est:
tab3[ 0] = 8
tab3[ 1] = 9
```

Case 6 :

Affiche la différence de deux listes. Elle consiste à renvoyer un tableau composé des éléments que les deux listes n’ont PAS en commun.

Exemple :

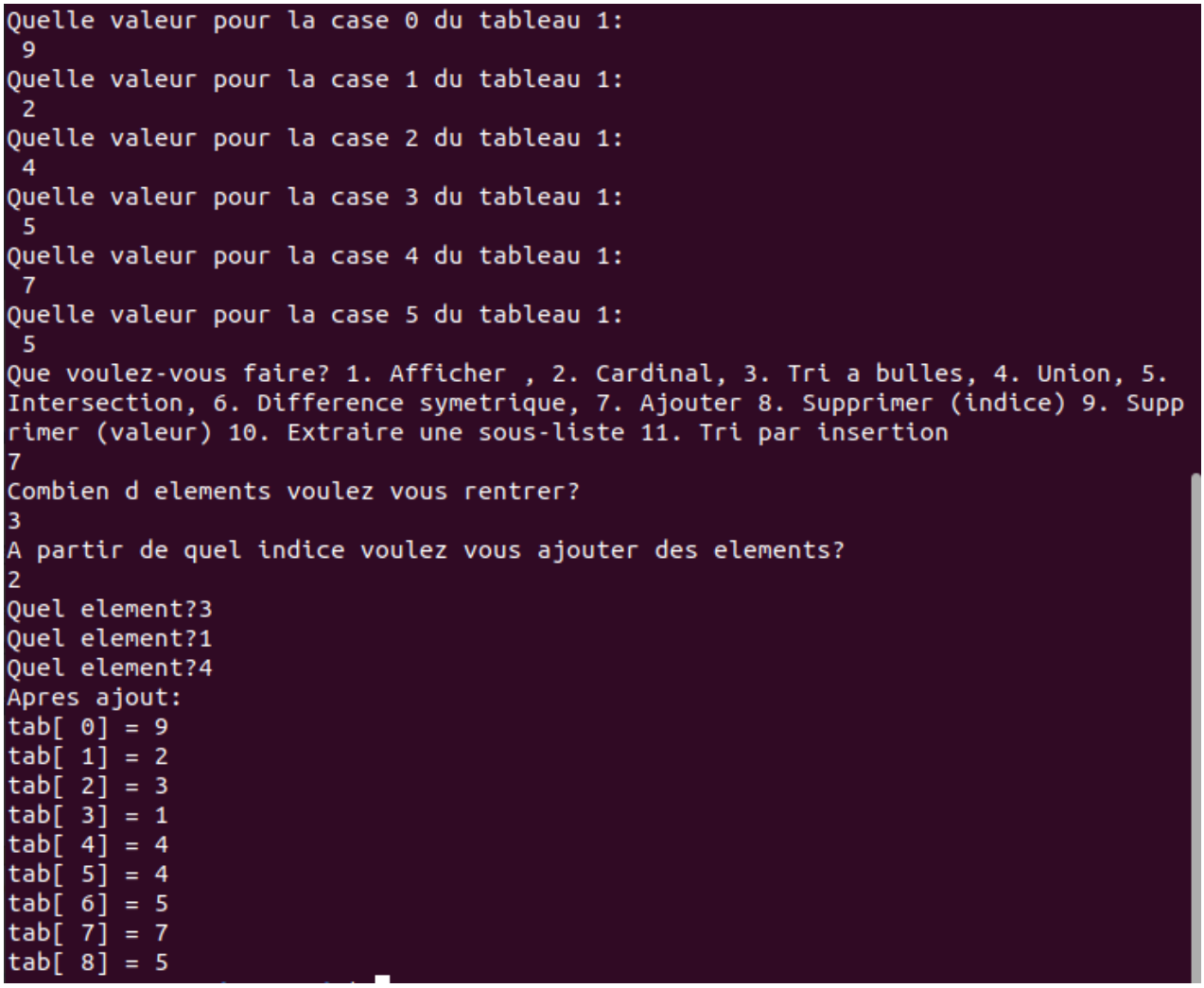
```
Quelle valeur pour la case 0 du tableau 1:
 1
Quelle valeur pour la case 1 du tableau 1:
 2
Quelle valeur pour la case 2 du tableau 1:
 3
Quelle valeur pour la case 3 du tableau 1:
 4
Quelle valeur pour la case 4 du tableau 1:
 4
Quelle valeur pour la case 5 du tableau 1:
 5
Que voulez-vous faire? 1. Afficher, 2. Cardinal, 3. Tri a bulles, 4. Union, 5. I
ntersection, 6. Difference symetrique, 7. Ajouter 8. Supprimer (indice) 9. Suppr
imer (valeur) 10. Extraire une sous-liste 11. Tri par insertion
6
Quelle valeur pour la case 0 du tableau 2:
 5
Quelle valeur pour la case 1 du tableau 2:
 4
Quelle valeur pour la case 2 du tableau 2:
 4
Quelle valeur pour la case 3 du tableau 2:
 9
La difference des deux tableaux est:
tab[ 0] = 1
tab[ 1] = 2
tab[ 2] = 3
tab[ 3] = 9
```

A partir du case 7, nous n’avons plus besoin d’une deuxième liste.

Case 7 :

Cette option permet d’ajouter des éléments à partir de l’indice choisi par l’utilisateur. Le programme nous renvoie un tableau composé des éléments de la liste 1 avec les éléments choisis insérés au bon endroit.

Exemple : l’utilisateur veut ajouter 3 éléments à partir de l’indice 2. Notons qu’en C, l’indice 2 correspond au 3^{e} indice, car ils commencent à 0.



Case 8 :

Nous allons pouvoir supprimer des éléments à partir de l’indice choisi par l’utilisateur. Le programme nous renvoie un tableau composé des éléments de la liste 1 sans les éléments choisis.

Exemple : l’utilisateur veut supprimer deux éléments à partir de l’indice 1.

```
Quelle valeur pour la case 0 du tableau 1:
 3
Quelle valeur pour la case 1 du tableau 1:
 5
Quelle valeur pour la case 2 du tableau 1:
 7
Quelle valeur pour la case 3 du tableau 1:
 8
Quelle valeur pour la case 4 du tableau 1:
 0
Quelle valeur pour la case 5 du tableau 1:
 2
Que voulez-vous faire? 1. Afficher , 2. Cardinal, 3. Tri a bulles, 4. Union, 5.
Intersection, 6. Difference symetrique, 7. Ajouter 8. Supprimer (indice) 9. Supp
rimer (valeur) 10. Extraire une sous-liste 11. Tri par insertion
8
Combien d element(s) voulez vous supprimer?
2
A partir de quel indice?
1
Apres suppression:
tab[ 0] = 3
tab[ 1] = 8
tab[ 2] = 0
tab[ 3] = 2
```

Case 9 :

Cette fois-ci, nous pouvons supprimer des éléments en fonction de leur valeur choisie par l'utilisateur. Le programme nous renvoie un tableau composé des éléments de la liste 1 sans les éléments de la valeur choisie.

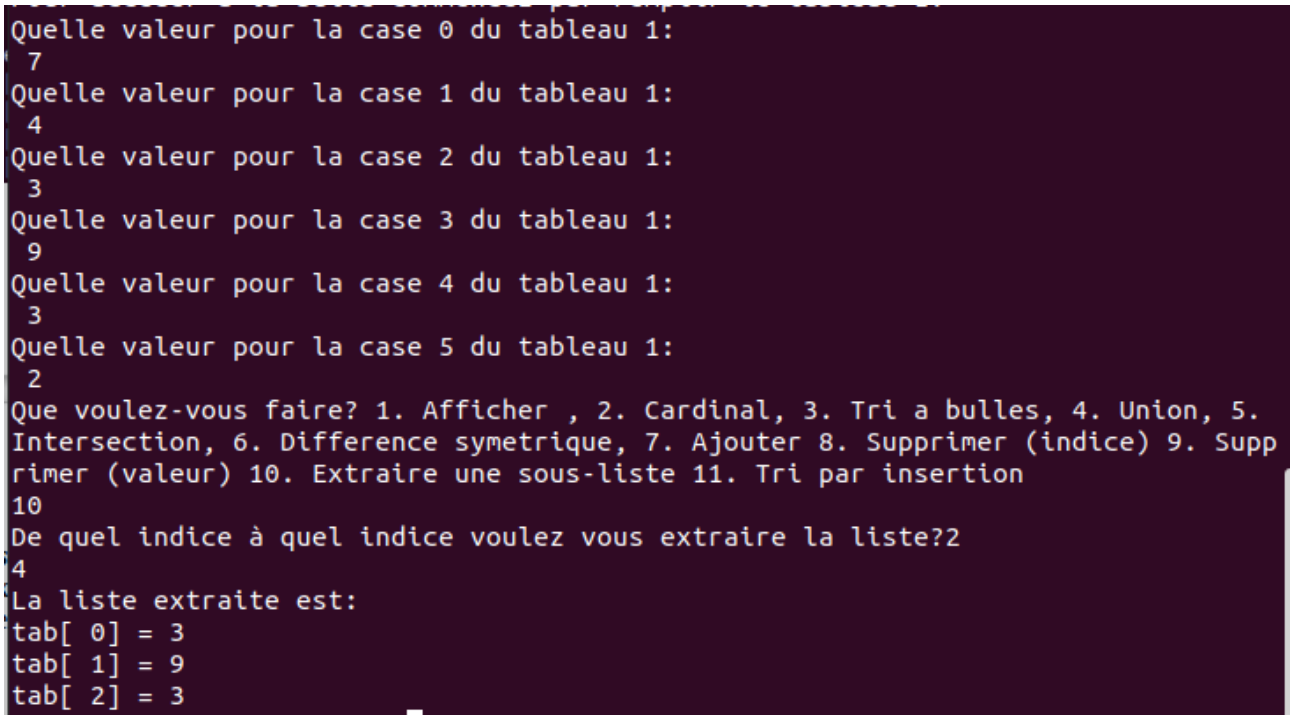
Exemple : l'utilisateur veut supprimer tous les 4 de la liste 1.

```
Quelle valeur pour la case 0 du tableau 1:
 5
Quelle valeur pour la case 1 du tableau 1:
 8
Quelle valeur pour la case 2 du tableau 1:
 4
Quelle valeur pour la case 3 du tableau 1:
 6
Quelle valeur pour la case 4 du tableau 1:
 4
Quelle valeur pour la case 5 du tableau 1:
 3
Que voulez-vous faire? 1. Afficher , 2. Cardinal, 3. Tri a bulles, 4. Union, 5.
Intersection, 6. Difference symetrique, 7. Ajouter 8. Supprimer (indice) 9. Supp
rimer (valeur) 10. Extraire une sous-liste 11. Tri par insertion
9
Quel(s) element(s) voulez vous supprimer?
4
Apres suppression:
tab[ 0] = 5
tab[ 1] = 8
tab[ 2] = 6
tab[ 3] = 3
```

Case 10 :

Permet d’extraire une sous liste. Le programme demande à l’utilisateur la partie de la liste à extraire (à partir des indices) et il nous renvoie la liste extraite.

Exemple : l’utilisateur veut extraire une sous liste de l’indice 2 à l’indice 4.

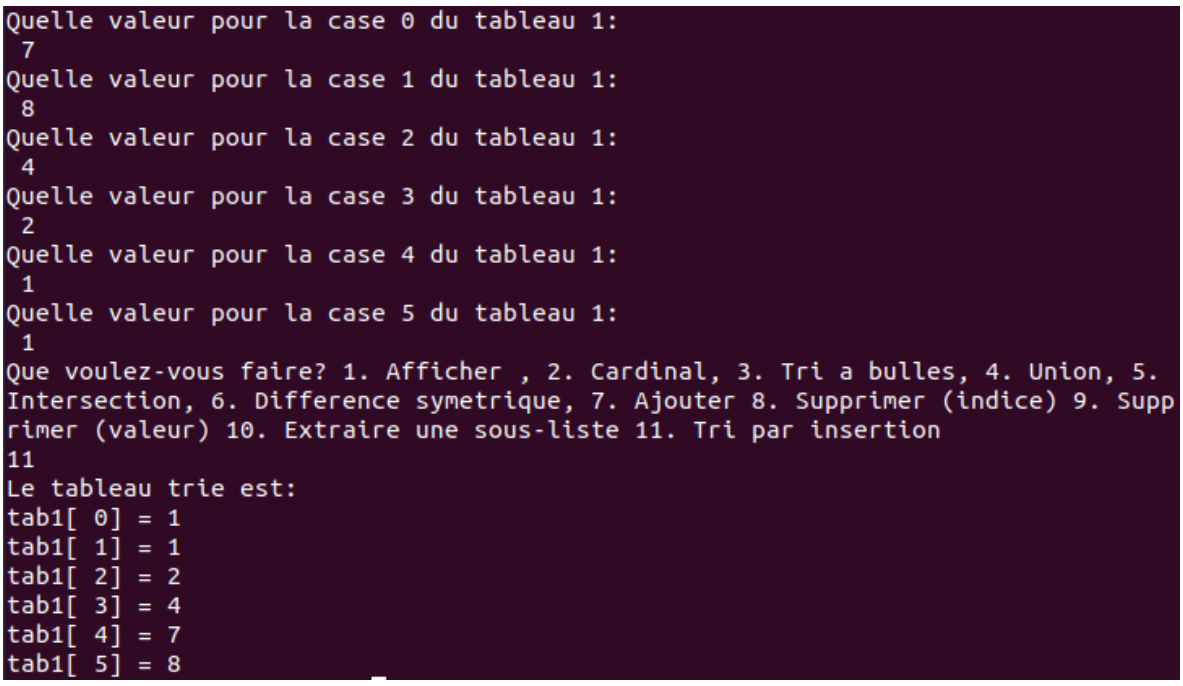


BONUS

Case 11 :

Permet d'afficher les éléments de la liste 1 triés dans l'**ordre croissant** grâce à un tri par insertion. Il considère chaque élément du tableau et l'insère à la bonne place parmi les éléments déjà triés jusqu'à ce qu'ils soient tous triés. Puisque c'est un tri, il renvoie le même tableau que pour le tri à bulles. En général, le tri par insertion est beaucoup plus lent que d'autres algorithmes comme le tri rapide (ou quicksort) et le tri fusion (ou mergesort) pour traiter de grandes séquences, car sa complexité est en $O(n^2)$. Le tri par insertion est cependant considéré comme l'algorithme le plus efficace sur des entrées de petite taille. Comme nous agissons ici principalement sur des listes courtes, ce tri nous a paru intéressant.

Exemple :



Case 0 :

Permet de lire les éléments d'une liste à partir d'un fichier et de les afficher. Au tout début du programme, on demande à l'utilisateur si il veut lire une liste depuis un fichier. Pour lire et écrire dans des fichiers, nous avons pu observer que nous devions apprendre à manipuler les pointeurs, choses que nous n'avons pas encore vu en cours. Nous nous sommes donc un petit peu renseignées sur la manière de manipuler des fichiers et voici ce que nous avons pu retenir : nous devons commencer par appeler la fonction d'ouverture de fichier fopen, qui nous renvoie un pointeur sur le fichier. Puis, on vérifie si l'ouverture a réussi en testant la valeur du pointeur qu'on a reçu. Si c'est le cas (si le pointeur est différent de NULL), alors on peut lire et écrire dans le fichier. Une fois qu'on a terminé de travailler sur le fichier, il faut penser à le « fermer » avec la fonction fclose.

Exemple :

```
Si vous voulez lire un tableau depuis un fichier entrez 0 sinon entrez n'importe
 quel chiffre.
0
La liste stockée dans le fichier est :
1 2 3 5 6
```

CONCLUSION

Ce projet nous a permis de faire nos premiers véritables pas dans le monde de la programmation algorithmique. Nous avons pu nous avancer un peu sur les TDs et découvrir par nous mêmes à nous servir de tableaux et nous avons commencé à manipuler les fichiers. Nous n'en sommes encore qu'au début de notre apprentissage du langage C, mais ce projet nous a permis d'être un peu plus à l'aise.

PS :

Nous n'avons vu que récemment comment utiliser des fonctions, c'est pourquoi, nous n'avons codé qu'une partie du projet à l'aide de ces dernières (nous avons bien évidemment repris le code existant mais nous l'avons simplement implémenté différemment), principalement afin de nous entraîner à manipuler ces structures.