L2 Programmation Impérative
TP2

1 Fichier Makefile

1. Récupérez les fichiers `makefile`, `tp2ex1.c`, `lire.c` et `ecrire.c` à l'adresse `http://www.lacl.fr/~matran/progimp/tp2/` Notez que le fichier `makefile` n'a pas d'extension.
   Utilisez ensuite la compilation séparée pour produire l'exécutable `tp2ex1` à partir de ces fichiers.
2. Comment modifier les fichiers `makefile` et `tp2ex1.c` si l'on veut remplacer dans `tp2ex1.c` l'instruction `printf(...)` par l'appel d'une fonction `ecrire_entier (x)` fournie dans le fichier `ecrire.c` ?
3. On suppose que l'on a exécuté avec succès la commande `make`, puis que l'on modifie `ecrire.c`. Quelles commandes de compilation seront effectuées si l'on exécute de nouveau le programme `make` ?

2 Liste doublement enchaînée

Construction d'un noeud dans une liste doublement enchaînée d'entiers :

```
struct noeud {
  int valeur;
  struct noeud *suivant;
  struct noeud *precedent;
} ;
```

Pour présenter une liste doublement enchaînée, on utilise une deuxième strcuture :

```
struct dliste {
  struct noeud *tete;
  struct noeud *queue;
} ;
```

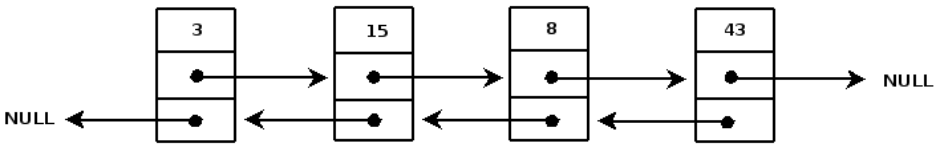

FIGURE 1 – Exemple d'une liste de 4 entiers

On se propose de créer une librairie de gestion de listes doublement enchaînées.

1. Écrire un fichier `liste.h` décrivant l'interface de manipulation de listes doublement enchaînées. Les méthodes utilisables devront être :
   – `initialiser()`, initialisant les champs d'une structure représentant une liste vide,
   – `liste_vide()`, retournant `vrai` si une liste est vide et `faux` sinon,
   – `est_present()`, retournant un pointeur sur un noeud contenant un entier recherché s'il est présent dans la liste,

- `ajouter_en_tete()`, ajoutant un entier en tête de la liste s'il n'y est pas présent,
- `cardinal()`, retournant le nombre d'éléments dans la liste,
- `afficher_liste_tete()`, affichant le contenu de la liste en commençant par la tête.
- `afficher_liste_queue()`, affichant le contenu de la liste en commençant par la queue.

2. Écrire un fichier `liste.c` implémentant les fonctionnalités ci-dessus.
3. Compiler `liste.c` en un fichier objet et écrire un fichier `essai_liste.c` pour tester les fonctions sur les listes. Créer un fichier `makefile` pour effectuer la création de l'exécutable `essai_liste`.