Université de Tunis Institut Supérieur de Gestion Département Informatique	
Projet Algorithmique et Structures de Données	
Niveau:	1ère année (fondamentale/appliquée) Informatique de Gestion (semestre 2)
Année Universitaire:	2017/2018

ENVIRONNEMENT :
Le projet doit être réalisé avec le langage de programmation C.

CONTENU DU RAPPORT A RENDRE :
Fiche signalétique du binôme
Listing des différents fichiers sources
CD sources
Il sera tenu compte de la présentation du projet
Une interface graphique n'est pas obligatoire mais pas interdite ☺

REMARQUE IMPORTANTE
Le projet doit se faire par binôme du **MEME GROUPE de TP**

PARTIE 1 : *Problème de Bin Packing*

Il s'agit de trouver le rangement le plus économique possible pour un ensemble d'objets dans des boîtes à capacité fixe (appelées bins).

Principe : Étant donné un ensemble d'objets dont on connaît les tailles, connaissant également la capacité d'un bin, répartir les objets dans des bins tout en minimisant le nombre total de bins.

Exemple: Placer les objets de taille : **8, 7, 14, 9, 6, 9, 5, 15, 6, 7, 8** dans des bins de taille **20**

1) Supposons qu'on va travailler sur 30 objets (dont la taille est ≤ 20) stockée dans un tableau **Lobj** tel que **Lobj**[i] est la taille de l'objet i. Les Bins sont stockés dans une liste **Lbins** tel que chaque élément dans cette liste pointe sur un bin représenté par un tableau **Tbin** de taille maximale égale au nombre d'objets (dans notre cas 30) et par une capacité **Cbin** initialisée au départ à une constante donnée (dans notre cas 20) et par le nombre d'objet qu'il contient **Nbobj**.

Donner le code nécessaire à la création de la structure d'un bin (le fichier ELTBIN.H)

2) Pour résoudre ce problème il existe plusieurs méthodes dont l'Algorithme : **First_Fit** qui procède comme suit :

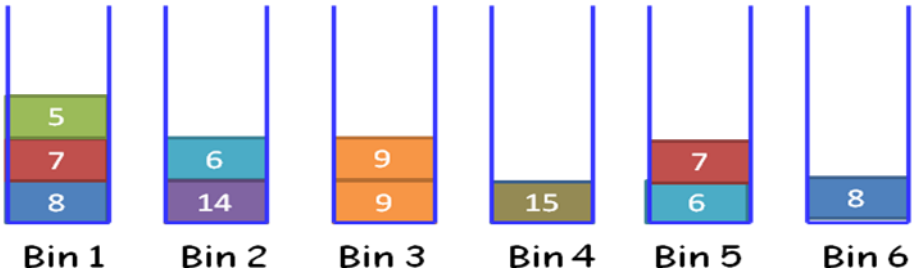
a) Prendre les objets dans l'ordre donné;
b) Placer chaque objet dans le premier bin disponible (on commence à partir du Bin 1 à chaque fois)

La trace d'exécution de cet algorithme sur l'exemple précédant est :

- On place objet 1 dans bin 1 (8): capacité restante (Cbin)=20-8= 12
- On place objet 2 dans bin 1 (7) : capacité restante (Cbin)=12-7= 5
- On place objet 3 dans bin 2 (14): capacité restante (Cbin)=20-14= 6
- On place objet 4 dans bin 3 (9): capacité restante (Cbin)=20-9= 11
- On place objet 5 dans bin 2 (6): capacité restante (Cbin)=6-6= 0 : saturé
- On place objet 6 dans bin 3 (9): capacité restante (Cbin)=11-9= 2
- On place objet 7 dans bin 1 (5): capacité restante (Cbin)=5-5= 0 : saturé

- On place objet 8 dans bin 4 (15): capacité restante (Cbin)=20-15= 5
- On place objet 9 dans bin 5 (6): capacité restante (Cbin)=20-6= 14
- On place objet 10 dans bin 5 (7): capacité restante (Cbin)=14-7= 7
- On place objet 11 dans bin 6 (8): capacité restante (Cbin)=20-8= 12

Ce qui donne 6 bins

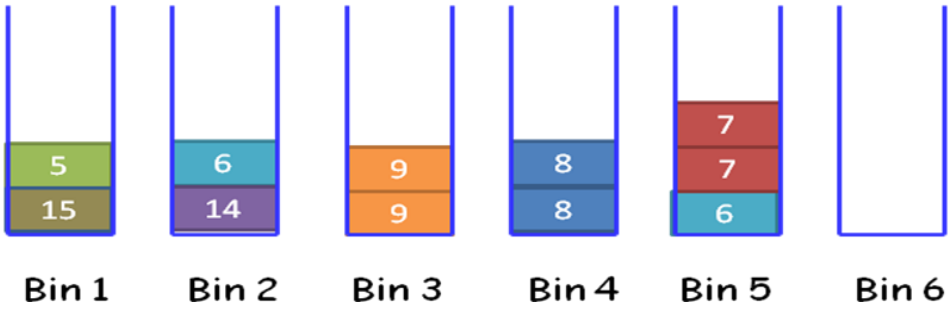


Implémenter First_Fit en vous limitant aux primitives de **LSTPRIM.H** et **ELTPRIM.H** (le choix de l'implémentation des listes est libre). Le programme doit afficher à la fin le nombre de bins et le contenu de chacun d'eux.

3) Pour résoudre le problème du *Bin Packing* , on peut également citer l'algorithme **First_Fit_Decreasing** dont le principe est :

a) Trier les éléments dans l'ordre décroissant.
b) Appliquez le premier algorithme à la liste.

Le résultat de cet algorithme sur l'exemple précédant donne les 5 bins suivants :

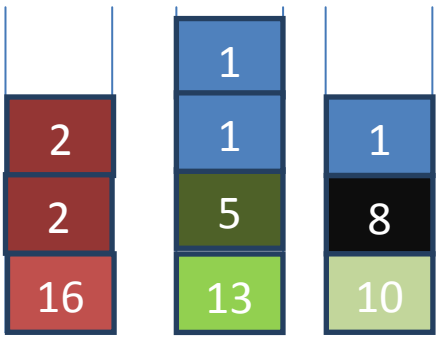


Implémenter l'algorithme **First_Fit_Decreasing** (en précisant la méthode de tri sélectionnée dans votre rapport).

4. Un autre algorithme un peu plus optimisé que le précédent, appelé *Best-fit Decreasing*, consiste à Trier les éléments dans l'ordre décroissant. Ensuite on range l'objet dans la boîte la mieux remplie qui puisse le contenir. Le résultat de cet algorithme sur l'exemple précédant donne les 5 bins suivants :

Liste Triée d'objets 16 13 10 8 5 2 2 1 1 1

Résultat selon First-Fit-Decreasing



Résultat selon Best-Fit-Decreasing