Université de Valenciennes et du Hainaut-Cambrésis
Institut des Sciences et Techniques de Valenciennes **2015-2016**

Master TNSI Semestre 7
TP de Bases de l'Optimisation Combinatoire

Consignes :

- Vos programmes doivent être écrits en C ou C++.
- Vos programmes doivent pouvoir être compilés et exécutés sous Windows.
- A la fin de chaque séance de TP vous devez déposer une **archive** sur Moodle contenant l'ensemble des fichiers nécessaires à la construction de l'exécutable (**et uniquement ceux-ci**). Le cours Moodle s'intitule `Introduction à l'optimisation combinatoire (M1 Informatique)`. L'archive contiendra :
  - l'ensemble de vos fichiers **sources** (.h, .c, .cpp)
  - un ReadMe.
- L'archive ne doit contenir **en aucun cas** de fichiers avec les extensions .o, .exe, ou assimilées.
- Les fichiers codes auront tous un en-tête en commentaire indiquant vos nom(s) et prénom(s) (important, en particulier si vous travaillez en binôme).
- N'oubliez pas de commenter le code, en donnant au minimum des explications sur les structures de données utilisées et les principaux sous-programmes développés.

Évaluation :

Votre travail sera évalué sur la base de trois notes :

- une note sur **10** portant sur le travail réalisé pendant la première séance et l'état d'avancement à la fin de cette séance.
- une note sur **15** portant sur le travail réalisé pendant la deuxième séance et l'état d'avancement à la fin de cette séance.
- une note sur **5** portant sur la version finale que vous serez autorisé à déposer sur Moodle après la fin des TP.

Problème :

Dans ce TP vous allez implémenter quelques algorithmes permettant de traiter le problème du sac-à-dos en variables 0-1 (SAD). Dans ce problème on a un ensemble de n objets. Chaque objet j est caractérisé par un profit c_j et un poids a_j, et le sac a une capacité maximale notée b. On rappelle la formulation mathématique de ce problème :

$$(0-1 \text{ SAD}) \begin{cases} \max & \sum_{j=1}^{n} c_j x_j \\ \text{s.c :} & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \in \{0,1\} \quad j = 1, \ldots, n \end{cases}$$

Exercice 1 : Structures de données

Proposer des structures de données pour représenter et manipuler facilement une instance et une solution du 0-1 SAD. L'instance sur laquelle on appliquera les algorithmes sera lue dans un fichier **texte**, respectant **scrupuleusement** le format suivant :

n b
c_1 c_2 $\ldots$ c_n
a_1 a_2 $\ldots$ a_n

Exercice 2 : Relaxation en continu et solution réalisable

Implémenter l'heuristique gloutonne permettant de résoudre la relaxation en continu d'une instance donnée de 0-1 SAD à l'optimum, et d'en récupérer la valeur et une solution optimale.

Dériver de cet algorithme une heuristique permettant d'obtenir une solution réalisable du 0-1 SAD.

Exercice 3 : Programmation dynamique

Proposer un algorithme de programmation dynamique pour résoudre une instance donnée du 0-1 SAD.

Exercice 4 : Branch-and-bound

Implémenter un algorithme de type évaluations et séparations progressives pour résoudre une instance du 0-1 SAD. Pour cela, vous pouvez par exemple utiliser les algorithmes de l'exercice 2 pour calculer des bornes inférieure et supérieure à chaque nœud de l'arbre de recherche.