TP 4 : fonctions et procédures

I. Le maximum

Nous souhaitons réaliser un programme qui calcule le maximum de deux réels.

1. Écrivez une classe qui contient une fonction `maximum` et un `main`. La fonction prend en paramètre deux réels et retourne le maximum des deux. Dans le `main`, le programme demande deux réels à l'utilisateur puis en utilisant la fonction `maximum`, il affiche le maximum des deux nombres.

Vous ne devez pas utiliser la fonction `max` située dans la classe `Math` (cf TP précédent). Vous devez bien écrire votre propre algorithme.

2. Cette fois-ci, demandez plutôt à l'utilisateur de saisir un entier `n` puis `n` réels. Le programme doit ensuite afficher le maximum des `n` réels saisis calculé avec la fonction `maximum`.

Pour écrire le programme, vous ne devez pas modifier la fonction `maximum`, mais l'utiliser telle qu'elle est.

II. PGCD et factorielle

Le but de cet exercice est de reprendre quelques algorithmes vus en CMTD ou dans les TP précédents et d'en faire des fonctions. Pour chaque question, écrivez la ou les fonctions demandées, puis testez-les dans un `main` en demandant les valeurs nécessaires à l'utilisateur puis en affichant le résultat à l'écran.

1. Écrivez la fonction `factorielle`.
2. Écrivez la fonction `puissance` qui permet de calculer x^y.

Le calcul du PGCD des entiers *a* et *b* par la méthode d'Euclide consiste à calculer *r* qui est le reste de la division de *a* par *b*. On remplace ensuite *a* par *b* et *b* par *r*. On recommence l'opération jusqu'à ce que *r* soit égal à 0. Le PGCD est alors égal à *b*.

3. Écrivez la fonction `PGCD`.
4. Que se passe-t-il si la valeur saisie pour *b* est supérieure à *a* ?
5. Que se passe-t-il si les valeurs saisies sont négatives ?
6. Écrivez une fonction `estValide` qui prend en paramètre *a* et *b* et qui indique s'ils peuvent être utilisés pour la fonction `PGCD`. Si ce n'est pas le cas, l'utilisateur devra saisir de nouvelles valeurs (la saisie doit être réalisée dans le `main`).

III. Facturation

Une entreprise de vente par correspondance du Québec fait appel à vous pour réaliser un programme de facturation. Cette entreprise vend du sirop d'érable (en cruchon de 1L) extra-clair à 28€/L, par lot de 3 uniquement. Elle vend aussi des bouteilles de sirop de bleuets sauvages 100% pur du Lac-St-Jean à 12.00€ la bouteille, vendues à l'unité.

Le but de cet exercice est de construire une application à l'aide de plusieurs fonctions/procédures différentes. L'application finale ne pourra fonctionner qu'une fois toutes les fonctions/procédures écrites. Cependant, pour chaque question, écrivez la fonction/procédure demandée et testez-la dans le `main`.

1. Écrivez une fonction `saisirNbSiropErable` qui demande à l'utilisateur un nombre de cruchons de sirop d'érable et qui retourne la valeur saisie. Si la valeur n'est pas multiple de 3 ou est incorrecte (inférieure à 0, par exemple), un message d'erreur doit être affiché et l'utilisateur doit saisir une nouvelle valeur.
2. Ecrivez une fonction `saisirNbSiropBleuets` qui demande à l'utilisateur un nombre de bouteilles de sirop de bleuets et qui retourne la valeur saisie. Si la valeur est incorrecte, un message d'erreur est affiché et l'utilisateur doit saisir une nouvelle valeur.
3. Écrivez une fonction `calculerFraisTransport` qui retourne les frais de transport en fonction du nombre de cruchons et de bouteilles commandées. Pour les cruchons de sirop d'érable, le coût est de 1.5€ par cruchon si le client en commande de 3 à 11, 1€ de 12 à 24 et gratuit au-delà. Pour les bouteilles de sirop de bleuets, c'est 0.5€ par bouteille quel que soit le nombre commandé.
4. Écrivez une fonction/procédure `afficherFacture` qui prend en paramètre le nombre de cruchons et de bouteilles, les frais de transport et le total de la facture. La facture est affichée à l'écran.
5. Écrivez le programme principal (un `main`) qui utilise les différentes fonctions/procédures précédentes. Une fois la facture affichée, l'utilisateur du programme peut décider d'arrêter ou de saisir une nouvelle commande.

L'entreprise souhaite se diversifier. Maintenant, elle décide de vendre en plus des cruchons de sirop d'érable médium à 21.4€/L et clair à 23.5€/L. Le nombre de cruchons de chaque sorte doit être multiple de 3. Les frais de transport sont indépendants du type de sirop d'érable. La facture devra afficher le détail de chaque sorte.

6. Modifiez la fonction `saisirNbSiropErable` en lui ajoutant en paramètre le type de sirop (sous forme d'une chaîne de caractères, le type `String` en *Java*).
7. Modifiez votre programme pour prendre en compte les modifications.

IV. Le jeu du Pop-Tarts

Le jeu du *Pop-Tarts* est un jeu à deux joueurs et se joue à l'aide d'un tas de *Pop-Tarts* (sorte de pâtisserie au goût très chimique appréciée par les jeunes Québécois). Chaque joueur ramasse, tour à tour, 1, 2 ou 3 *Pop-Tarts* au maximum (et peut prendre le risque de les manger). Le joueur qui ramasse le dernier *Pop-Tarts* a perdu.

1. Écrivez une fonction `jeuOrdi(n : entier) : entier` qui détermine le nombre de *Pop-Tarts* ramassés par l'ordinateur, *n* étant le nombre de *Pop-Tarts* restant.
2. Écrivez une fonction `jeuJoueur(n : entier) : entier` qui demande au joueur le nombre de *Pop-Tarts* qu'il désire ramasser et retourne cette valeur, le paramètre *n* représentant le nombre de *Pop-Tarts* restants. Cette fonction devra vérifier que le nombre retourné est correct (entre 1 et min(3,n)).
3. Écrivez une fonction `jouer() : entier` gérant le déroulement d'une partie. On choisira à chaque partie un nombre aléatoire de *Pop-Tarts* entre 10 et 20 et on laissera le joueur commencer. La fonction retourne 1 si c'est le joueur qui a gagné, 0 sinon.
4. Écrivez le `main` qui permet de faire une partie de *Pop-Tarts*.
5. Modifiez le `main` pour qu'un menu soit affiché après la première partie. Le joueur peut rejouer, arrêter le programme ou afficher les statistiques des parties précédentes (nombre de parties jouées, nombre de parties gagnées, etc.)