Rattrapage ADA

2010 – 2011

Conditions d'examen :

- Tous les documents sont autorisés sauf Internet.
- Le devoir se fait sur machine et se rendra sur une clef USB fournie par les examinateurs.
- **Tout code qui ne compile pas ne sera pas corrigé et entrainera une note nulle. Si une partie de votre code ne compile pas, commentez-le. Ceci implique évidemment que vous compiliez au fur et à mesure!**
- La lisibilité du code et les commentaires seront pris en compte dans la notation.
- Pour vous simplifier le travail de décomposition, nous avons identifié les différentes fonctions/procédures que vous deviez coder. Vous devez au minimum coder ces fonctions/procédures, cependant si vous souhaiter en créer d'avantage pour clarifier le code vous le pouvez.
- La seule variable globale tolérée sera la variable contenant le générateur de nombres aléatoires, si vous mettez d'autres variables en visibilité globale, vous aurez des pénalités.
- **Chaque exercice sera traité dans un fichier source (.adb) différent**

Exercice 1 : Un joli Triangle (/8)

Le but de cet exercice est d'afficher un triangle isocèle sur le terminal, en ne se servant que des caractères '*' et ' ' (espace).
Chacune des lignes affichées est composée de **p** espaces suivis de **q** étoiles.

Voici un rendu de 2 exécutions de ce que l'on cherche à obtenir (pour **n** = 5 et **n** = 20) :

```
Entrez le nombre de lignes du triangle :
5
    *
   ***
  *****
 *******
*********
```

```
Entrez le nombre de lignes du triangle :
20
                    *
                   ***
                  *****
                 *******
                *********
               ***********
              *************
             ***************
            *****************
           *******************
          *********************
         ***********************
        *************************
       ***************************
      *****************************
     *******************************
    *********************************
   ***********************************
  *************************************
 ***************************************
*****************************************
```

Question 1 (/1): Créez les structures suivantes pour représenter le problème :

- **TabInt** : Pour stocker un tableau d'entiers
- **Triangle:** Pour stocker les informations sur le triangle : sa taille (nombre de lignes), un tableau nbEspaces contenant le nombre d'espaces à afficher sur chaque ligne, un tableau nbEtoiles contenant le nombre d'étoiles à afficher sur chaque ligne.

Question 2 (/2): Créez une **fonction Saisie** sans paramètre, demandant à l'utilisateur le nombre de lignes du triangle à afficher et renvoyant cette valeur entière. Vous devez gérer les cas d'**exception** pouvant survenir lors de la saisie, en affichant un message d'erreur à l'utilisateur et en lui redemandant de saisir une valeur.

Question 3 (/2): Créez une **procédure CalculerTriangle** prenant en paramètre un Triangle et modifiant cette variable en calculant et en affectant le contenu des champs nbEtoiles et nbEspaces.

Question 4 (/2): Créez une **procédure AfficherTriangle** prenant en paramètre un Triangle et permettant de l'afficher. Vous pourrez vous appuyer de la représentation graphique fournit sur la première page de cet énoncé.

Question 5 (/1): Créez la **procédure principale** permettant d'appeler les différentes fonctions et procédures écrites précédemment.

Exercice 2 : Des polynômes... (/8)

Dans cet exercice, nous supposons que l'utilisateur peut saisir un polynôme de degré n (en saisissant les différents coefficients de ce polynôme). Le but est de calculer la valeur numérique de ce polynôme pour une valeur X saisie ensuite par ce même utilisateur.
Voici un rendu de ce que l'on cherche à obtenir :

```
Entrez le degre de votre polynome :
3
Entrez la valeur du coefficient de degre 0 :
1.2
Entrez la valeur du coefficient de degre 1 :
2.3
Entrez la valeur du coefficient de degre 2 :
3.4
Entrez la valeur du coefficient de degre 3 :
4.5
P(X) = 4.50 X^3 + 3.40 X^2 + 2.30 X^1 + 1.20 X^0
Entrez le nombre reel :
1.3
Pour X = 1.30, le polynome P vaut 19.82
```

Question 1 (/0.5): Créez les structures suivantes pour représenter le problème :

- **Coefficients** : Pour stocker un tableau de coefficients réels.
- **Polynome :** Pour stocker le degré du polynôme ainsi que le tableau Coefficients.

Question 2 (/0.5): Créez une **fonction SaisieR** sans paramètre, demandant à l'utilisateur de saisir un nombre réel et le retournant. Vous devez gérer les cas d'**exception** pouvant survenir lors de la saisie, en affichant un message d'erreur à l'utilisateur et en lui redemandant de saisir une valeur.

Question 3 (/2): Créez une **fonction SaisieP** sans paramètre, demandant à l'utilisateur de saisir et de renvoyer un Polynome. Vous devez gérer les cas d'**exception** pouvant survenir lors de la saisie, en affichant un message d'erreur à l'utilisateur et en lui redemandant de saisir une valeur.

Question 4 (/2): Créez une **fonction CalculerValeur** prenant en paramètres un polynome P et un réel X et retournant la valeur de P(X) pour ce X donné.

Question 5 (/2): Créez une **procédure AfficherPolynome** prenant en paramètres un polynome et l'affichant à l'écran. Vous pourrez vous inspirez de la capture d'écran pour représenter le polynôme le plus clairement possible.

Question 6 (/1): Créez la **procédure principale** permettant d'appeler les différentes fonctions et procédures écrites précédemment.

Note : La puissance en ADA est définie par **
Ex : X^2 s'écrit : X ** 2

Exercice 3 : ...persistants ! (/4)

Dans cet exercice, nous reprenons l'exercice précédent sur les polynômes. Cette fois-ci, l'utilisateur doit gérer une liste de polynômes. Créez un nouveau fichier .adb et ne reprenez que les fonctions/procédures utiles à la résolution de cet exercice.

A chaque lancement de l'application, celle-ci doit charger la liste de polynômes déjà créés (à partir d'un fichier binaire nommé *poly.sav*) si elle existe et les afficher. Puis elle demande à l'utilisateur de saisir un nouveau polynôme, l'ajoute à la liste, et sauvegarde la nouvelle liste de polynômes avant de quitter.

Vous êtes libres de créer les fonctions/procédures que vous voulez pour cette question.