1 Programmation de la fonction sinus

1. Implantez la fonction `terme` vue en TD permettant de calculer un terme de la série $S_n(x) = \sum_{n>0}(-1)^{(n-1)}.\frac{x^{(2n-1)}}{(2n-1)!}$. Rappel: le type de la fonction est `float-> int -> float`.

2. Implantez la fonction `serie_sinus` calculant $S_n(x)$. Là encore le type est `float-> int -> float`.

3. Implantez la fonction `diff` de profil `float -> int -> float` qui calcule la différence entre `sin x` et `serie_sinus` pour des valeurs de x et n passées en paramètre.

4. Remplissez le tableau suivant avec les valeurs calculées par la fonction `diff` pour les valeurs correspondantes de x et n (on prendra `let pi=4.*.atan(1.0);;`).

n:	5	10	20	30
$x = 0.$				
$x = \pi/6$				
$x = \pi/4$				
$x = \pi/3$				
$x = \pi/2$				

2 Approximation d'une solution d'une équation numérique par une méthode dichotomique

2.1 Introduction

La méthode dichotomique est une technique simple pour résoudre une équation de la forme $f(x) = 0$ où f est une fonction continue. En effet on sait que si a et b sont deux réels tels que $f(a) < 0 < f(b)$ alors f a au moins un zéro entre a et b.

La méthode consiste à choisir deux valeurs initiales a et b telles que $f(a) < 0 < f(b)$, puis à réduire l'intervalle d'extrémités a et b de la façon suivante:
soit $m = (a + b)/2$

- si $f(m) < 0$ alors on a $f(m) < 0 < f(b)$, f a donc un zéro dans l'intervalle d'extrémités m et b.
- si $f(m) > 0$ alors on a $f(a) < 0 < f(m)$, f a donc un zéro dans l'intervalle d'extrémités a et m.

Dans les deux cas l'intervalle dans lequel on recherche une solution a été réduit de moitié. Le processus s'arrête lorsque l'on est arrivé à un intervalle assez petit.

2.2 Questions

1. Écrire une fonction `moyenne` qui calcule la moyenne de deux `float`.

2. Écrire une fonction `estPositive` qui teste si un `float` est positif.

3. Écrire une fonction `estAssezProche` à valeurs booléennes qui prend en argument deux `float` et détermine si la distance entre ces deux nombres est inférieure à un millième.

4. Compléter la fonction `cherche` suivante qui met en œuvre la stratégie décrite ci-dessus

```
let rec cherche =
function f -> function pt_neg -> function pt_pos ->
    let pt_milieu = moyenne pt_neg pt_pos in
        if assez_proche pt_neg  pt_pos
            then <<RESULTAT1>>
            else let valeur_test = f pt_milieu in
                 if positive valeur_test
                    then <<RESULTAT2>>
                    else <<RESULTAT3>>;;
```

   Proposer 3 équations dont vous connaissez des solutions et utiliser les pour tester cette fonction. Donner le résultat des tests.

5. Écrire une fonction `dichotomie` qui prend en argument la fonction f et deux `float` correspondant aux deux extrémités de l'intervalle. Cette fonction regarde quelle extrémité correspond aux valeurs négative et positive de la fonction `f`, puis appelle, avec les arguments appropriés, la fonction `cherche`. Si la fonction `f` a le même signe aux deux points donnés, la méthode dichotomique ne peut être utilisée et la fonction signale une erreur (on utilisera la fonction prédéfinie `failwith`).

6. Application

   - Écrire l'instruction CAML permettant de trouver une solution de $x^2 = 2$.
   - Écrire l'instruction CAML permettant de trouver une solution de $\cos(x) = x/4$ dans l'intervalle $[1, 2]$.

7. Modifier les fonctions `estAssezProche`, `cherche` et `dichotomie` en introduisant un paramètre `epsilon`, de telle façon que le processus s'arrête lorsque l'intervalle a une longueur inférieure à `epsilon`.

3 Fonction Modulo

Dans cet exercice, vous écrirez une fonction `modulo_bis`. La fonction `modulo_bis` est une fonction qui prend en entrée deux entiers naturels a et b et qui renvoie le reste de la division euclidienne de a par b.

1. Précisez le type de la fonction `modulo_bis`

2. Écrire une fonction **récursive** `modulo_bis` n'utilisant que des tests, additions et soustractions (bien entendu, vous n'avez pas le droit ni à la fonction mod, ni à la division)

3. Tester modulo_bis.