Université d'El Oued
Département d'informatique

Algorithmique et Structures de données

Licence 2 Informatique

Correction de l'Examen du 15 Janvier 2023

Q1.

(a) Fonction qui réalise l'exponentiation rapide (3 pts)

```
double quickPwr(double x, int n)
{
      if (n == 0) return 0.0;
      if (n % 2) return quickPwr(x * x, n / 2);
      else return (x * quickPwr(x, n - 1));
}
```

(b) Le nombre de multiplications est donnée par la fonction $\mathbf{log_2(n)}$. (1 pt)

Q2.

(a) Illustration du tri bulle (2 pts)

Input : V = {6, 2, 8, 4, 3, 7, 1, 5}
Itération 1 : V = {2, 6, 4, 3, 7, 1, 5, **8**}
Itération 2 : V = {2, 4, 3, 6, 1, 5, **7, 8**}
Itération 3 : V = {2, 4, 3, 1, 5, **6, 7, 8**}
Itération 4 : V = {2, 4, 3, 1, **5, 6, 7, 8**}
Itération 5 : V = {2, 3, 1, **4, 5, 6, 7, 8**}
Itération 6 : V = {2, 1, **3, 4, 5, 6, 7, 8**}
Itération 7 : V = {1, **2, 3, 4, 5, 6, 7, 8**}
Output: V = {1, 2, 3, 4, 5, 6, 7, 8}

(b) Fonction qui réalise le tri bulle (3 pts)

```
void triBulle (double t[], int n)
{
      int i, k;
      for (i = n - 1; i > 0; i--)
            for (k = 0; k < i; k++)
                  if (t[k] > t[k + 1]) swap (t, k, k + 1);
}
```

Q3.

(a) Déclarations de types pour une liste d'entiers implémentée en utilisant une liste doublement chaînée. (3 pts)

```
struct Cellule
{
      int contenu;
      struct Cellule *suivant;
      struct Cellule *precedent ;
};
typedef struct Cellule Cellule, *ListeDC;
```

(b) Une fonction qui inverse l'ordre des éléments d'une liste doublement chaînée. (3 pts)

```
ListeDC inserver(ListeDC a)
{
      ListeDC b, c ;
      b = a;
      while (!EstListeVide(a))
      {
            c = a->suivant;
            a->suivant = a->precedent;
            a->precedent = c;

            b = a;
            a = a->precedent;
      }
      return b;
}
```

Q3.

Une fonction (ici récursive) qui insère x dans une suite triée représentée par une pile **(5 pts)**

```
void Inserer(int x, Pile *p)
/*
      x est la valeur du nouvel élément ;
      p est un pointeur vers la structure pile ;
*/
{
      int y;

      if (EstPileVide(*p) || x < Sommet(*p))
      {
            Empiler(x, p);
      }
      else
      {
            y = Sommet(*p);
            Depiler(p);
            Inserer(x, p);
            Empiler(y, p);
      }
}
```