

1. Transformation d'une expression arithmétique parenthésée en expression arithmétique postfixée.

Exemple :  $(2 + 3) \times (4 + 5) \rightarrow 2\ 3 + 4\ 5 + \times$

```
Parenthesee_vers_postfixee (expr_par, n) {
  P = Pile_vide (); //P est une pile de caracteres
  n_post <- 0;
  Pour i de 1 A n Faire {
    c <- expr_par[i];
    Selon que {
      c dans '0'..'9' :
        expr_post[n_post] <- c;
        n_post <- n_post + 1;

      c = '(' :
        Empiler(P, c);

      c = ')':
        Tant que (Valeur_sommet(P) <> '(') Faire {
          expr_post[n_post] <- Valeur_sommet (P);
          n_post <- n_post + 1;
          Depiler (P);
        }
        Depiler (P);

      c dans ('+', '-', '*', '/') :
        Tant que (Non Pile_vide? (P) ET
          Valeur_sommet (P) <> '(' ET
          Priorite (Valeur_sommet (P)) >= Priorite (c))) {
          expr_post[n_post] <- Valeur_sommet (P);
          n_post <- n_post + 1;
          Depiler (P);
        }
        Empiler (P, c);
    }
  }
  Tant que (Non Pile_vide? (P)) Faire {
    expr_post[n_post] <- Valeur_sommet (P);
    n_post <- n_post + 1;
    Depiler (P);
  }
  retourner expr_post, n_post;
}
```

2. Écrire un algorithme d'inversion d'une file.

```

Inverser_file (F) {
  P = Pile_vide ();
  Tant que Non File_vide? (F) Faire {
    Empiler (P, Valeur_premier(F));
    Defiler (F);
  }
  Tant que Non Pile_vide? (P) Faire {
    Enfiler (F, Valeur_sommet (P));
    Depiler (P);
  }
}

```

3. Écrire un algorithme de tri à l'aide de trois piles.

```

Tri_Pile (P) {
  Tant que non Pile_Vide(P) faire {
    Tant que non Pile_vide(P1) et sommet(P) <= sommet(P1) faire
      Empiler(Dépiler(P1),P3);
    Empiler(Dépiler(P),P1);
    Tant que non Pile_Vide(P3) faire
      Empiler(P1, Dépiler(P3));
    Tant que non Pile_Vide(P1) faire
      Empiler(P,Dépiler(P1))
  }
}

```