Les pointeurs

Exercice 1

type x = pointeur sur (int)
type y = pointeur sur (pointeur sur (int))
type z = int

Exercice 2

type x = pointeur sur (int)
type y = int
type z = pointeur sur (int)

Exercice 3

- **Analyse lexicale**

  Ajout de token : *&*, *new* et *null*

- **Analyse syntaxique**

  TYPE -> TYPE *

  I -> A = E;

  A -> id

  A -> * A

  E -> & id

  E -> (new TYPE)

  E -> null

  E -> A

- **AST Syntaxe / TDS / Type**

```
type typ = ... | Pointeur of typ

    type expression =
    | ...
    | Ident of string
    | Null
    | Affectable of affectable
    | Adresse of string
    | New of typ
```

```
    type affectable = Deref of affectable | Ident of info_ast
```

- **Gestion d'identifiant**

```
let analyse_tds_instruction tds i =
match i with
| AstSyntaxe.Affectation(a, e) ->
    let na = analyse_tds_affectable tds a in
    let ne = analyse_tds_expression tds e in
        AstTDS.Affectation(na, ne)

let analyse_tds_affectable tds modif a =
    match a with
    | AstSyntaxe.Ident n ->
        recherche_globale       (*  | N'existe pas -> exception
                                    | Existe et fonction -> exception
                                    | Existe et const ->
                                        si modif exception sinon
Entier
                                    | Existe et var -> AstTds.Ident
info
                                   *)
    | AstSyntax.Deref ai ->
        AstTds.Deref(analyse_tds_affectable tds modif ai)

let analyse_tds_expression tds e =
    match e with
    | AstSyntaxe.Null -> AstTDS.Null
    | AstSyntaxe.New t -> AstTDS.New t
    | AstSyntaxe.Affectable a ->
        AstTDS.Affectable(analyse_tds_affectable tds ??? a)
    | AstSyntaxe.Adresse n ->
        recherche_globale       (*  | N'existe pas -> exception
                                    | Existe et fonction ou const ->
exception
                                    | Existe et var -> AstTds.Adresse
info
                                   *)
```

- **Typage**
    - Jugement de typage

$$\sigma \vdash nul : \text{Pointeur(Undefined)}$$

$$\frac{\sigma \vdash T : \tau}{\sigma \vdash newT : Pointeur(\tau)}$$

$$\frac{\sigma \vdash id : \tau}{\sigma \vdash \&id : Pointeur(\tau)}$$

$$\frac{\sigma \vdash a : Pointeur(\text{T})}{\sigma \vdash *a : \tau}$$

- **TAM**

  Code RAT :

```
int *x = (new int);
*x = 4;
int z = 18;
int *y = &z;
*y = *x;
```

  Code TAM :

```
PUSH 1
LOADL 1
SUBR MAlloc
STORE (1) 0[SB]
LOADL 4
LOAD (1) 0[SB]
STOREI (1)
PUSH 1
LOADL 18
STORE (1) 1[SB]
PUSH 1
LOADA 1[SB]
STORE (1) 2[SB]
LOADA 0[SB]
LOADI (1)
LOADI (1)
LOAD (1) 2[SB]
STOREI (1)
```