NOTE IMPORTANTE:

Comme pour le solveur SAT du TME 4, il nous a été impossible de lancer SATPLAN depuis Windows. Nous avons donc eu recours à l'éditeur et le solveur fournis sur le site suivant.
A noter que les plans retournés par ce solveur ne sont exportables qu'au format `.txt` et non `.pddl.soln`.

1 Représentation de problème de plannification en PDDL

Exercice 1: Prise en main de PDDL

Voir dans le dossier `Exercice 1` les fichiers:

- `blockWorld-domain.pddl`
- `blockWorld-problem.pddl`

La solution a été exportée au format `.txt` (fichier `plan.txt`).

2 Planification par encodage SAT

Exercice 2: Prise en main de SATPLAN

Comme précisé plus haut, nous n'avons pas pu utiliser SATPLAN sur Windows et avons donc utilisé l'éditeur et le solveur fournis sur ce site.

Le plan retourné par résolution du problème tel que décrit dans l'exercice 1 du TD 7 est le suivant *(cf. fichier* `Exercice 1/plan.txt`*)*:

```
(unstack b a)
(stack b c)
(pick-up a)
(stack a b)
```

Nous définissons maintenant des problèmes plus compliqués que nous nous attacherons à résoudre...

Problème 1: *(Inversion de blocs)*
Au départ, le bloc C est sur le bloc D, le bloc B sur le bloc A (fichier `blockWorld-problem-1.pddl`). Dans l'état final, il faut que les blocs soient inversés: le bloc D doit être sur le bloc C et le bloc A sur le bloc B.

```
plan-problem-1.txt:
    (unstack b a)
    (put-down b)
    (unstack c d)
    (put-down c)
    (pick-up d)
    (stack d c)
    (pick-up a)
    (stack a b)
```

Problème 2:
Au départ, deux blocs A-C-E avec E sur la table, et B-D-F avec F sur la table (fichier `blockWorld-problem-2`). On veut obtenir un seul bloc A-B-C-D-E-F avec F sur la table.

```
plan-problem-2.txt:
    (unstack a c)
    (put-down a)
    (unstack b d)
    (put-down b)
    (unstack d f)
    (put-down d)
    (unstack c e)
    (stack c d)
    (pick-up e)
    (stack e f)
    (unstack c d)
    (put-down c)
    (pick-up d)
    (stack d e)
    (pick-up c)
    (stack c d)
    (pick-up b)
    (stack b c)
    (pick-up a)
    (stack a b)
```

Problème 3: *(Permutation dans un bloc)* Au départ, un seul bloc A-B-C-D-E-F, avec A tout en haut de la pile et F sur la table (fichier `blockWorld-problem-3`). On veut obtenir un seul bloc C-E-A-F-B-D.

```
plan-problem-3.txt:
    (unstack a b)
    (put-down a)
    (unstack b c)
    (put-down b)
    (unstack c d)
    (put-down c)
    (unstack d e)
    (put-down d)
    (pick-up b)
    (stack b d)
    (unstack e f)
    (put-down e)
    (pick-up f)
    (stack f b)
    (pick-up a)
    (stack a f)
    (pick-up e)
    (stack e a)
    (pick-up c)
    (stack c e)
```

Exercice 3: Variante du monde des blocs

Afin de n'utiliser que deux actions `moveTo(X,Y,Z)` et `moveToTable(X,Y)` et pouvoir retirer les prédicats `ontable`, `holding` et `handempty`, nous avons dû apporter les modifications suivantes:

- Nous ne considérons plus seulement le type `block` mais également le type `support` qui servira pour définir notre table. Ces deux types sont eux-mêmes inclus dans le type `object`. Ainsi un block pourra être pris depuis un autre bloc ou depuis un support: de manière générale depuis un `object`
- Ainsi il est possible de supprimer le prédicat `ontable` en généralisant le prédicat `on`, qui prend désormais deux arguments `?x` de type `block` et `?y` de type `object` (un block peut être sur un autre block ou sur un support)
- On ajoute une constante `table` de type `support`
- Pour les préconditions de l'action `moveToTable`, on n'a pas besoin de spécifier que la table doit être vide !

Notre problème est traduit dans les deux fichiers suivant, dans le dossier `Exercice 3`:

- `blocksSimp-domain.pddl`
- `blocksSimp-problemTD.pddl`

Le plan renvoyé est stocké dans le fichier `problemTD-plan.txt`:

```
problemTD-plan.txt:
    (moveto b a c)
    (moveto a table b)
```

Exercice 4: Singe et bananes

Notre problème est traduit dans les deux fichiers suivant, dans le dossier `Exercice 4`:

- `chasseAuxBananes-domain.pddl`
- `chasseAuxBananes-problem.pddl`

Le plan renvoyé est stocké dans le fichier `chasseAuxBananes-plan.txt`:

```
chasseAuxBananes-plan.txt:
    (sedeplace a c)
    (prend caisse c bas)
    (sedeplace c b)
    (depose caisse b bas)
    (montecaisse b)
    (prend bananes b haut)
```

Autrement dit, le singe doit se déplacer de la position `a` à la position `c`, prendre la caisse, se déplacer en `b`, poser la caisse, monter dessus puis prendre les bananes.

3 Planification par ASP

Le dossier `Exercices 5-6` contient les fichiers python permettant la planification, ainsi qu'un sous-dossier `exemples` ne contenant à l'origine que les fichiers `domain` et `problem`.

Lire le README pour des explications detaillees sur l'utilisation des classes et les commandes.

Exercice 5: Parseur PDDL vers ASP-STRIPS

Notre parseur est écrit dans le fichier python `Parser.py` du dossier `Exercices 5-6`. *(Note: les commandes à utiliser sur nos fichiers jouets sont données en commentaires, en fin de fichier)*.

Les traductions des problèmes du monde des blocs et des bananes sont stockées dans les fichiers suivants:

- `exemples/asp_blockWorld.txt`
- `exemples/asp_chasseAuxBananes.txt`

Exercice 6: Planificateur STRIPS

Notre planificateur pour un horizon `n` donné est écrit dans le fichier python `Planner.py` du dossier `Exercices 5-6`. La classe permettant de générer un plan minimal (avec un horizon maximal `nmax` donné) est la classe `ASPPLAN` du fichier `ASPPLAN.py`. *(Note: Il faut absolument lire le README. Les commandes à utiliser sur nos fichiers jouets sont données en commentaires, en fin de fichiers)*.

Les planificateurs générés pour les problèmes du monde des blocs et des bananes, avec un horizon maximal `nmax = 50`, sont stockées dans les fichiers suivants:

- `exemples/plan_blockWorld.txt` (plan minimal pour un horizon $n = 4$)
- `exemples/plan_chasseAuxBananes.txt` (plan minimal pour un horizon $n = 6$)

Afin d'afficher les plans minimaux générés pour les deux problèmes, il faut exécuter sur un IDE les commandes données dans le REAME (fonction `main` du fichier `ASPPLAN.py`).

Capture des plans affichés par la fonction `main` de `ASPPLAN.py`:

```
*** Recherche d'un plan minimal pour le domaine exemples/
blockWorld-domain.pddl et le problème exemples/blockWorld-
problem.pddl

SATISFIABLE: Plan minimal trouvé pour n = 4


Plan:
 perform(unstack(b,a),0) perform(stack(b,c),1)
perform(pickup(a),2) perform(stack(a,b),3)
```

Abbildung 1: Plan pour le problème des blocs

```
*** Recherche d'un plan minimal pour le domaine exemples/
chasseAuxBananes-domain.pddl et le problème exemples/
chasseAuxBananes-problem.pddl

SATISFIABLE: Plan minimal trouvé pour n = 6


Plan:
 perform(seDeplace(a,c),0) perform(prend(caisse,c,bas),1)
perform(seDeplace(c,b),2) perform(depose(caisse,b,bas),3)
perform(monteCaisse(b),4) perform(prend(bananes,b,haut),5)
```

Abbildung 2: Plan pour le problème des bananes