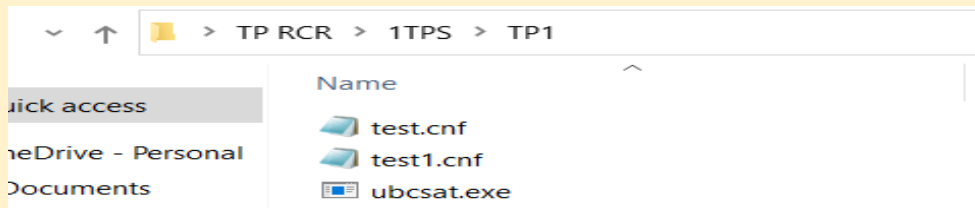




Partie N° 1

Etape 1 :

Installation de l'outil



Etape 2 :

Exécution du solveur SAT et teste de la satisfiabilité des deux fichiers : test.cnf et test1.cnf.

Pour L'essai numéro 1 Nous avons exécuté la commande suivante :

```
ubcsat -alg saps -i test.cnf -solve
```

1^{er} test :

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2364]
(c) Microsoft Corporation. All rights reserved.

A:\TP RCR\tools>ubcsat -alg saps -i test.cnf -solve
#
# UBCSAT version 1.1.0 (Sea to Sky Release)
# http://www.satlib.org/ubcsat
#
# ubcsat -h for help
#
# -alg saps
# -runs 1
# -cutoff 100000
# -timeout 0
# -gtimeout 0
# -noimprove 0
# -target 0
# -wtarg 0
# -seed 5051455
# -solve 1
# -findj -numsol 1
# -findunique 0
# -srestart 0
# -prestart 0
# -drestart 0
#
# -alpha 1.3
# -rho 0.8
#

```

Solution :

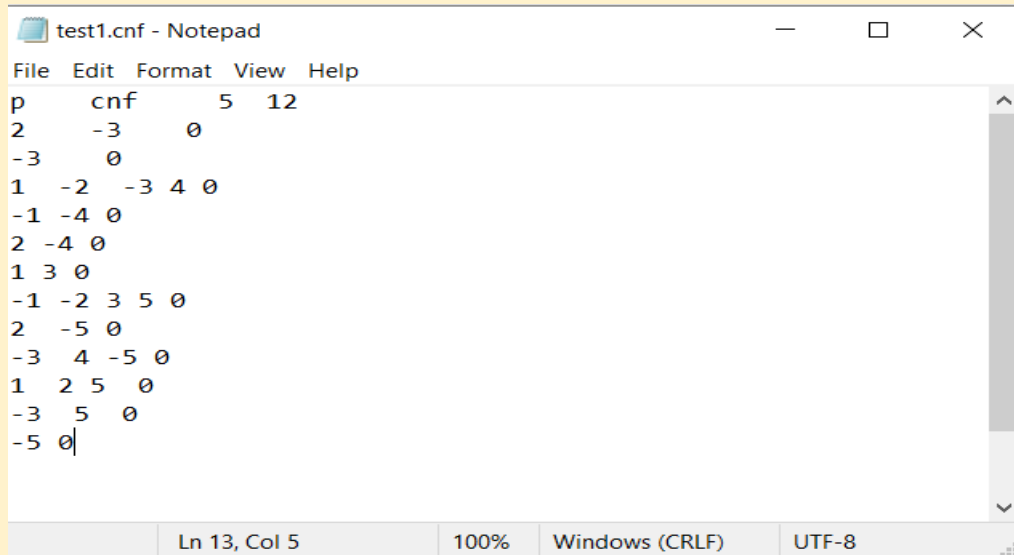
Cette partie affiche la sortie de l'UBCSAT. Puisque le solveur a trouvé une solution, nous pouvons dire que le fichier « test.cnf » est satisfiable

```

# beststep: Step of Best (Lowest) # of False Clauses Found
# steps: Total Number of Search Steps
#
#      F Best      Step      Total
#      Run N Sol'n  of      Search
#      No. D Found  Best    Steps
#
#      1 1      0      4      4
#
# Solution found for -target 0
#
# 1 2 -3 -4 5
#

```

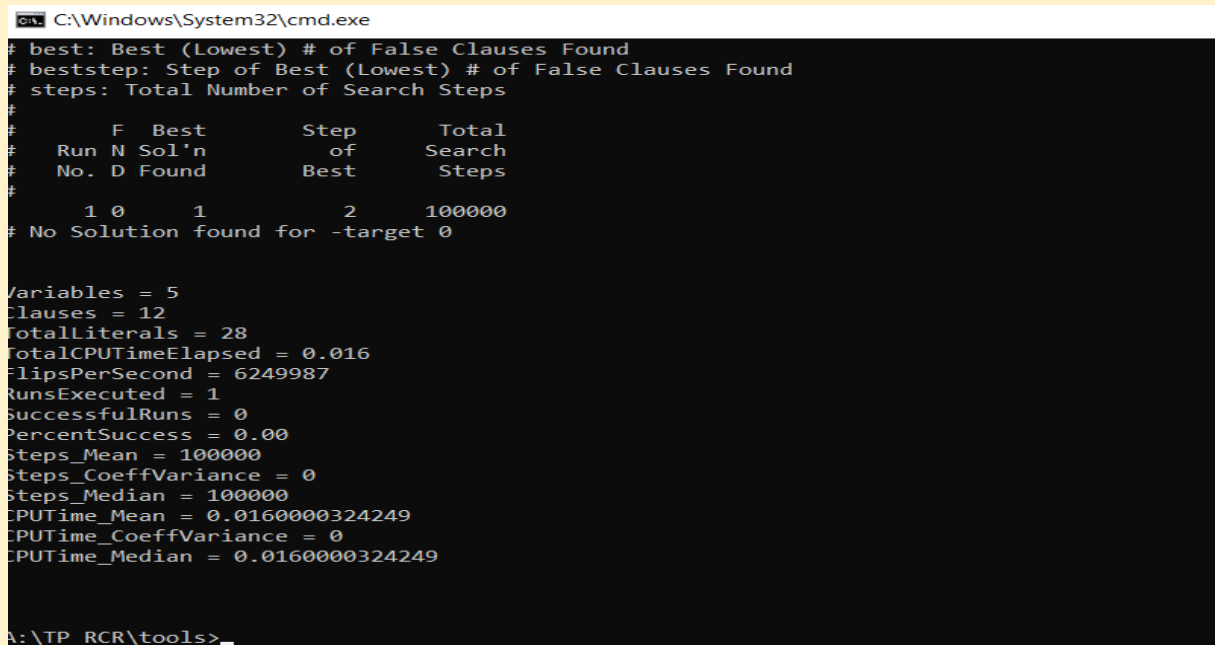
Test sur :



```
test1.cnf - Notepad
File Edit Format View Help
p cnf 5 12
2 -3 0
-3 0
1 -2 -3 4 0
-1 -4 0
2 -4 0
1 3 0
-1 -2 3 5 0
2 -5 0
-3 4 -5 0
1 2 5 0
-3 5 0
-5 0
Ln 13, Col 5 100% Windows (CRLF) UTF-8
```

Solution :

Après l'exécution du solveur sur "test1.cnf", le message (No Solution found) s'affiche signifiant que la base n'est passatisfiable et donc non exploitable



```
C:\Windows\System32\cmd.exe
# best: Best (Lowest) # of False Clauses Found
# beststep: Step of Best (Lowest) # of False Clauses Found
# steps: Total Number of Search Steps
#
#      F Best      Step      Total
#      Run N Sol'n  of      Search
#      No. D Found  Best    Steps
#
#      1 0      1      2      100000
# No Solution found for -target 0

Variables = 5
Clauses = 12
TotalLiterals = 28
TotalCPUtimeElapsed = 0.016
FlipsPerSecond = 6249987
RunsExecuted = 1
SuccessfulRuns = 0
PercentSuccess = 0.00
Steps_Mean = 100000
Steps_CoeffVariance = 0
Steps_Median = 100000
CPUtime_Mean = 0.0160000324249
CPUtime_CoeffVariance = 0
CPUtime_Median = 0.0160000324249

A:\TP_RCR\tools>
```

Etape 3 :

a) Ci-dessous, les énoncés vu en cours :

- Les nautilus sont des céphalopodes ;
- Les céphalopodes sont des mollusques ;
- Les mollusques ont généralement une coquille ;
- Les céphalopodes n'en ont généralement pas ;
- Les nautilus en ont une.
- a est un nautilus,
- b est un céphalopode,
- c est un mollusque.

Application de la règle : $a \supset b \equiv \neg a \vee b$

$(\neg Na \vee Cea); (\neg Nb \vee Ceb); (\neg Nc \vee Cec)$

$(\neg Cea \vee Ma); (\neg Ceb \vee Mb); (\neg Cec \vee Mc)$

$(\neg Na \vee Coa); (\neg Nb \vee Cob); (\neg Nc \vee Coc)$

$(\neg Ma \vee Cea \vee Coa); (\neg Ma \vee \neg Na \vee Coa)$

$(\neg Mb \vee Ceb \vee Cob); (\neg Mb \vee \neg Nb \vee Cob)$

$(\neg Mc \vee Cec \vee Coc); (\neg Mc \vee \neg Nc \vee Coc)$

$(\neg Cea \vee Na \vee \neg Coa)$

$(\neg Ceb \vee Nb \vee \neg Cob)$

$(\neg Cec \vee Nc \vee \neg Coc)$

$Na; Ceb; Mc$

Nous avons 12 variables et 21 clauses au total.

Représentation dans le fichier : 1 = Na ;

2 = Nb ;

3 = Nc ;

4 = Cea ;

5 = Ceb ;

6 = Cec ;

7 = Coa ;

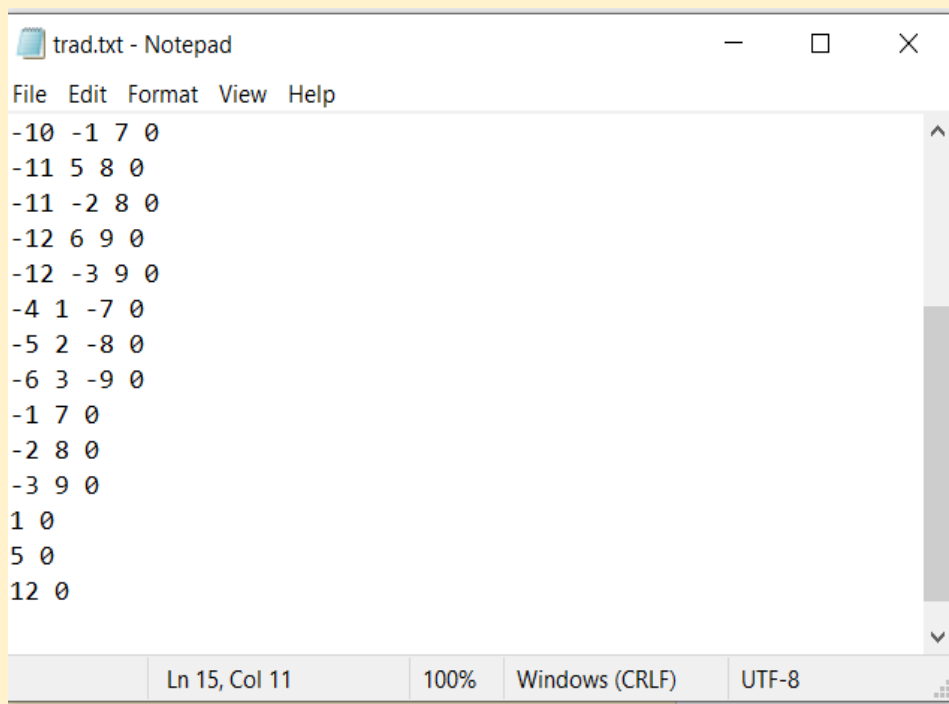
8 = Cob ;

9 = Coc ;

10 = Ma ;

11 = Mb ;

12 = Mc ;

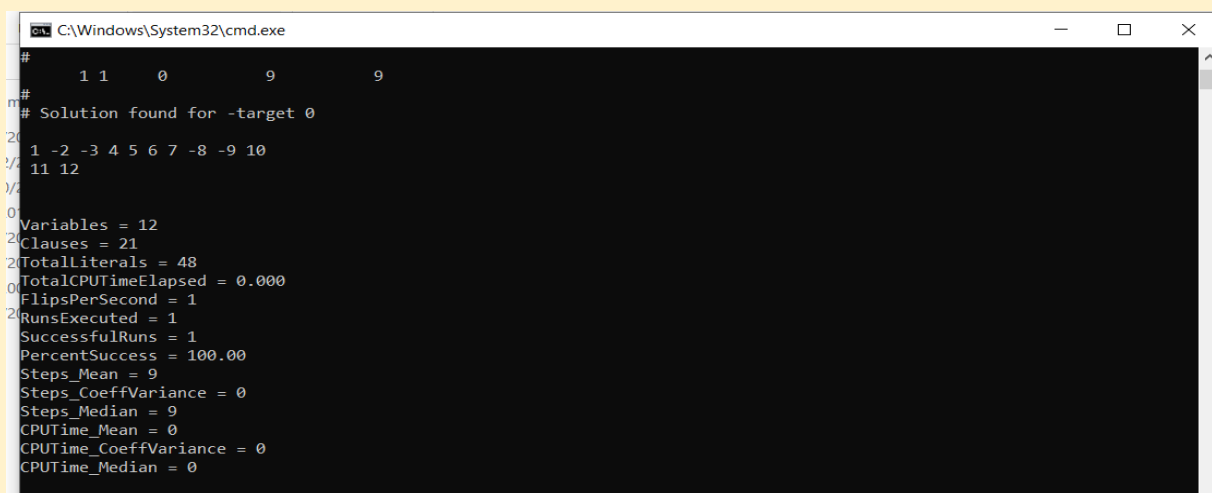


```

File Edit Format View Help
-10 -1 7 0
-11 5 8 0
-11 -2 8 0
-12 6 9 0
-12 -3 9 0
-4 1 -7 0
-5 2 -8 0
-6 3 -9 0
-1 7 0
-2 8 0
-3 9 0
1 0
5 0
12 0
Ln 15, Col 11 100% Windows (CRLF) UTF-8

```

Resultat :



```

C:\Windows\System32\cmd.exe
#
# 1 1 0 9 9
#
# Solution found for -target 0
1 -2 -3 4 5 6 7 -8 -9 10
11 12
Variables = 12
Clauses = 21
TotalLiterals = 48
TotalCPUTimeElapsed = 0.000
FlipsPerSecond = 1
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 9
Steps_CoeffVariance = 0
Steps_Median = 9
CPUTime_Mean = 0
CPUTime_CoeffVariance = 0
CPUTime_Median = 0

```

Solution : $Na \wedge \neg Nb \wedge \neg Nc \wedge Cea \wedge Ceb \wedge CeC \wedge Coa \wedge \neg Cob \wedge \neg CoC \wedge Ma \wedge Mb \wedge Mc$

b) Teste de Benchmarking

On télécharge deux fichiers : un satisfiable et un non-satisfiable, et nous le testons.

1)

```
C:\Windows\System32\cmd.exe
#
# Solution found for -target 0

1 2 -3 4 -5 -6 -7 -8 -9 -10
-11 -12 13 14 15 -16 17 18 19 20
21 22 -23 -24 -25 -26 -27 28 -29 -30
-31 32 -33 34 35 -36 -37 38 39 -40
-41 -42 -43 44 45 -46 -47 48 -49 50
51 -52 -53 -54 -55 56 57 58 59 60
61 -62 -63 -64 -65 -66 -67 -68 -69 -70
-71 -72 -73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90
91 92 93 94 -95 96 97 -98 99 100
101 102 103 104 -105 106 -107 108 109 110
111 112 113 114 115 116 117 118 -119 120
121

Variables = 121
Clauses = 207
TotalLiterals = 518
TotalCPUTimeElapsed = 0.000
FlipsPerSecond = 1
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 419
Steps_CoeffVariance = 0
Steps_Median = 419
CPUTime_Mean = 0
```

La base est satisfiable.

2)

```
C:\Windows\System32\cmd.exe
# best: Best (Lowest) # of False Clauses Found
# beststep: Step of Best (Lowest) # of False Clauses Found
# steps: Total Number of Search Steps
#
#      F Best      Step      Total
#      Run N Sol'n  of      Search
#      No. D Found  Best    Steps
#
#      1 0      1      21      100000
# No Solution found for -target 0

Variables = 100
Clauses = 160
TotalLiterals = 480
TotalCPUTimeElapsed = 0.015
FlipsPerSecond = 6666620
RunsExecuted = 1
SuccessfulRuns = 0
PercentSuccess = 0.00
Steps_Mean = 100000
Steps_CoeffVariance = 0
Steps_Median = 100000
CPUTime_Mean = 0.0150001049042
CPUTime_CoeffVariance = 0
CPUTime_Median = 0.0150001049042

A:\TP RCR\tools>
```

La base est no satisfiable.

Etape 4 : simulation de l'inférence d'une base de connaissances

L'algorithme de résonnement par l'absurde en langage python:

```
import sys, os, re, random
if len(sys.argv) < 2:
    print("veuillez indiquer un fichier de base de connaissances")
    exit()
# on utilise un fichier temporaire pour représenter BCU{¬phi}
f_input = open(sys.argv[1], mode='r', encoding='utf-8')
f_tmp = open("tmp.cnf", mode='w', encoding='utf-8')
phi = 0
for line in f_input:
    # lecture du fichier jusqu'à trouver la ligne d'entête
    if len(line) > 0 and line[0] == 'p':
        # générer un littéral et mettre à jour l'entête
        x = re.split(r'\s+', line)[:4]
        print(x)
        phi = random.randint(1, int(x[2])) # générer random phi
        if random.randint(0, 1) == 0: # générer random sing
            phi = -phi
        print("phi =", phi)
        f_tmp.write("p cnf " + x[2] + " " + str(int(x[3]) + 1) + "\n") #
        # incrémenter le nbr des clauses
        break;
    else:
        f_tmp.write(line)
# continuer la lecture du fichier jusqu'à la fin
for line in f_input:
    f_tmp.write(line)
# ajouter la négation du littéral généré à la fin de fichier
f_tmp.write(str(-phi) + " 0\n")
f_input.close()
f_tmp.close()
# on exécute ubcsat avec le nouveau fichier comme argument et on ajoute
# -r out null -r stats null pour retourner juste la solution
os.system("ubcsat -alg saps -i tmp.cnf -solve -r out null -r stats null")
os.remove("tmp.cnf")
```

Déroulement :

<pre>A:\TP_RCR\1TPS\TP1>py.py test.cnf ['p', 'cnf', '5', '11'] phi = -3 # No Solution found for -target 0</pre>	<pre>A:\TP_RCR\1TPS\TP1>py.py test.cnf ['p', 'cnf', '5', '11'] phi = -1 # # Solution found for -target 0 1 2 -3 -4 5</pre>
PHI = -3	PHI = -1

On voit depuis ces exécutions que pour $\phi = -3$, on ne peut pas inférer ϕ depuis la base de connaissances, Par contre, pour $\phi = -1$, $BCU\{\neg\phi\} \vdash \perp$ car $SAT(BCU\{\neg\phi\}) \vdash \perp$, donc $BC \vdash \phi$.