**Nom : Yesmine Zribi**
**Numéro Etudiant : 8402454**
**CSI2520**

**Prolog Devoir 0**

-Veuillez entrer la requête : minimumTransportCost(X) afin d'exécuter le programme.

Description des prédicats :

| Prédicats | Description |
|---|---|
| listStream(File,L) ; listStreamInit(File,L) | Ces prédicats ouvrent des fichiers les lisent et retournent une list contenant chaque élément dans le fichier |
| warehouseNum(File,N) | Ce prédicat calcule le nombre d'entrepôts dans un fichier |
| factoryNum(File,N) | Ce prédicat calcule le nombre d'usines dans un fichier |
| listUntilSupply(File,L) ; listUntilSupplyInit(File,L) | Ces prédicats retournent une liste de tous les éléments dans un ficher jusqu'au mot 'Supply' |
| listAfterSupply(File,L) ; listAfterSupplyInit(File,L) | Ces prédicats retournent une liste de tous les éléments après le mot 'Supply' |
| listOfCostsAndSupplies(File,L) ; listOfCostsAndSuppliesInit(File,L) | Ces prédicats retournent une liste de tous les éléments après le mot 'Supply' et avant le mot 'Demand ' |
| findSubStartingFromF(File,F,Result) ; findSubStartingFromFInit(File,F,Result) | Ces prédicats retournent une liste commençant par la rangée spécifiée par F |
| costsAndSuppliesF(File,F,Result) ; costsAndSuppliesFInit(File,F,Result) | Ces prédicats retournent une rangée du fichier spécifiée par F (ex : si F est 1, ce prédicat retourne la première rangée). |
| listOfCosts(File,F,Costs) | Ce prédicat retourne une liste des coûts de de chaque cellule dans la rangée F |
| costs(File,Result) | Ce prédicat retourne une matrice des coûts de transport de chaque cellule du fichier File |
| listOfCostsInit(File,F,Costs) | Ce prédicat retourne une liste contenant les biens dans chaque cellule dans une rangée F |
| costsInit(File,Result) | Ce prédicat retourne une matrice de la solution initiale. |
| supplyF(File,F,Result) | Ce prédicat retourne la production totale de l'usine F |
| supplies(File,Result) | Ce prédicat retourne une liste de la production totale de chaque usine. |
| demand(File,Result) | Ce prédicat retourne une liste des demandes de chaque entrepôt. |

| | |
|---|---|
| convert(FactoryNum, WarehouseChar, [Cost\|RestC],[Supply\|RestS], Result) | Ce prédicat prend le numéro d'identification de l'usine, la lettre d'identification de l'entrepôt, la matrice des coûts et la matrice de la solution initiale pour crée une matrice du format supply(FactoryNum,WarehosueNum,Cost,Supply, Label). |
| matrix(File1,File2,X) | Ce prédicat retourne une matrice de transport des fichiers File1 et File2 donnés. |
| select_row(TargetC,Matrix,Result) | Ce prédicat retourne toutes les celluls (supply) dans la même ligne que la cellule TargetC dans la matrice Matrix |
| select_column(supply(Fac,TargetL,_,_,_),Matrix, Result) | Ce prédicat retourne toutes les cellules (supply) dans la même colonne que la cellule TargetC dans la matrice Matrix |
| adjacentCells(TargetC,Matrix,Result) | Ce prédicat retourne toutes les cellules adjacentes à la cellule TargetC dans Matrix |
| isAdjacent(Cell1,Matrix,Cell2) | Ce prédicat retourne vrai si cellules1 et cellule2 sont adjacentes dans Matrix |
| fullAdjacent(TargetC,Matrix,Result) | Ce prédicat retourne toutes les cellules adjacentes à TargetC mais qui sont **remplies** |
| path(Matrix,EmptyCell,Result) | Ce prédicat trouve un chemin à partir de la cellule vide EmptyCell dans Matrix |
| get_emptyCell(Row,Result) | Ce prédicat retourne toutes les cellules vides dans une rangée donnée Row |
| get_all_emptyCells(Matrix,Result) | Ce prédicat retourne toutes les cellules vides dans Matrix |
| find_paths(Matrix,Result) | Ce prédicat retournes tous les chemins qu'on peut trouver qui commencent par une cellule vide dans Matrix |
| path_cost(Path,Result) | Ce prédicat calcule le coût marginal d'un chemin donné |
| all_path_cost(Matrix,Result) | Ce prédicat calcule le coût marginal de chaque chemin dans Matrix |
| smallest_cost(Matrix,Result) | Ce prédicat détermine le coût marginal le plus bas dans une matrice Matrix |
| smallest_path(Matrix,Result) | Ce prédicat retourne le plus petit chemin dans une matrice Matrix |
| negative_cells(Matrix,Result) | Ce prédicat retourne toutes les cellules dans le chemin le plus petit auxquelles on va soustraire des biens |
| smallest_negative_cell(Matrix,Result) | Ce prédicat retourne la cellule négative avec la plus petit valeur (celle qui va devenir la nouvelle cellule vide après transfert des biens sur le chemin) |
| transfer_InPath(Matrix,Result) | Ce prédicat entreprend le transfert des biens dans le chemin le plus petit (de la plus petite cellule négative) |

| | |
|---|---|
| modify_matrix(Matrix,Supply,Result) | Ce prédicat modifie la matrice basée sur la modification faite sur le chemin le plus petit trouvé |
| replace(Matrix,Supply,Replaced,NewRow) | Ce prédicat remplace une rangée de Matrix avec une rangée dans laquelle Supply est mise à jour avec les nouvelles valeurs |
| transfer_InMatrix(Matrix,Result) | Ce prédicat entreprend le réarrangement de la matrice basé sur le transfert du plus petit chemin |
| optimal_sol(File1,File2,Result) | Ce prédicat trouve la solution optimale du problème de transport en étant fourni un ficher (File1) de la description du problème et un fichier (File2) de la solution initiale |
| row_tc(Row,Result) | Ce prédicat calcule le coût total d'une rangée donnée Row |
| total_cost(File1,File2,Result) | Ce prédicat calcul le coût total d'une matrice générée par la solution initiale et la description du problème donnés par File1 et File2 |
| **minimumTransportCost(Cost)** | **Prédicat a appeler qui fera appel à une série des prédicat ci-dessus et qui résoudra le problème.** Ce prédicat final demande à l'utilisateur un fichier de la description du problème, un fichier de la solution initiale, en fait une matrice, continue à chercher le chemin le plus court jusqu'à ce qu'aucune nouvelle optimization n'est possible et retourne le coût marginal le plus optimal. |

-Exemples des résultats obtenus en passant la requête principale minimumTransportCost(Cost) avec les fichiers du petit problème sur Brighspace :

```
5 ?- minimumTransportCost(C).
Please enter the cost file:'prob1.txt'.
Please enter the initial solution file:|: 'prob1_InitialSol.txt'.
[[supply(1,A,11,25,UnexploredFull),supply(1,B,13,225,UnexploredFull),supply(1,C,17,0,UnexploredEmpty)
,supply(1,D,14,0,UnexploredEmpty)],[supply(2,A,16,0,UnexploredEmpty),supply(2,B,18,0,UnexploredFull),
supply(2,C,14,275,UnexploredFull),supply(2,D,10,25,UnexploredEmpty)],[supply(3,A,12,175,UnexploredEmp
ty),supply(3,B,24,0,UnexploredEmpty),supply(3,C,13,0,UnexploredFull),supply(3,D,8,225,UnexploredFull)
]]
C = 11200.
```

-Exemple des résultats obtenus en passant la requête principlale minimumTransportCost(Cost) avec les fichiers de la matrice 10x10 sur Brightspace :

```
6 ?- minimumTransportCost(C).
Please enter the cost file:'prob2.txt'.
Please enter the initial solution file:|: 'prob2_InitialSol.txt'.
[[supply(1,A,6,20,UnexploredFull),supply(1,B,7,25,UnexploredFull),supply(1,C,10,0,UnexploredEmpty),su
pply(1,D,16,0,UnexploredEmpty),supply(1,E,5,75,UnexploredFull),supply(1,F,8,55,UnexploredFull),supply
(1,G,15,0,UnexploredEmpty),supply(1,H,20,0,UnexploredEmpty),supply(1,I,6,0,UnexploredEmpty),supply(1,
J,8,0,UnexploredEmpty)],[supply(2,A,10,0,UnexploredEmpty),supply(2,B,14,0,UnexploredEmpty),supply(2,C
,8,110,UnexploredFull),supply(2,D,17,0,UnexploredEmpty),supply(2,E,13,0,UnexploredEmpty),supply(2,F,9
,90,UnexploredFull),supply(2,G,18,0,UnexploredEmpty),supply(2,H,20,0,UnexploredEmpty),supply(2,I,9,0,
UnexploredEmpty),supply(2,J,7,0,UnexploredEmpty)],[supply(3,A,9,0,UnexploredEmpty),supply(3,B,4,225,U
nexploredFull),supply(3,C,8,0,UnexploredEmpty),supply(3,D,12,0,UnexploredEmpty),supply(3,E,10,0,Unexp
loredEmpty),supply(3,F,10,0,UnexploredEmpty),supply(3,G,14,0,UnexploredEmpty),supply(3,H,5,0,Unexplor
edEmpty),supply(3,I,9,0,UnexploredEmpty),supply(3,J,10,0,UnexploredEmpty)],[supply(4,A,12,0,Unexplore
dEmpty),supply(4,B,8,0,UnexploredEmpty),supply(4,C,9,0,UnexploredEmpty),supply(4,D,10,0,UnexploredEmp
ty),supply(4,E,6,0,UnexploredEmpty),supply(4,F,15,0,UnexploredEmpty),supply(4,G,4,300,UnexploredFull)
,supply(4,H,9,0,UnexploredEmpty),supply(4,I,7,0,UnexploredEmpty),supply(4,J,0,0,UnexploredEmpty)],[su
pply(5,A,6,130,UnexploredFull),supply(5,B,9,0,UnexploredEmpty),supply(5,C,17,0,UnexploredEmpty),suppl
y(5,D,7,0,UnexploredEmpty),supply(5,E,6,0,UnexploredEmpty),supply(5,F,13,0,UnexploredEmpty),supply(5,
G,6,0,UnexploredEmpty),supply(5,H,7,0,UnexploredEmpty),supply(5,I,6,0,UnexploredEmpty),supply(5,J,0,1
20,UnexploredFull)],[supply(6,A,9,0,UnexploredEmpty),supply(6,B,10,0,UnexploredEmpty),supply(6,C,9,0,
UnexploredEmpty),supply(6,D,13,0,UnexploredEmpty),supply(6,E,9,0,UnexploredEmpty),supply(6,F,8,0,Unex
ploredFull),supply(6,G,9,0,UnexploredEmpty),supply(6,H,3,100,UnexploredEmpty),supply(6,I,4,0,Unexplor
edEmpty),supply(6,J,9,0,UnexploredEmpty)],[supply(7,A,16,0,UnexploredEmpty),supply(7,B,18,0,Unexplore
dEmpty),supply(7,C,7,0,UnexploredEmpty),supply(7,D,14,0,UnexploredEmpty),supply(7,E,5,0,UnexploredEmp
ty),supply(7,F,6,60,UnexploredFull),supply(7,G,10,0,UnexploredEmpty),supply(7,H,5,0,UnexploredEmpty),
supply(7,I,4,90,UnexploredFull),supply(7,J,5,0,UnexploredEmpty)],[supply(8,A,7,0,UnexploredEmpty),sup
ply(8,B,5,0,UnexploredEmpty),supply(8,C,8,0,UnexploredEmpty),supply(8,D,3,275,UnexploredFull),supply(
8,E,8,0,UnexploredEmpty),supply(8,F,5,25,UnexploredFull),supply(8,G,10,0,UnexploredEmpty),supply(8,H,
8,0,UnexploredEmpty),supply(8,I,8,0,UnexploredEmpty),supply(8,J,14,0,UnexploredEmpty)],[supply(9,A,8,
0,UnexploredEmpty),supply(9,B,10,0,UnexploredEmpty),supply(9,C,9,0,UnexploredEmpty),supply(9,D,6,0,Un
exploredEmpty),supply(9,E,4,100,UnexploredFull),supply(9,F,9,0,UnexploredEmpty),supply(9,G,17,0,Unexp
loredEmpty),supply(9,H,7,0,UnexploredEmpty),supply(9,I,5,0,UnexploredEmpty),supply(9,J,8,0,Unexplored
Empty)],[supply(10,A,5,0,UnexploredEmpty),supply(10,B,8,0,UnexploredEmpty),supply(10,C,4,0,Unexplored
Empty),supply(10,D,5,0,UnexploredEmpty),supply(10,E,7,0,UnexploredEmpty),supply(10,F,4,120,Unexplored
Full),supply(10,G,6,0,UnexploredEmpty),supply(10,H,3,80,UnexploredFull),supply(10,I,13,0,UnexploredEm
pty),supply(10,J,9,0,UnexploredEmpty)]]
C = 8770.
```