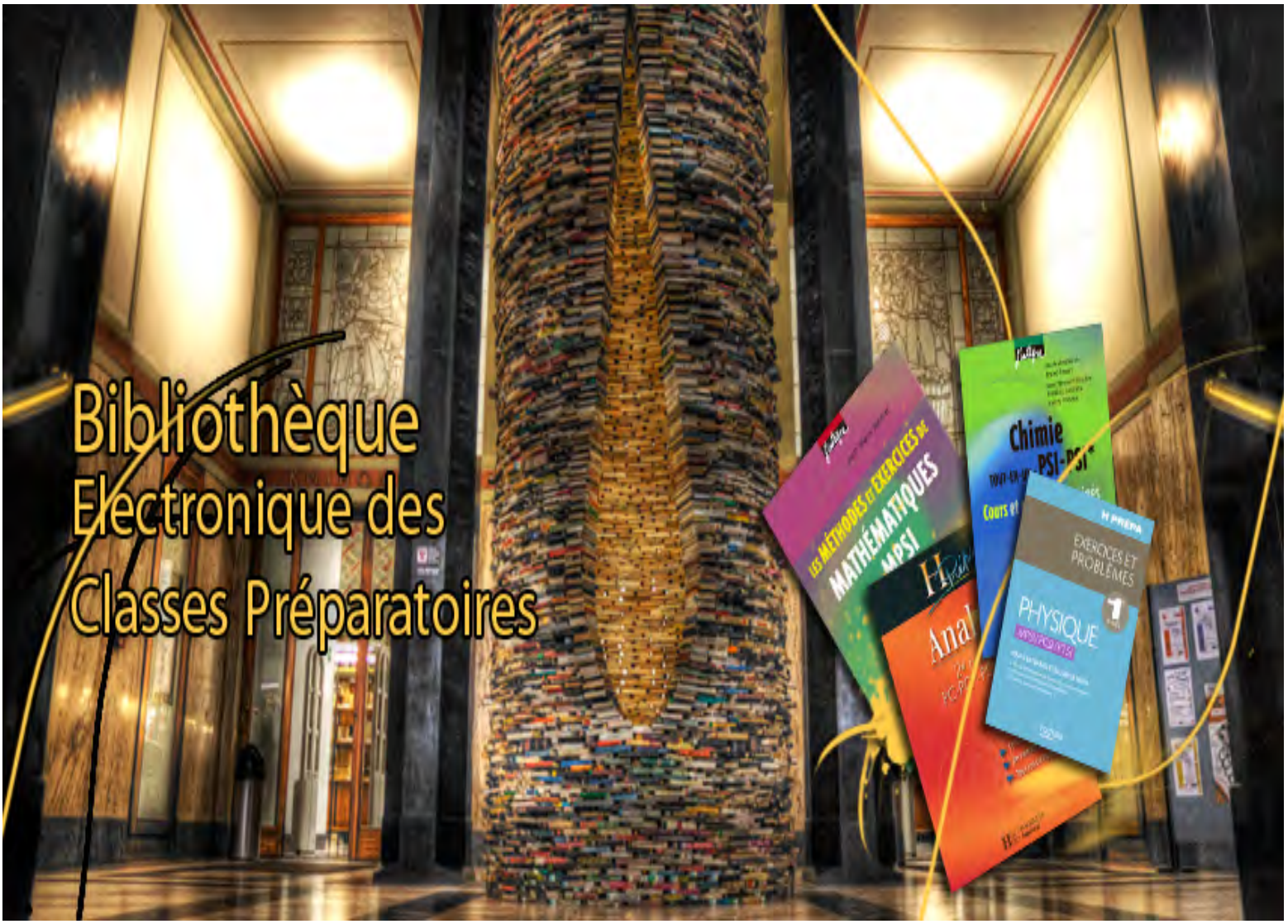


المملكة المغربية

ROYAUME DU MAROC

Ministère de l'Enseignement Supérieur de la Recherche
Scientifique de la Formation des Cadres

Présidence du Concours National Commun
Ecole Hassania des Travaux Publics

CONCOURS NATIONAL COMMUN

d'admission aux Établissements de Formation d'Ingénieurs et Établissements Assimilés

Session 2013

ÉPREUVE D'INFORMATIQUE

Filière **MP**

Durée **2** heures

Cette épreuve comporte 10 pages au format A4, en plus de cette page de garde
L'usage de la calculatrice est **interdit**

L'énoncé de cette épreuve, commune aux candidats des filières MP/ PSI/ TSI, comporte 10 pages.
L'usage de la calculatrice est interdit .

*Les candidats sont informés que la précision des raisonnements **algorithmiques** ainsi que le soin apporté à la rédaction et à la présentation des copies seront des éléments pris en compte dans la notation. Il convient en particulier de rappeler avec précision les références des questions abordées. Si, au cours de l'épreuve, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.*

Remarques générales :

- L'épreuve se compose de deux problèmes indépendants.
- Toutes les instructions et les fonctions demandées seront écrites en **langage C**.
- Les questions non traitées peuvent être admises pour aborder les questions ultérieures.
- Toute fonction peut être décomposée, si nécessaire, en plusieurs fonctions

PROBLÈME I : ANALYSEUR LEXICAL D'UN LANGAGE

✧✧✧

Présentation du problème :

Avant son exécution, tout code source d'un programme informatique doit être traduit en un autre code appelé code machine. Cette traduction est réalisée par un compilateur dont le rôle est de transformer le code source en une suite d'instructions élémentaires directement exécutables par le processeur.

Ainsi le rôle de l'analyse lexicale est d'identifier puis supprimer les caractères superflus du code source (commentaires, espaces,...), de reconnaître les mots-clés, les identificateurs, les opérateurs, *ect* qui sont définies par un ensemble de règles.

En plus, l'analyseur lexical signale les éventuelles erreurs de syntaxe et associe à chaque erreur le numéro de la ligne dans laquelle elle intervient. Dans le problème qui suit, on se propose de mettre en oeuvre un **analyseur lexical.**

Mise en oeuvre d'un analyseur lexical :

Il s'agit d'implémenter des fonctions en langage C pour un analyseur lexical d'un pseudo langage informatique noté **PL**.

Rappels :

Les fonctions suivantes définies dans la bibliothèque du langage **C** peuvent être appelées au besoin sans être définies :

Fonctions définies dans le fichier stdio.h

- int printf(const char* format, ...) : affiche sur l'écran la chaîne en paramètre

- FILE *fopen(char *nom, char *m) : ouvre le fichier nom avec le mode d'ouverture m (m="r" pour la lecture et m="w" pour l'écriture)
-int fgetc(FILE *f) : lit un caractère du fichier texte f et le retourne
- char *fgets(char * ligne, int max, FILE *f) : lit une ligne du fichier texte f, et la met dans la chaîne ligne, max est le nombre maximum de caractères de la ligne. fgets retourne l'adresse de la ligne lue ou NULL à la fin de fichier,

Fonctions définies dans le fichier string.h

- int strlen (const char*) retourne la longueur de la chaine en paramètre
- int strcmp (const char*, const char*) compare les 2 chaînes en paramètres et retourne 0 si elles sont identiques.

A : Gestion des commentaires et des espaces.

Question A-1 : Test d'un Commentaire

Un commentaire est une instruction qui n'est pas traduite par le compilateur. Pour ce pseudo langage PL, un commentaire est une chaîne de caractères qui doit commencer obligatoirement par les 2 caractères * (slash étoile) et se terminer par les 2 caractères *\ (étoile slash).

✍ Écrire le code de la fonction d'entête : **int comment(char instr[ ])** qui retourne **1** si la chaîne **instr** en paramètre est un commentaire de **PL** ou **0** (zéro) sinon.

Exemple : - Soit la chaîne de caractères instr="* explication *\" alors l'appel de la fonction **comment(instr)** retourne 1.

- Si la chaîne de caractères instr="explication" alors l'appel **comment(instr)** retourne 0.

Question A-2- Suppression d'espaces multiples dans une instruction

✍Écrire une fonction d'entête : **void supprim_espaces(char instr[ ])** qui supprime les espaces multiples consécutifs(qui se suivent) entre les caractères de la chaîne **instr** en paramètre. S'il y'a N (1 < N) espaces au début, ou à la fin, ou entre deux caractères de la chaîne **instr**, on ne laissera qu'un seul espace.

Exemple : Soit la chaîne instr=" repeter (max<10) max++;" après l'appel de la fonction **supprim_espaces**(instr), instr=" repeter (max<10) max++;"

B : Reconnaissance des mots-clés et des identificateurs.

Tout langage de programmation définit un ensemble de mots clés qui sont des chaînes de caractères ayant des significations et des utilisations spécifiques pour ce langage.
Pour le pseudo langage **PL**, on suppose avoir déclaré et défini **N** et **tmotscles** tels que :

- **N** est une constante entière strictement positive contenant le nombre de mots clés du pseudo langage **PL**.
- **tmotscles** est un tableau de N **chaines de caractères** contenant tous les mots clés de **PL**

Question B-1 : Vérification d'un mot clé

✍Ecrire la fonction d'entête : **int motcle(char mot[ ])** qui retourne **1** si mot (le paramètre de la fonction) est un mot clé de **PL** ou **0** sinon. Le paramètre **mot** est un **mot clé de PL** si c'est un élément du tableau **tmotscles** (supposé déclaré et défini).

Exemple : Pour N = 5 Si **tmotscles[N][80]={"entier", "reel", "repeter","si", "sinon"** } Alors l'appel **motcle("si")** retourne **1** et l'appel **motcle("pour")** retourne **0**

Question B-2 : validité d'un identificateur

Un identificateur est une chaîne de caractères qui permet de déclarer et d'identifier les éléments d'un programme (les constantes, les variables, ...). Il doit respecter un ensemble de règles particulières pour chaque langage.

Un identificateur du pseudo langage **PL** est valide s'il respecte les quatre conditions (C1, C2, C3 et C4) suivantes :

C1- Sa longueur est inférieure strictement à 80

C2- Ne contient aucun espace

C3- Ne commence pas par un chiffre ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9').

C4- Ne doit pas être un mot clé du langage PL (n'est pas dans **tmotscles**)

✍Ecrire une fonction d'entête **int identificateur(char id[ ])** qui retourne **1** si son paramètre **id** est un identificateur valide de **PL** ou retourne **0** (zéro) sinon.

C : Implémentation de l'analyseur lexical .

On suppose maintenant que les mots clés du pseudo langage **PL** sont mis dans un fichier texte de nom physique "c :\fmotscles.txt"supposé existant. Chaque ligne de ce fichier texte correspond à un mot clé de PL

Question C-1 : Analyse d'une instruction

Une instruction de ce pseudo langage PL est correcte si elle vérifie l'une des deux conditions suivantes (C1 ou C2) :

C1- - L'instruction est un commentaire du pseudo langage **PL**(voir question A-1)

C2- L'instruction commence obligatoirement par un mot clé de **PL** suivi par un espace et se termine par le caractère **;** (point virgule).

✍Écrire la fonction **int analyserInstruction(char instr[ ])** qui retourne **1** si la chaîne **instr** en paramètre correspond à une instruction correcte de **PL** ou **0** sinon.

Exemple : Soit le fichier **"c :\fmotscles.txt"**contenant les 5 lignes suivantes :

entier
reel
repeter
si
sinon

Alors l'appel de la fonction **analyserInstruction("reel x=10 , y;")** retourne 1 et l'appel de la fonction **analyserInstruction("var=250 ;")** retourne **0**

Question C-2 : Analyse d'un fichier source

On suppose avoir écrit un programme avec ce pseudo langage **PL.** Le code source de ce programme est enregistré dans un fichier texte.
Chaque ligne de ce fichier correspond à une instruction de **PL.**
✍ Écrire la fonction d'entête **void analyserSource (char source[ ])** qui analyse le fichier source dont le nom est en paramètre de la fonction . Cette fonction affiche sur l'écran les numéros de toutes les lignes du fichier source qui correspondent à des instructions incorrectes ou affiche sur l'écran **"succès"** dans le cas où toutes les lignes du fichier sources sont des instructions correctes.

Exemple : Soit le fichier **"c :\fmotscles.txt"**contenant les 5 mots clés de PL suivantes :

```
entier
reel
repeter
si
sinon
```

Et soit le fichier de nom **"sourcePL"** contenant les 6 lignes suivantes du code source d'un programme écrit en pseudo langage **PL** :

```
reel x=0,y=1 ;
tantque ( x<10) x=x+1 ;
\* instruction *\
si ( y==0) alors afficher(NUL) ;
sinon afficher(Non NUL)
fin ;
```

Après l'appel de la fonction **analyserSource("sourcePL")** , on aura sur l'écran : Les numéros des lignes correspondantes à des instructions incorrectes sont : 2 , 5 , 6

PROBLÈME II : DÉTERMINATION DES ITINÉRAIRES

✧✧✧

Contexte du problème :

Dans le monde actuel, il est très utile de pouvoir localiser géographiquement des personnes ou des objets aussi bien pour des besoins privés que professionnels.
En effet, au niveau personnel, la localisation géographique ou géo-localisation peut être d'une grande assistance dans certains cas (Suivi et sécurité des personnes en situation d'urgence...).

Au niveau professionnel, la géo-localisation est très utilisée notamment pour la navigation routière mais aussi dans plusieurs autres domaines (Transport, logistique, énergie, ...) favori-

sant ainsi un gain de productivité et une sécurité accrue.

Vus ces besoins, et avec le développement fulgurant des technologies de l'informatique et des télécommunications, le déploiement d'applications de géo-localisation est en plein essor. En parallèle, nous assistons aussi à une émergence d'équipements dont certains comme le système de navigation automobile par GPS sont très largement diffusés.

Ces systèmes de géo-localisation permettent le repérage de la position géographique exacte de leurs usagers, ainsi que le calcul d'itinéraires.

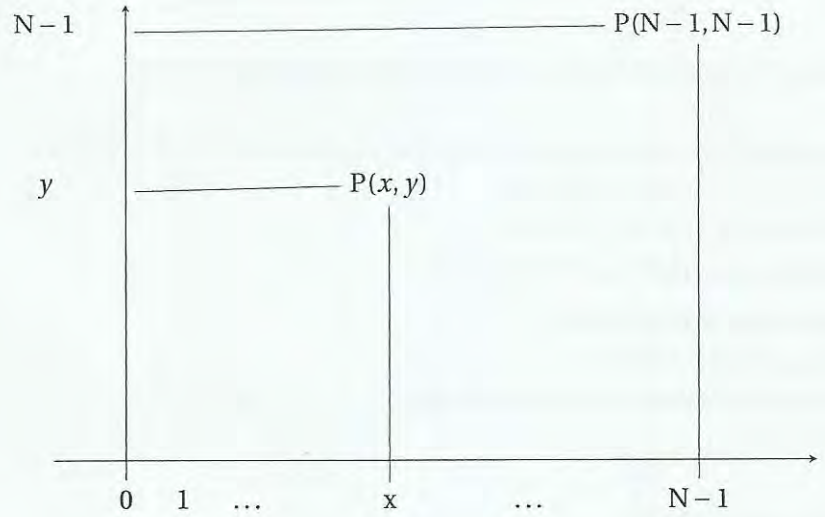
C'est dans cette optique que s'inscrit le problème suivant concernant des algorithmes de détermination de chemins et d'itinéraires entre les points d'un plan.

Implémentation de fonctions de détermination d'itinéraires

Notations :

- Soit $\mathscr{P}$ un plan, on notera $P(x, y)$, un point de coordonnées x et y dans ce plan.

-Soit **N** une constante entière strictement positive. On notera $\mathscr{P}(N)$, l'ensemble des points du plan $\mathscr{P}$ ayant des coordonnées x et y entières positives telles que $(0 <= x < N)$ et $(0 <= y < N)$.



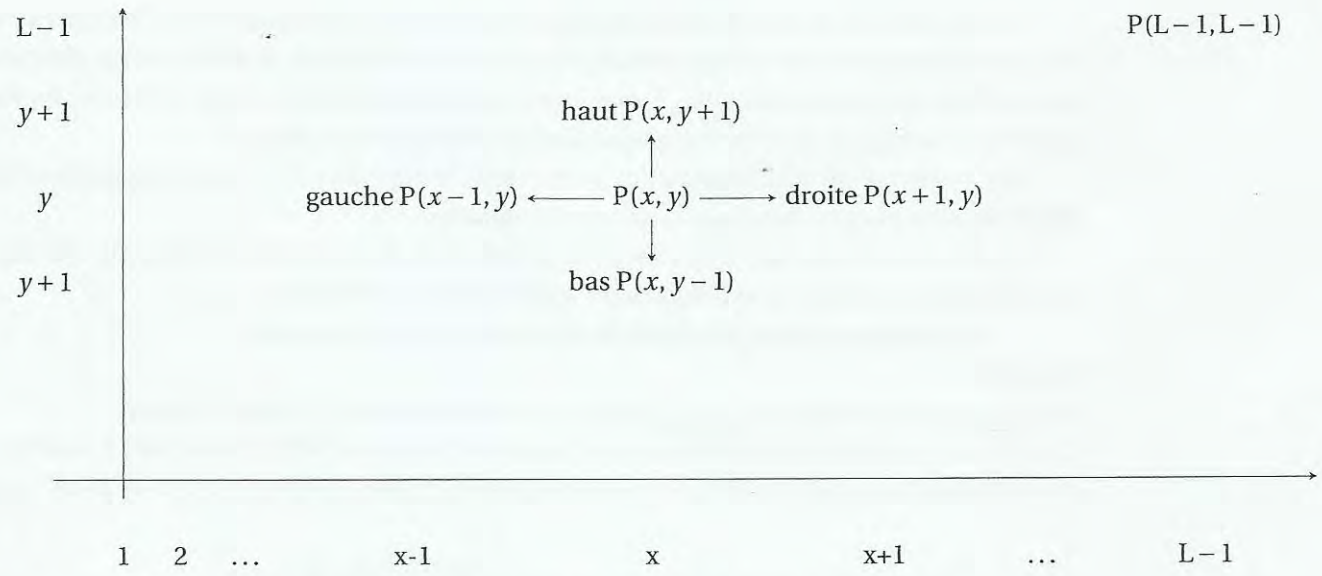
- Soient A et B deux points distincts de $\mathscr{P}(N)$. On appelle chemin de A vers B, un ensemble **C**, non vide et ordonné de points distincts (tous différents) de $\mathscr{P}(N)$ tel que : $\mathbf{C} = \{A = P(x_0, y_0), P(x_1, y_1), \ldots, P(x_i, y_i), P(x_{i+1}, y_{i+1}), \ldots, P(x_k, y_k) = B\}$. Chaque point de ce chemin $P(x, y)$ est relié au point suivant $P(s, t)$ par l'une de ces 4 relations de direction (gauche, droite, haut, bas) définies ci dessous :

gauche : $P(s, t)$ est à gauche de $P(x, y)$ et dans ce cas $s = x - 1$ et $t = y$

droite : $P(s, t)$ est à droite de $P(x, y)$ dans le cas où $s = x + 1$ et $t = y$

haut : $P(s, t)$ est au dessus de $P(x, y)$ dans le cas où $s = x$ et $t = y + 1$

bas : $P(s, t)$ est en dessous de $P(x, y)$ dans le cas où $s = x$ et $t = y - 1$



Remarque : Il peut exister plusieurs chemins différents de A vers B dans $\mathscr{P}$(N)

Exemple :

Soit A et B deux points de $\mathscr{P}$(10) (dans ce cas **N** = 10) tels que : A = P(2,3) et B = P(5,5)

$\mathbf{C_1}$ = {P(2,3), P(3,3), P(4,3), P(4,4), P(4,5), P(5,5)}

$\mathbf{C_2}$ = {P(2,3), P(2,4), P(3,4), P(3,5), P(4,5), P(5,5)}

sont deux exemples de chemins différents de A vers B.

Déclarations et suppositions préliminaires :

Pour toutes les questions de ce problème :

-On suppose avoir **déclaré et définie** une constante strictement positive **N**

- On suppose avoir déclaré le type point ainsi :

```
typedef struct
        { int x; //abscisse du point
          int y; // ordonné du point
        } point;
```

Ce problème est composé de deux parties A et B indépendantes :

A : Construction de chemins en utilisant des tableaux .

Dans cette partie, on représentera tout chemin d'un point **A** de $\mathscr{P}$(**N**) vers un point **B** de $\mathscr{P}$(**N**) par un tableau d'éléments de type **point**.

- Soit **Max** une constante positive (supposée déclarée et définie) contenant le nombre maximum de points que peut contenir un chemin entre 2 points quelconques distincts de $\mathscr{P}$(**N**)

- Soit **C** un tableau de **Max** points déclaré ainsi : **point C[Max] ;**

Question A-1 : Initialisation du tableau C

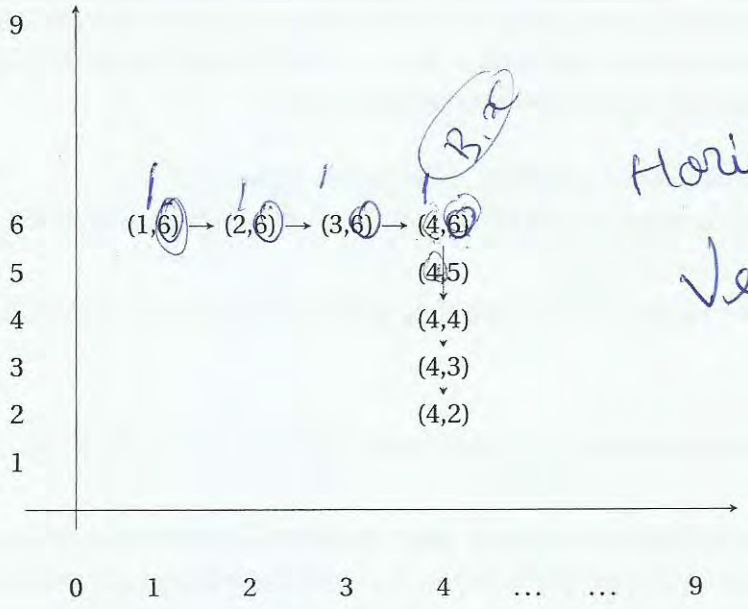
✍ Écrire une fonction d'entête : **void initialiserC()** qui permet d'initialiser tous les points du tableau **C** par les points **P(-1,-1)** (Les points **P(-1,-1)** n'appartiennent pas à $\mathscr{P}(\mathbf{N})$) mais permettent juste d'initialiser le tableau **C**.

Question A-2- Chemin horizontal puis vertical

Soient **A** et **B** deux points de $\mathscr{P}(\mathbf{N})$, on appelle **CheminHV** de **A** vers **B** le chemin construit de telle sorte à se déplacer à partir de A **horizontalement** (soit à gauche, soit à droite) jusqu'à arriver au point ayant la même abscisse que **B**, puis se déplacer ensuite **verticalement** (soit en haut, soit en bas) jusqu'à arriver au point **B**.

Exemple :

Soit **N = 10**, soient les points **A=P(1,6)** et **B=P(4,2)** de $\mathscr{P}(10)$, alors le **CheminHV** de **A** vers **B** sera représenté par l'ensemble ordonné suivant :
{ P(1,6), P (2,6) , P (3,6) , P(4,6) , P (4,5) , P(4,4), P (4,3), P(4,2) } (voir la représentation graphique de ce chemin ci dessous)



✍ Écrire la fonction d'entête : **void cheminHV(point A, point B)** qui crée le chemin **CheminHV** de **A** vers **B** décrit plus haut. Ce chemin sera représenté dans le tableau **C**. Pour cela le premier élément du tableau **C**, **C[0]** contiendra le point **A** , le deuxième élément **C[1]** le point suivant du **CheminHV** , et ainsi de suite jusqu'au point **B**, les autres éléments du tableau **C** seront initialisés à **P(-1,-1)**.

Exemple :

Soit **N=20**, soient les points **A=P(5,2)** et **B=P (0,4)** de $\mathscr{P}(20)$, alors après l'appel de la fonction **cheminH (A,B)**, on aura le tableau **C** défini ainsi : **C[0]=P(5,2), C[1]=P(4,2) ; C[2]=P(3,2) ; C[3]=**

P(2,2) ; C[4]= P(1,2) ; C [5]=P(0,2) ; C[6]=P(0,3) ; C[7]=P (0,4), C [8]=P(-1,-1), C[9]=P(-1,-1), ...,C[Max-1]=P(-1,-1).

Question A-3 : Chemin optimal

- Soient **A** et **B** deux points distincts et quelconques de $\mathscr{P}$(**N**)
- Soit **C** un chemin défini de **A** vers **B**. On appelle distance du chemin **C** le nombre de points du chemin **C** différents du point **A**.

Exemple :

soient **A** et **B** deux points de $\mathscr{P}$(**10**) définis ainsi **A=P(2,6)** et **B=P(4,3)** et soit le chemin **C** de **A** vers **B** suivant :

C ={P(2,6) ; P(2,5) ;P(3,5) ; P(3,4) ; P(4,4) ;P(4,3)} ; alors la distance du chemin **C** est **5**

- Soit **NC**(**0 < NC**) le nombre de chemins distincts de **A** vers **B**. Considérons le tableau **tabC** déclaré ainsi : **point tabC[NC][Max]**. Ce tableau est destiné à représenter tous les chemins distincts de **A** vers **B** comme suit :

chaque ligne i (0 <= i < NC) du tableau **tabC** (**tabC[i][j], 0<=j<Max**) représente le chemin numéro **i** de **A** vers **B** tel que : **tabC[i][0]** représente le point **A** (premier point du chemin numéro **i**) , **tabC[i][1]** représente le deuxième point du chemin numéro **i**, et ainsi de suite jusqu'au point **B** (dernier point du chemin numéro **i**) représenté par **tabC[i][k] (0<=k<Max).** Tous les autres points (**tabC[i][j], j= k+1, ..., Max-1**) représenteront les points **P(-1,-1)** et ne sont pas des points significatifs du chemin numéro i.

Exemple :

Soit A=P(6,2) et B=P(3,4) de $\mathscr{P}$(10) , alors par exemple :

- le premier chemin ($i = 0$) est $\{P(6,2), P(6,3), P(5,3), P(4,3), P(4,4), P(3,4), P(-1,-1), \ldots, P(-1,-1)\}$

- le deuxième chemin ($i = 1$) est $\{P(6,2), P(5,2), P(4,2), P(3,2), P(3,3), P(3,4), P(-1,-1), \ldots, P(-1,-1)\}$

$\vdots \quad \vdots \qquad\qquad\qquad \vdots$

- le chemin numéro NC ($i = NC - 1$) est $\{P(6,2), \ldots, P(3,4), P(-1,-1), \ldots, P(-1,-1)\}$

Pour les questions suivantes **A -3-a** et **A -3-b**, on suppose avoir déclaré et défini :

- Deux variables **A** et **B** de type point (qui représentent deux points distincts de $\mathscr{P}$(**N**)
- La constante entière **NC** qui contient le nombre de chemins distincts de **A** vers **B**
- Le tableau **tabC** qui représente tous les chemins de **A** vers **B** comme indiqués ci dessus

Question A -3-a : distance d'un chemin

✍En considérant le tableau **tabC** déclaré et défini plus haut, écrire la fonction d'entête **int distance (int num)** qui retourne la distance du chemin numéro **num** de **A** vers **B** du tableau **tabC**(on suppose que **0<=num<NC**)

Question A-3-b : Distance minimale entre 2 points

✍En considérant le tableau **tabC**, écrire la fonction d'entête **int distancemin** () qui retourne la distance minimale de tous les chemins de **A** vers **B** définis dans le tableau **tabC**.

B : Construction d'itinéraires en utilisant des listes chaînées .

Dans cette partie, on représentera tout chemin d'un point **A** de $\mathscr{P}$(**N**) vers un point **B** de $\mathscr{P}$(**N**) par une liste chaînée de points.

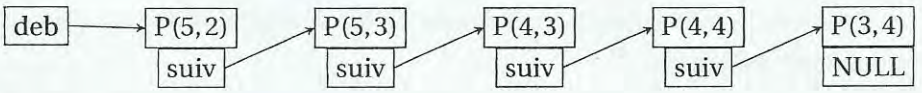
Pour toutes les questions de cette partie, on suppose avoir déclaré le type **cheminListe** suivant :

```
typedef struct typechemin
        { point p;
        struct typechemin * suiv;
        } cheminListe;
```

- On représentera tout chemin d'un point **A** de $\mathscr{P}$(**N**) vers un point **B** de $\mathscr{P}$(**N**) par une liste chaînée d'éléments de type **cheminListe** définie par l'adresse de son premier élément.

Exemple :

Soient les points **A=P(5,2)** et **B=P(3,4)** de $\mathscr{P}$(**10**), et un chemin **C** de **A** vers **B** défini ainsi **C ={ P(5,2), P(5,3) , P (4,3) , P(4,4) , P (3,4) }**. Ce chemin **C** peut être représenté par **deb** l'adresse du premier élément de la liste chaînée de type **cheminListe** suivante :



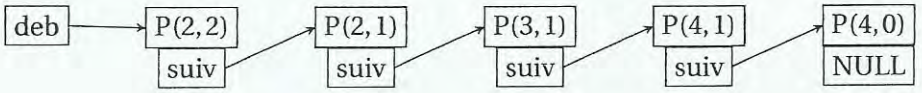
Question B-1- Chemin de retour

Soient **A** et **B** deux points de $\mathscr{P}$(**N**) . On suppose avoir défini une fonction de prototype **cheminListe *AversB()** ; qui définit un chemin quelconque de **A** vers **B** sous forme d'une liste chaînée de type **cheminListe** et retourne l'adresse de son premier élément,

✍En utilisant l'appel de la fonction **AversB()**, écrire la fonction d'entête **cheminListe *BversA()** qui crée le chemin de **B** vers **A** qui est le chemin inverse du chemin **AversB()** et retourne l'adresse de son premier élément.

Exemple :

Soient les points **A=P(4,0)** et **B=P(2,2)**, si l'appel de la fonction **AversB()** crée une liste chaînée représentant le chemin **{P(4,0), P(4,1) ; P(3,1), P(2,1), P(2,2)}** Alors l'appel de **BversA()** crée une liste chaînée représentant le chemin suivant **{P(2,2) ; P(2,1), P(3,1), P(4,1), P(4,0)}** et retourne **deb** l'adresse du premier élément.

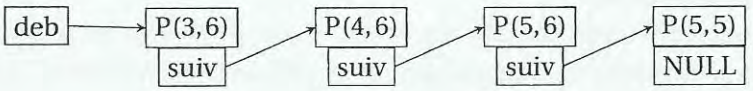


Question B-2- Affichage des points d'un chemin

✍ Écrire la **fonction récursive** d'entête **void afficherChemin(cheminListe *deb)** qui permet d'afficher sur l'écran et dans l'ordre, les coordonnées des points constituant le chemin représenté en paramètre par l'adresse **deb** (adresse du premier élément de la liste chaînée).

Exemple :

Si on a la liste chaînée suivante :



Alors l'appel de la fonction **afficherChemin(deb)** va afficher sur l'écran :

P(3,6) , P(4,6) , P(5,6) , P(5,5)

Question B-3- Itinéraire passant par un repère

Soient **A , B** et **R** 3 points distincts de $\mathscr{P}$**(N)** , il s'agit de créer un chemin de **A** vers **B** passant par le point **R**.

✍ Écrire la fonction **cheminListe *cheminRepere(point A, point B, point R)** qui permet de créer un chemin de **A** vers **B** passant par le point repère **R** . Cette fonction doit retourner l'adresse du premier élément de ce chemin.

Rappel : Tous les points d'un chemin sont distincts.

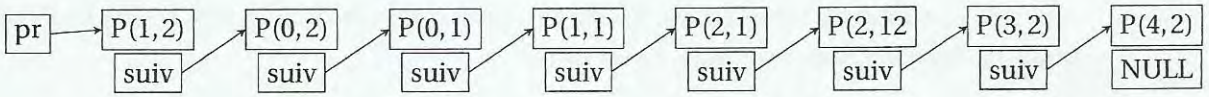
Remarque : Plusieurs chemins sont possibles. Il est demandé de créer et retourner un seul.

Exemple :

Soient les points **A, B** de $\mathscr{P}$**(10)** telle que **A=P(1,2) ; B= P(4,2)**

Si le point **R=P(0,1)**, alors l'appel de **cheminRepere(A,B,R)** peut retourner par exemple l'adresse **pr** du chemin **C** suivant :

C={P(1,2) ; P(0,2), P(0,1), P(1,1), P(2,1), P(2,2) ; P(3,2), P(4,2)}



✧✧✧

FIN DE L'ÉPREUVE