ÉCOLE POLYTECHNIQUE MONTRÉAL

Le génie sans frontières

Solutionnaire examen final

INF2010

Sigle du cours

Identification de l'étudiant(e)		
Nom :	**Prénom :**	
Signature :	**Matricule :**	**Groupe :**

Sigle et titre du cours	*Groupe*	*Trimestre*
INF2010 - Structures de données et algorithmes	Tous	A2008
Professeur	***Local***	***Téléphone***
Tarek Ould Bachir (gr. 1) Ettore Merlo (gr 2)	A-416	5758

Jour	*Date*	*Durée*	*Heures*
Jeudi	18 décembre 2008	2h30	9h30

Documentation	*Calculatrice*	
☐ Toute	☐ Aucune	**Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.**
☒ Aucune	☐ Programmable	
☐ Voir directives particulières	☒ Non programmable	

Directives particulières

Bonne chance à tous!

Important

Cet examen contient 6 questions sur un total de 12 pages (excluant cette page)

La pondération de cet examen est de 40 %

Vous devez répondre sur : ☒ le questionnaire ☐ le cahier ☐ les deux

Vous devez remettre le questionnaire : ☒ oui ☐ non

Question 1 : Graphes acycliques (15 points)

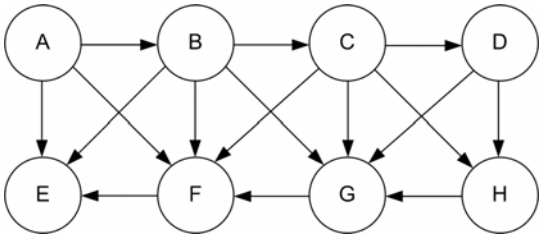
On vous présente un ensemble de graphes dirigés. Pour chacun des graphes qui suivent :

Indiquer si le graphe est cyclique

1.) Si c'est le cas, indiquer au moins un cycle

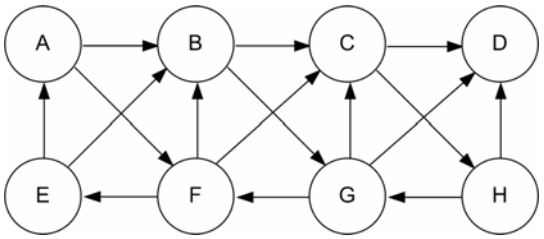
2.) Sinon, numéroter les nœuds de **A** à **H** de sorte que, si il existe un chemin du noeud **i** au nœud **j**, alors **i<j**.

a) Cyclique : OUI [ ] NON [X]



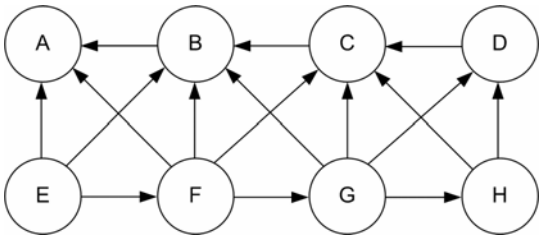
	Indegree avant la sortie de file							
Noeud	1	2	3	4	5	6	7	8
A	0	0	0	0	0	0	0	0
B	1	0	0	0	0	0	0	0
C	1	1	0	0	0	0	0	0
D	1	1	1	0	0	0	0	0
E	3	2	1	1	1	1	1	0
F	4	3	2	1	1	1	0	0
G	4	4	3	2	1	0	0	0
H	2	2	2	1	0	0	0	0
Entrée	A	B	C	D	H	G	F	E
Sortie	A	B	C	D	H	G	F	E

b) Cyclique : OUI [X] NON [ ] A -> F -> E -> A



	Indegree avant la sortie de file							
Noeud	1	2	3	4	5	6	7	8
A								
B								
C								
D								
E								
F								
G								
H								
Entrée								
Sortie								

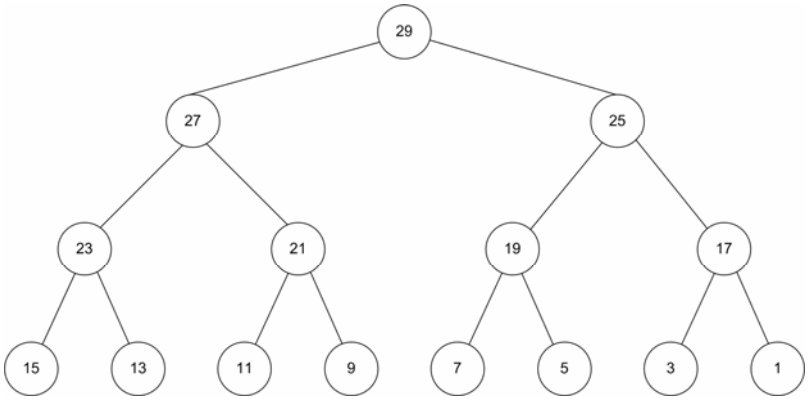
c) Cyclique : OUI [ ] NON [X]



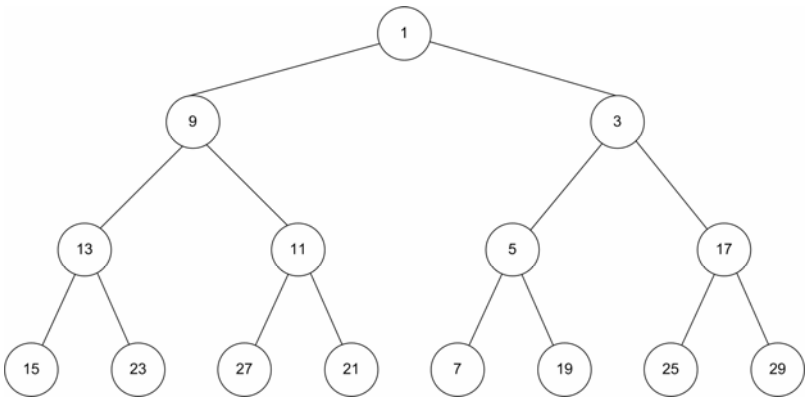
	Indegree avant la sortie de file							
Noeud	1	2	3	4	5	6	7	8
A	3	2	1	1	1	1	1	0
B	4	3	2	1	1	1	0	0
C	3	3	2	1	1	0	0	0
D	2	2	2	1	0	0	0	0
E	0	0	0	0	0	0	0	0
F	1	0	0	0	0	0	0	0
G	1	1	0	0	0	0	0	0
H	1	1	1	0	0	0	0	0
Entrée	E	F	G	H	D	C	B	A
Sortie	E	F	G	H	D	C	B	A

Question 2 : Monceaux (20 points)

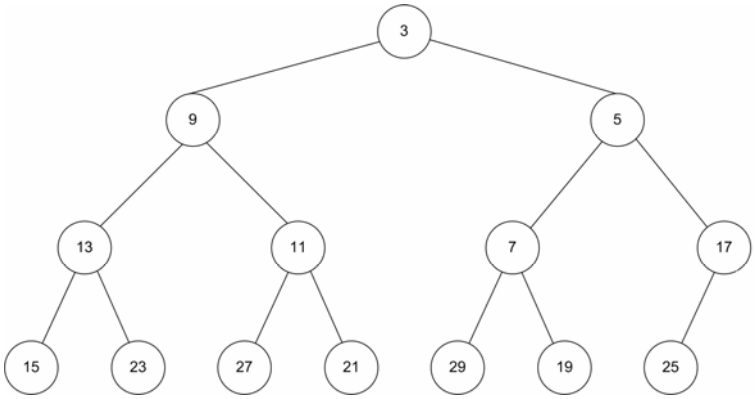
a) Construire, selon la technique vue dans le cours, un monceau à partir de l'arbre binaire suivant :



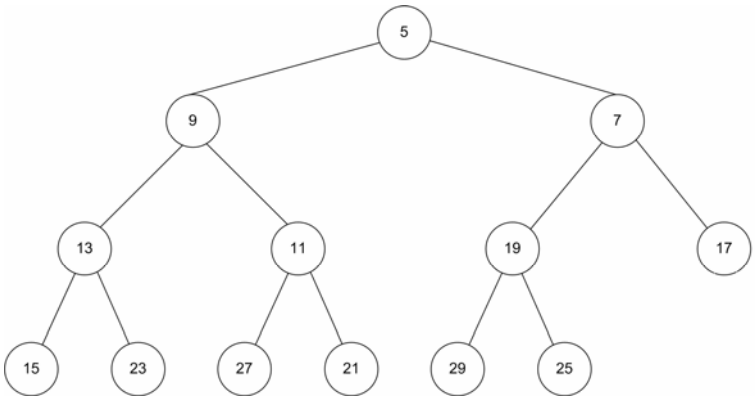
Monceau :



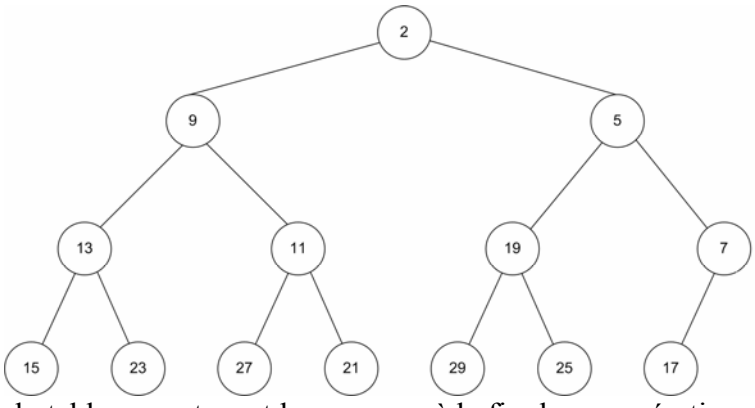
b) Dessiner l'état du monceau ainsi obtenu après l'appel à `deleteMin()` :



c) Dessiner l'état du monceau ainsi obtenu après l'appel à `deleteMin()` :



d) Ajouter au monceau ainsi obtenu un nœud dont la clé est 2 :

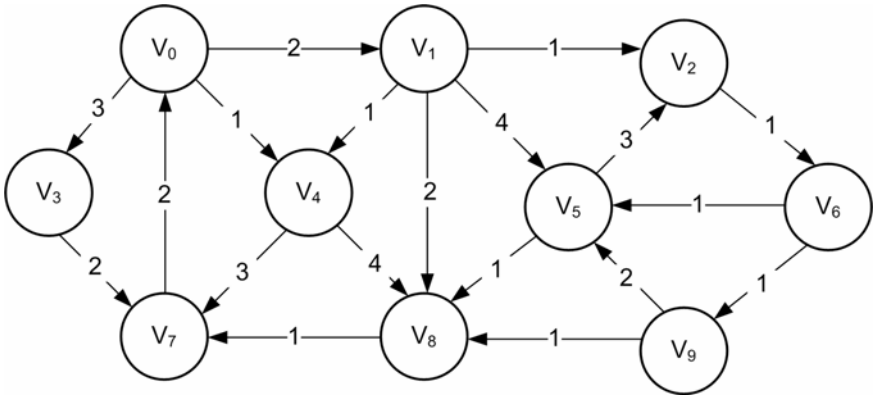


e) Dessiner l'état du tableau contenant le monceau à la fin de ces opérations :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	2	9	5	13	11	19	7	15	23	27	21	29	25	17	

Question 3 : Le plus court chemin dans un graphe (15 points)

En appliquant l'algorithme de Dijkstra utilisant une file de priorité (ici est partiellement réalisé), trouver la longueur du plus court chemin menant à chacun des nœuds du graphe en partant de V_0.



a) Continuer l'exécution de l'algorithme en vous référant à l'état de la file de priorité

Nœud	Connu	Dist min.	Parent
V_0	√	0	
V_1	√	∞, 2	V_0
V_2	√	∞, 3	V_1,
V_3	√	∞, 3	V_0
V_4	√	∞, 1	V_0
V_5	√	∞, 6, 5	V_1, V_6,
V_6	√	∞, 4	V_2,
V_7	√	∞, 4	V_4
V_8	√	∞, 5, 4	V_4, V_1
V_9	√	∞, 5	V_6,

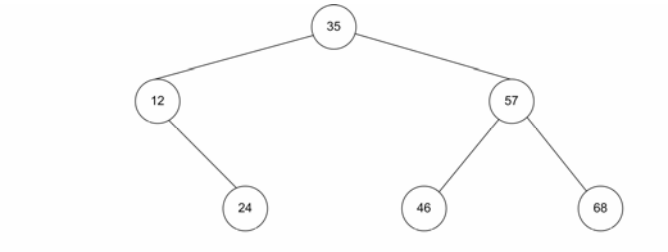
File de priorité
(V_3, 3)
(V_2, 3)
(V_7, 4)
(V_8, 4)
(V_6, 4)
(V_5, 5)
(V_9, 5)

b) Détailler chacun des chemins les plus courts trouvés :

Nœud	Le plus court chemin	Distance parcourue
V_1	V_0, V_1	2
V_2	V_0, V_1, V_2	3
V_3	V_0, V_3	3
V_4	V_0, V_4	2
V_5	V_0, V_1, V_2, V_6, V_5	5
V_6	V_0, V_1, V_2, V_6	4
V_7	V_0, V_4, V_7	4
V_8	V_0, V_1, V_8	4
V_9	V_0, V_1, V_2, V_6, V_9	5

Question 4 : AVL (15 points)

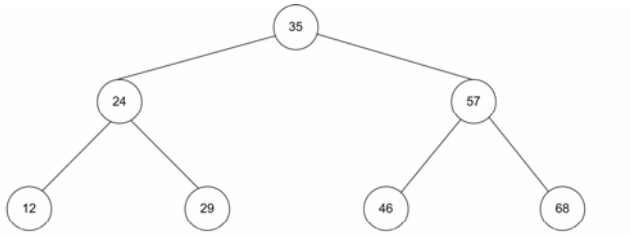
En considérant l'arbre AVL suivant :



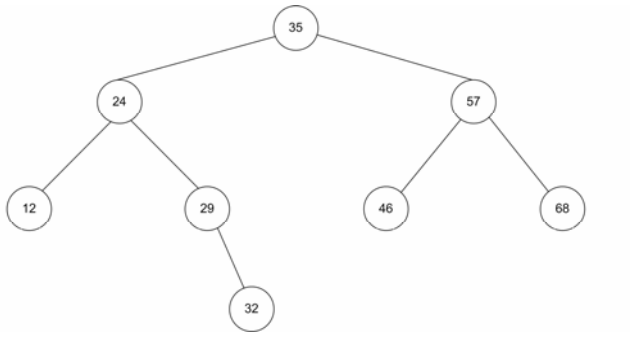
Effectuer l'ensemble des opérations suivantes dans l'ordre en vous servant des arbres ci-bas :

a) Insérer 29
b) Insérer 32
c) Insérer 51
d) Insérer 34
e) Insérer 33
f) Insérer 49

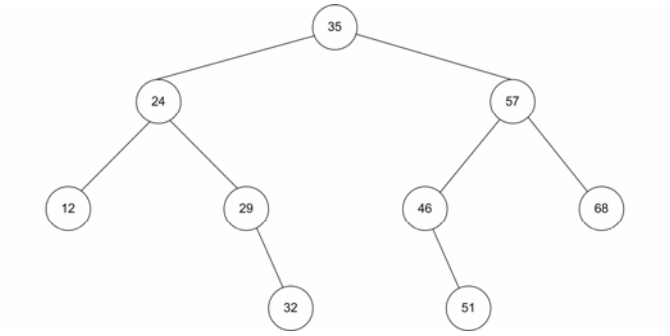
a) Insérer 29



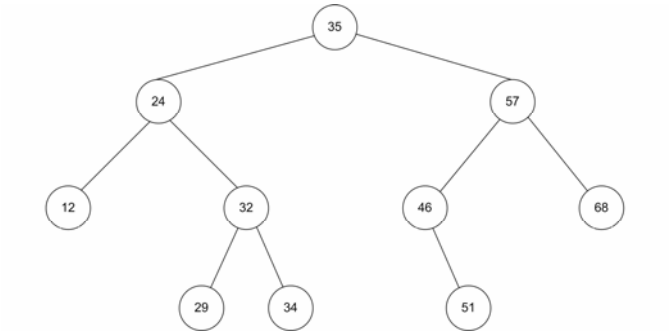
b) Insérer 32



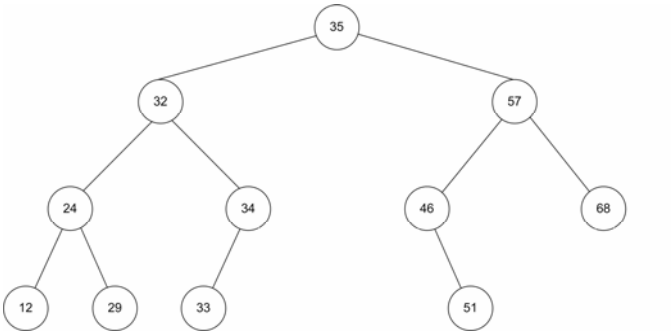
c) Insérer 51



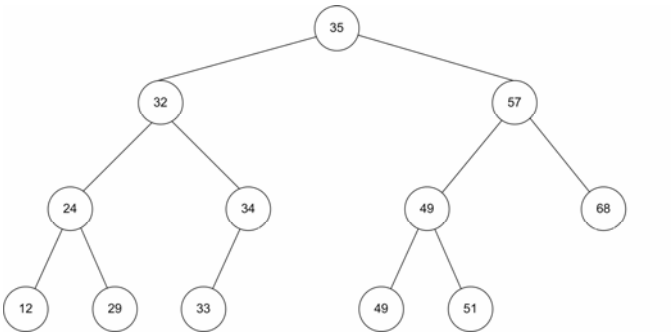
d) Insérer 34



e) Insérer 33



f) Insérer 49



Question 5 : Rabin-Karp (20 points)

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. La chaîne de caractères recherchée est alors remplacée par un nombre qu'il faut pré-calculer et on compare toute les valeurs des sous-séquences du texte à la valeur pré-calculée. Les différentes implémentations (**code java**) discutées dans cette question vous sont données â l'**annexe I**.

Dans ces implémentations de Rabin-Karp, le texte et le patron ne contiennent que les caractères numériques '0'-'9' dans l'encodage ASCII. Les valeurs respectives de ces caractères sont données ci-dessous :

Caractère	Val. ASCII	Caractère	Val. ASCII
'0'	48	'5'	53
'1'	49	'6'	54
'2'	50	'7'	55
'3'	51	'8'	56
'4'	52	'9'	57

On vous propose trois méthode de calcule de la valeur numérale associée aux sous-chaînes :

Méthode	Description
#1	Polynôme dans la base d = 58; ex : P = '123' p= (49*58+50)*58 + 51 = 167787
#2	Polynôme dans la base d = 10; ex : P = '123' p = (49*10+50)*58 +51 =5451
#3	Polynôme modulaire dans la base d = 58 modulo 10; ex : P= '123' p = 167787%10=7

a) Donner les avantages (au moins un) et les désavantages (au moins un) que vous voyez à utiliser chacune de ces méthodes :

Méthode	Avantages	Désavantages
#1	Pas de faux positifs	Résultat trop gros pouvant provoquer un débordement
#2	Résultat de valeur modérée	Beaucoup de faux positifs
#3	Résultat modéré, facile à calculer	Peut causer quelques faux-positifs

On vous propose la méthode de calcul suivante (option par défaut dans le code) :

Polynôme modulaire dans la base d = 10 modulo 11

Par exemple, P = '123'

p = ((((49%11)*10 + 50) % 11) * 10 + 51) % 11
= ((((5)*10 + 50) % 11) * 10 + 51) % 11
= (((100) % 11) * 10 + 51) % 11
= ((1) * 10 + 51) % 11
= (61) % 11
= 6

b) Donner la valeur de p pour le patron P = '32123'

7

c) Retrouver tous les décalages donnant la présence de P dans le texte T= '3212323212321'

Décalage (s)	Sous-chaîne	Valeur du polynôme	Égalité?	Faux positif?	Correspondance?
0	'32123'	7	√		√
1	'21232'	6			
2	'12323'	7	√	√	
3	'23232'	4			
4	'32321'	7	√	√	
5	'23212'	6			
6	'32123'	7	√		√
7	'21232'	6			
8	'12321'	5			

Faux positifs trouvés :

2 et 4

Décalages retournés :

0 et 6

Question 6 : DP-Matching (15 points)

En utilisant le tableau suivant, retrouver la plus longue sous-séquence commune aux chaînes d'entrée : X = 'CTGAATGACTAG' et Y = 'CATAGTCACTAG'

	Y	**C**	A	**T**	A	**G**	**T**	C	**A**	**C**	**T**	**A**	**G**
X	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	**D, 1**	**G, 1**	G, 1	G, 1	G, 1	G, 1	D, 1	G, 1	D, 1	G, 1	G, 1	G, 1
T	0	H, 1	H, 1	**D, 2**	**G, 2**	G, 2	D, 2	G, 2	G, 2	G, 2	D, 2	G, 2	G, 2
G	0	H, 1	H, 1	H, 2	H, 2	**D, 3**	G, 3	G, 3	G, 3	G, 3	G, 3	G, 3	D, 3
A	0	H, 1	D, 2	H, 2	D, 3	**H, 3**	H, 3	H, 3	D, 4	G, 4	G, 4	D, 4	G, 4
A	0	H, 1	D, 2	H, 2	D, 3	**H, 3**	H, 3	H, 3	D, 4	H, 4	H, 4	D, 4	H, 4
T	0	H, 1	H, 2	D, 3	H, 3	H, 3	**D, 4**	**G, 4**	H, 4	H, 4	D, 5	G, 5	G, 5
G	0	H, 1	H, 2	H, 3	H, 3	D, 4	H, 4	**H, 4**	H, 4	H, 4	H, 5	H, 5	D, 6
A	0	H, 1	D, 2	H, 3	D, 4	H, 4	H, 4	H, 4	**D, 5**	G, 5	H, 5	D, 6	H, 6
C	0	D, 1	H, 2	H, 3	H, 4	H, 4	H, 4	D, 5	H, 5	**D, 6**	G, 6	H, 6	H, 6
T	0	H, 1	H, 2	D, 3	H, 4	H, 4	D, 5	H, 5	H, 5	H, 6	**D, 7**	G, 7	G, 7
A	0	H, 1	D, 2	H, 3	D, 4	H, 4	H, 5	H, 5	D, 6	H, 6	H, 7	**D, 8**	G, 8
G	0	H, 1	H, 2	H, 3	H, 4	D, 5	H, 5	H, 5	H, 6	H, 6	H, 7	H, 8	**D, 9**

Longueur de la plus longue sous-séquence commune :

9

Plus longue sous-séquence commune :

CTGTACTAG

Annexe I

```
import java.util.ArrayList;


public class RabinKarp {

	public static ArrayList<Integer>
			RabinKarpFind(String Text, String Pattern)
	{
	  ArrayList<Integer> decalages = new ArrayList<Integer>();

	  if( Text.length() < Pattern.length() )
	    return decalages;

	  int p = ComputePatternValue( Pattern );

	  for(int i=0; i <= Text.length() - Pattern.length(); i++)
	  {
	    int t = ComputePatternValue(
	                Text.substring( i, i+Pattern.length() )
	                );

	    if( t == p)
	    {
	          int j;
	          for(j=0; j< Pattern.length(); j++)
	          {
	                  if( Pattern.charAt( j ) != Text.charAt( i +j ) )
	                  {
	                          break;
	                  }
	          }

	          if(j == Pattern.length())
	          {
	                  decalages.add( i );
	                  System.out.println("Correspodance à " + i);
	          }
	                  else
	          {
	                  System.out.println("Faux positif à " + i);
	          }

	    }

	  }

	  return decalages;
	}

	public static int ComputePatternValue(String Pattern)
	{
	  return ComputePatternValue(Pattern, 0);
	}
```

```
public static int ComputePatternValue(String Pattern, int method)
{
  // TODO Auto-generated method stub
  int p = 0;

  switch( method )
  {
  case 1:
   for(int i=0; i<Pattern.length(); i++)
   {
         p *=58;
         p += (int) Pattern.charAt( i );
   }
  break;

  case 2:
   for(int i=0; i<Pattern.length(); i++)
   {
         p *=10;
         p += (int) Pattern.charAt( i );
   }
  break;

  case 3:
   for(int i=0; i<Pattern.length(); i++)
   {
         p *=58;
         p += (int) Pattern.charAt( i );
         p %= 10;
   }
  break;

  default:
   for(int i=0; i<Pattern.length(); i++)
   {
         p *=10;
         p += (int) Pattern.charAt( i );
         p %= 11;
   }

  }

  return p;
}

public static void main(String[] args)
{
  // TODO Auto-generated method stub
  String Pattern = "123";
  int p = ComputePatternValue(Pattern, 1);
  System.out.println( p );

  p = ComputePatternValue(Pattern, 2);
  System.out.println( p );

  p = ComputePatternValue(Pattern, 3);
  System.out.println( p );
```

```
        p = ComputePatternValue(Pattern);
        System.out.println( p );

        Pattern = "32123";
        p = ComputePatternValue(Pattern);
        System.out.println( p );

        String Text = "3212323212321";
        ArrayList<Integer> decalages = RabinKarpFind(Text, Pattern);

        for(int s : decalages )
        {
         System.out.print( s + "\t" );
        }
      }
}
```