```java
import java.io.* ;

class Monnaie {

  // Cardinal d'un paiement
  static int card(int [] t) {
    int r = 0 ;
    for (int i = 0 ; i < t.length ; i++)
      r += t[i] ;
    return r ;
  }

  static int [] glouton(int [] system, int m) {
    int k = system.length ;
    int [] r = new int[k] ;
    for (int i = 0 ;  m > 0 ; i++) {
      int d = system[i] ;
      r[i] = m/d ;
      m %= d ;
    }
    return r ;
  }

  /*****************************************/
  /* Afficher tous les paiements possibles */
  /*****************************************/

  static void dump(PrintStream out, int [] system, int [] t) {
    for (int i = 0 ; i < t.length ; i++) {
      int d = system[i] ;
      for (int j = 0 ; j < t[i] ; j++)
        out.print(" " + d) ;
    }
    out.println() ;
  }

  private static void zyva (int d[], int m, int i, int [] r) {
    if (d[i] == 1) {
      r[i] = m ;
      dump(System.out, d, r) ;
    } else {
      int di = d[i] ;
      int max = m/d[i] ;
      for (int j = 0 ; j <= max ; j++) {
        r[i] = j ;
        zyva(d, m-j*di, i+1, r) ;
      }
    }
  }

  static void tous(int [] system, int m) {
    zyva(system, m, 0, new int[system.length]) ;
  }
```

```
/************************************/
/* Optimal par recherche exhaustive */
/************************************/

private static int minCard ;
private static int [] minR  ;

private static void check(int card, int [] r) {
  if (card < minCard) {
    minCard = card ;
    for (int i = 0 ; i < r.length ; i++)
      minR[i] = r[i] ;
  }
}

private static void look(int [] d, int m, int i, int card, int [] r) {
  int di = d[i] ;
  if (di == 1) {
    r[i] = m ;
    check(card + m, r) ;
  } else {
    int max = m/di ;
    for (int j = 0 ; j <= max ; j++) {
      r[i] = j ;
      look(d, m-j*di, i+1, card+j,r) ;
    }
  }
}

static int [] exhaustif(int [] system, int m) {
  int len = system.length ;
  minR = new int [len] ;
  minCard = Integer.MAX_VALUE ;
  look(system, m, 0, 0, new int [len]) ;
  return minR ;
}

/*********************************************/
/* Optimal par recherche exhaustive, optimisé */
/*********************************************/

private static void lookOpt(int [] d, int m, int i, int card, int [] r) {
  int di = d[i] ;
  if (card < minCard && di*(minCard-card) >= m) {
    if (di == 1) {
      r[i] = m ;
      check(card + m, r) ;
    } else {
      int max = m/di ;
      for (int j = max ; j >= 0 ; j--) {
        r[i] = j ;
        lookOpt(d, m-j*di, i+1, card+j,r) ;
      }
    }
  }
}

static int [] exhaustifOpt(int [] system, int m) {
  int len = system.length ;
  minR = glouton(system, m) ;
  minCard = card(minR) ;
  lookOpt(system, m, 0, 0, new int [len]) ;
  return minR ;
}
```

```
  /*************************************************/
  /* Cardinal de l'optimal, programmation dynamique */
  /*************************************************/

  static int [] dynamiqueCard(int [] d, int m) {
    int k = d.length ;
    int [] card = new int [m+1] ;

    card[0] = 0 ; // Inutile en Java
    for (int n = 1 ; n <= m ; n++) {
      int r = m+1 ;
      int i ;
      for (i = 0 ; d[i] > n ; i++)
        ;
      for (; i < k ; i++) {
        int di = d[i] ;
        if (card[n-di]+1 < r) {
          r = card[n-di]+1 ;
        }
      }
      card[n] = r ;
    }
    return card ;
  }


  /*************************************/
  /* Optimal, programmation dynamique */
  /*************************************/

  static int [] dynamique(int [] d, int m) {
    int k = d.length ;
    int [] card = dynamiqueCard(d, m) ;
    int [] r = new int[k] ;
    while (m > 0) {
      int c = card[m] ;
      int i ;
      for (i=0 ; d[i] > m ; i++)
        ;
      for ( ; ; i++) {
        int di = d[i] ;
        if (card[m-di] == c-1) {
          m -= di ; r[i]++ ;
          break ;
        }
      }
    }
    return r ;
  }

  public static void main (String [] arg) {
    int m = Integer.parseInt(arg[0]) ;
    int [] system = new int[arg.length-1] ;

    for (int i = 0 ; i < system.length ; i++)
      system[i] = Integer.parseInt(arg[i+1]) ;

    System.out.println("*** Glouton ***") ;
    dump(System.out, system, glouton(system, m)) ;
    System.out.println("*** Dynamique ***") ;
    dump(System.out, system, dynamique(system, m)) ;
    System.out.println("*** Opt Opt ***") ;
    dump(System.out, system, exhaustifOpt(system, m)) ;
  }
}
```