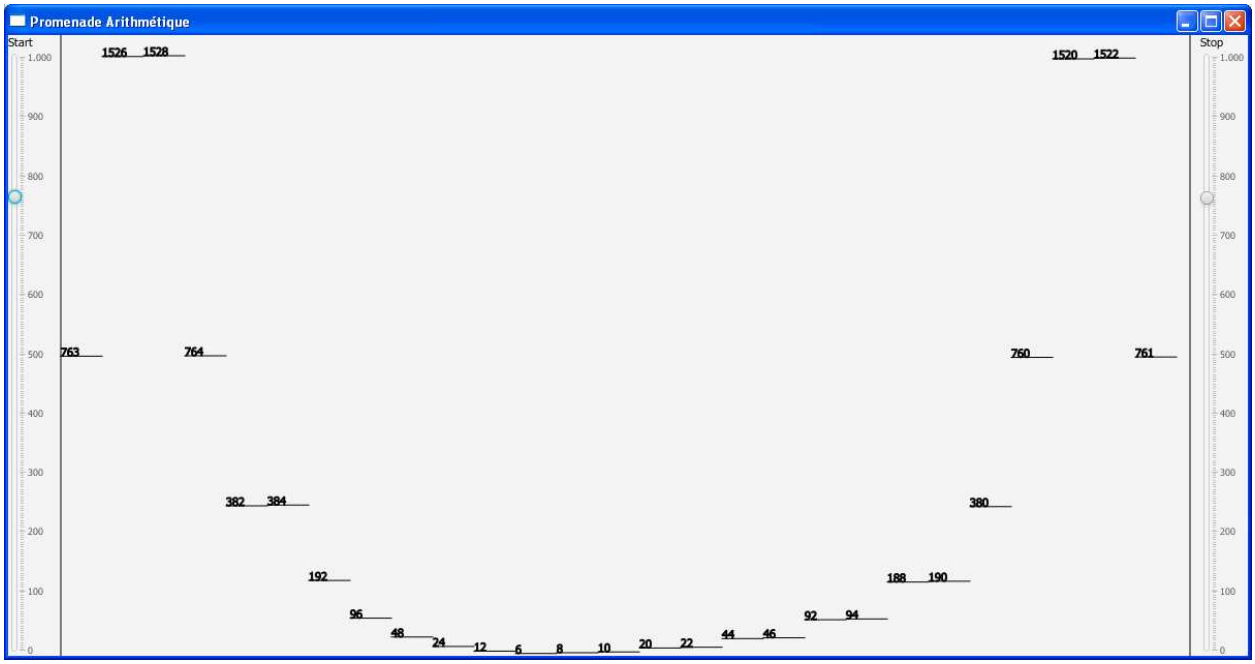
11ème édition

Le 24 avril 2015



Promenade arithmétique

Règlement

Date et Lieu : La 11ème édition du concours de programmation de l'EPFC se déroulera le **vendredi 24 avril 2015** dans les locaux de l'***EPFC*** à Woluwe-Saint-Pierre (2, av. Charles Thielemans, 1150 Bruxelles) de 13h25 à 21h30.

Candidats : Le concours est ouvert aux étudiants en deuxième et troisième année du bachelier en informatique de l'***EPFC***, ainsi qu'aux anciens étudiants. Des dérogations pour d'autres étudiants peuvent être accordées. Des préinscriptions auprès de A. SILOVY, B. LACROIX et B. VERHAEGEN sont souhaitées. Les étudiants peuvent concourir en équipe de 2 ou 3 à condition que ces équipes soient clairement identifiées, auprès des responsables, au départ de l'épreuve. Bien entendu, dans ce cas, un éventuel prix sera partagé par les membres de l'équipe.

Epreuve : Programmation d'une (ou plusieurs) application(s) console. Les sujets des problèmes posés ne privilégient aucun thème particulier. Il s'agit avant tout de problèmes purement algorithmiques qui ne nécessitent pas de connaissances spécifiques sur des domaines précis d'informatique (programmation graphique, réseau, etc.) ni d'autre domaine scientifique (mathématiques, physique, etc.). Tout langage de programmation approprié à l'écriture d'une telle application est accepté. En particulier, l'infrastructure de l'EPFC prévoit l'usage des langages *Java*, *C#*, *C/C++*, *Python*, *Scala* et *Haskell*. Les participants désireux d'utiliser un autre langage de programmation doivent s'adresser préalablement à l'un des responsables. A l'issue du concours, les candidats devront remettre le code source de leur solution (et si d'application, l'exécutable) sur le serveur du réseau informatique de l'***EPFC***.

Conditions : Les participants peuvent apporter toute documentation écrite ou électronique qu'ils peuvent juger utile. Ils peuvent donc employer des librairies de code sur clé *USB*… Cependant, durant la durée de l'épreuve, ils ne peuvent pas communiquer avec l'"extérieur". L'usage d'*Internet* et du *GSM* (à l'exception évidente d'une impérieuse exigence) est proscrit.

Classement : Contrairement aux habitudes pédagogiques, le premier critère de sélection sera le fonctionnement des exécutables répondants aux demandes formulées dans l'énoncé. Il correspond donc à **la correction du programme**. Pour départager les candidats, le classement se fera ensuite sur base des critères suivants :

- L'algorithme choisi (et sa complexité)
- Le choix des structures de données
- L'analyse du problème
- L'élégance du code source
- L'efficacité de la solution

Prix : **Des prix sont prévus, soit sous forme de matériel, soit sous forme de bons d'achat.**

La proclamation des résultats aura lieu le vendredi 23 mai à 17h30.
Elle sera suivie d'un drink offert à tous les participants

Le problème: Étant donnés deux nombres entiers positifs, comment passer de l'un à l'autre en n'utilisant que les opérations arithmétiques suivantes :

- **ajouter 2**
- **multiplier par 2**
- **diviser par 2** (la division par 2 n'étant possible que pour les nombres pairs).

Par exemple, pour passer de 2 à 9, on pourrait obtenir la suite $2 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 18 \rightarrow 9$ en effectuant successivement les opérations +2, *2, *2, *2, /2.

Remarquez qu'il est possible d'obtenir un résultat équivalent en effectuant une autre séquence d'opérations. Bien sur, trivialement, en remplaçant la 1ère opération (+2) de l'exemple ci-dessus par *2 pour donner la même suite. Mais aussi, de façon plus intéressante, en effectuant successivement les opérations /2, +2, +2, +2, +2 pour donner la suite $2 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 9$. Cette solution est également acceptable.

Les deux suites proposées ci-dessus ont une longueur de 6. Il existe aussi des solutions plus longues. Par exemple : $2 \rightarrow 4 \rightarrow 8 \rightarrow 10 \rightarrow 5 \rightarrow 7 \rightarrow 9$ ou $2 \rightarrow 4 \rightarrow 6 \rightarrow 12 \rightarrow 14 \rightarrow 7 \rightarrow 9$ ou même, $2 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 16 \rightarrow 32 \rightarrow 34 \rightarrow 36 \rightarrow 18 \rightarrow 9$.

Même si les promenades parmi les nombres sont agréables, les plus courtes sont les meilleures. **Vous veillerez donc à fournir une solution la plus courte possible**.

D'autres exemples vous sont fournis sous forme de fichiers textes (voir ci-dessous). Comme le montre l'exemple présenté en page de garde, il est aussi possible que la valeur initiale soit plus grande que la valeur finale.

En pratique, votre programme doit lire, sur l'entrée standard, deux nombres entiers positifs et écrire sur la sortie standard la suite de nombres, espacés par un blanc, commençant par le premier fourni et se terminant par le deuxième. Par exemple, si votre programme se nomme `promenade`, son exécution pourrait donner lieu à l'interaction ci-dessous :

```
C:\>promenade↵
2↵
9↵
2 4 8 16 18 9
C:\>
```

Remarques

Dans cet exemple, la saisie de l'utilisateur est soulignée. Le reste est affiché par l'ordinateur. Bien entendu, en *Java*, la commande serait `java Promenade`↵.

Consignes

Vous ne devez pas vérifier la saisie de l'utilisateur. Vous pouvez donc supposer que le format spécifié ci-dessus sera toujours respecté, l'utilisateur n'entrant jamais que deux nombres entiers positifs.

Par contre, votre affichage doit respecter scrupuleusement les spécifications données, à savoir, afficher uniquement la suite de nombres, séparés par un espace. En particulier, il ne doit pas afficher de message pour la saisie des données (du genre « `entrez deux nombres` »…).

Ces consignes ont pour objectif de permettre la mise en œuvre de tests automatiques de votre programme. Veillez à les respecter scrupuleusement.

Annexes

Afin de vous illustrer l'usage des entrées/sorties dans différents langages (*Java*, *C++*, *C*, *C#*, *Scala*, *Python* et *Haskell*), vous trouverez, ci-après, et/ou sur le serveur, au format électronique, les codes sources d'un programme simple qui utilise le même format d'Entrée/Sortie que celui de votre problème. Vous devriez utiliser le format de ces E/S sans modification pour votre programme!

Tous les fichiers utiles à votre travail sont disponibles sur le serveur au point **`PROFS-PUBLIC\CONCOURS`**. Les programmes d'exemples d'entrées/sorties dans les différents langages sont dans le dossier **`DemosIO`**, les quelques fichiers d'exemples de solutions possibles dans le dossier **`Exemples`**.

Réponses

Vous copierez vos solutions (sources et exécutables) dans un répertoire portant votre nom sur le serveur au point **`PROFS-PUBLIC\CONCOURS\REPONSES`**.

Bonne Promenade

DemoIO.java

```
/*
Programme de démo des E/S pour le concours de programmation EPFC 2015.
Vous ne devez pas modifier le format de ces E/S

Ce programme lit deux nombres entiers positifs et
affiche tous les entiers de l'un à l'autre, séparés par un espace.
*/

import java.util.*;

public class DemoIO {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int start = s.nextInt();
    int stop = s.nextInt();
            if(start <= stop)
      for(int k = start; k <= stop; ++k)
                System.out.print(k + " ");
    else
      for(int k = start; k >= stop; --k)
                System.out.print(k + " ");
  }
}
```

DemoIO.cpp

```
/*
Programme de démo des E/S pour le concours de programmation EPFC 2015.
Vous ne devez pas modifier le format de ces E/S

Ce programme lit deux nombres entiers positifs et
affiche tous les entiers de l'un à l'autre, séparés par un espace.
*/

#include <iostream>

using namespace std;

int main() {
  unsigned start, stop;
  cin >> start >> stop;
  if(start <= stop)
    for(unsigned k = start; k <= stop; ++k)
      cout << k << ' ';
  else
    for(unsigned k = start; k >= stop; --k)
      cout << k << ' ';
}
```