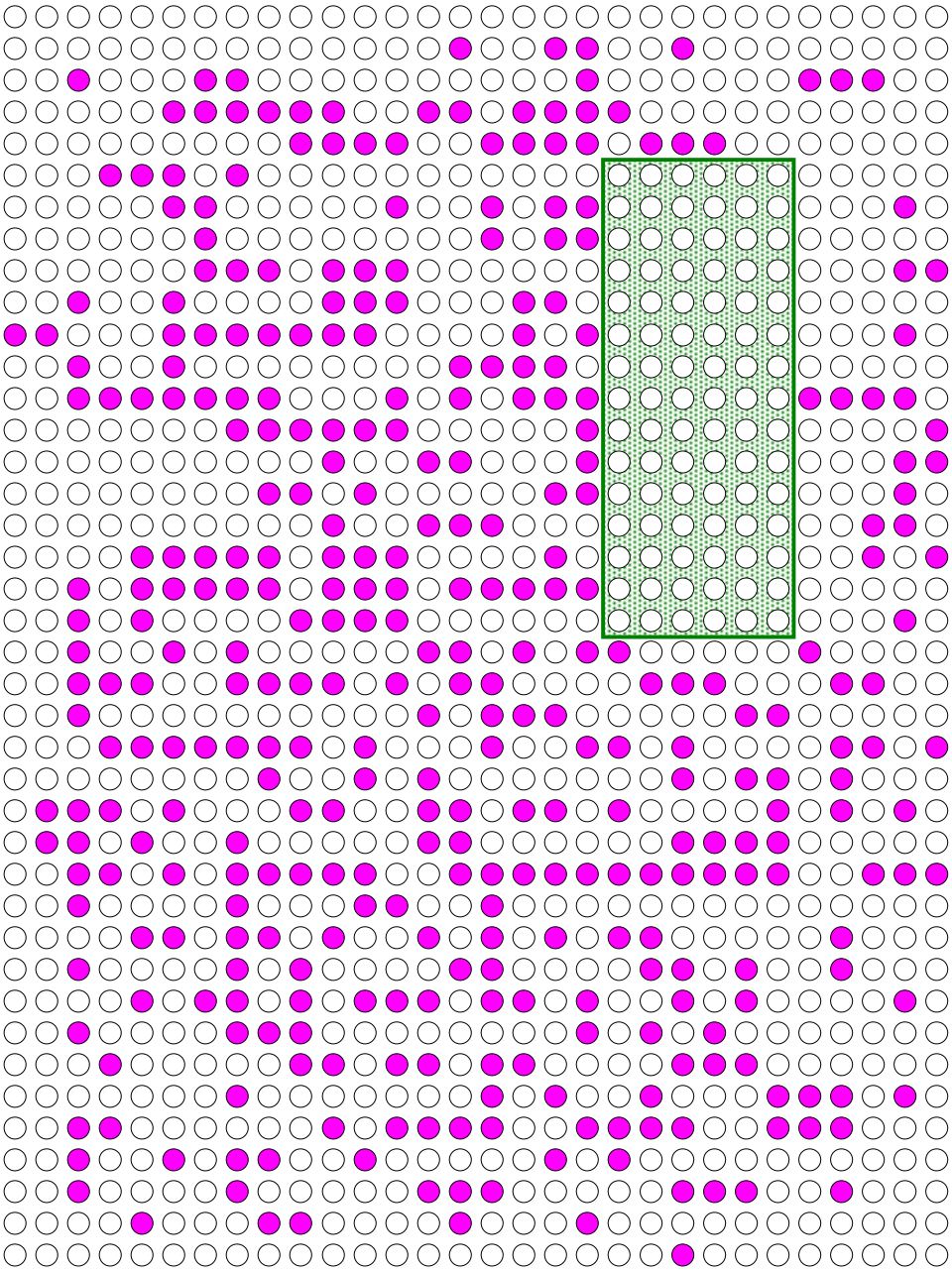
12ème édition

Le 22 avril 2016



Tous ensemble au spectacle

Règlement

Date et Lieu : La 12ème édition du concours de programmation de l'EPFC se déroulera le **vendredi 22 avril 2016** 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. Afin de faciliter l'organisation du concours, des préinscriptions auprès de A. SILOVY (asilovy@ulb.ac.be) 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 (ainsi que 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 répondant 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

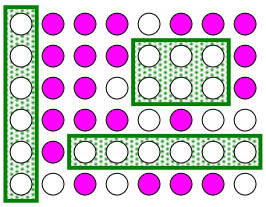
Prix : **Des prix sont prévus, soit sous forme de matériel, soit sous forme de bons d'achat.**

Une proclamation des résultats sera organisée avec remise des prix.
Elle sera suivie d'un drink offert à tous les participants

Le problème: Vous formez un groupe d'amis et vous vous rendez au spectacle. Les places sont attribuées par siège et on vous présente un tableau des places libres. La salle de spectacle comporte *nbR* rangées de *nbS* sièges. Comme vous voudriez être groupés, vous décidez de chercher la plus grande zone rectangulaire possible composée exclusivement de sièges libres.

Sur l'image de la couverture, les sièges occupés sont colorés et la plus grande zone rectangulaire de sièges libres est mise en évidence. Elle est formée de 15 rangées de 6 sièges et comporte donc 90 places disponibles. Remarquez que nous ne considérons que des zones rectangulaires dont les bords sont alignés aux bords de la salle (pas de rectangle oblique).

L'illustration ci-dessous donne un exemple plus simple :



Elle montre qu'il existe 3 zones rectangulaires comportant le plus grand nombre de sièges libres possibles avec chacune 6 sièges. Elle montre aussi que des sièges sur une seule rangée (ou colonne) forment un rectangle.

En pratique, votre programme doit lire sur l'entrée standard deux nombres entiers positifs, représentant respectivement *nbR*, le nombre de rangées et *nbS*, le nombre de sièges par rangée (les dimensions de la salle). Ces nombres sont suivis de *nbR* lignes formées chacune de *nbS* nombres valant uniquement `1` ou `0`, un `1` indiquant un siège libre et un `0` indiquant un siège réservé. **Votre programme doit juste écrire sur la sortie standard le nombre de sièges de la plus grande zone rectangulaire de sièges libre**.

Par exemple, si votre programme se nomme `sl` (pour **s**ièges **l**ibres), son exécution pourrait donner lieu à l'interaction ci-dessous :

```
C:\>sl↵
6↵
8↵
1 0 0 0 1 0 0 0↵
1 0 0 0 1 1 1 0↵
1 0 0 1 1 1 1 0↵
1 0 0 0 1 0 1 1↵
1 0 1 1 1 1 1 1↵
1 1 0 1 0 0 0 1↵
6
C:\>
```

Remarques

Dans cet exemple, la saisie de l'utilisateur est soulignée. Le reste (c'est-à-dire la dernière ligne) est affiché par l'ordinateur. Bien entendu, en *Java*, la commande serait `java Sl↵`.

En pratique, il n'est pas commode d'entrer de grandes séries de nombres au clavier. Nous utiliserons donc la redirection des entrées sur des fichiers textes préparés avec l'input voulu. Par exemple, si vous disposiez d'un fichier texte `exemple.txt` avec le contenu ci-dessous

```
6
8
1 0 0 0 1 0 0 0
1 0 0 0 1 1 1 0
1 0 0 1 1 1 1 0
1 0 0 0 1 0 1 1
1 0 1 1 1 1 1 1
1 1 0 1 0 0 0 1
```

il vous suffirait de taper la commande suivante :

```
C:\>sl < exemple.txt ↵
6
C:\>
```

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 des nombres entiers positifs tels qu'indiqués.

Par contre, votre affichage doit respecter scrupuleusement les spécifications données, à savoir, afficher uniquement un nombre. En particulier, il ne doit pas afficher de message pour la saisie des données (du genre « `entrez deux nombres` »…) ni pour la sortie (comme : « `Voici le nombre de sièges : 6` »).

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

Remarquez bien ce qui est dit dans l'énoncé : un 1 représente un siège libre et un 0 représente un siège occupé. Vous devez donc chercher la plus grande zone rectangulaire comportant exclusivement des 1.

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`**. Des programmes d'exemples d'entrées/sorties dans les différents langages sont dans le dossier **`DemosIO`**. Quelques fichiers textes d'exemples d'inputs possibles sont 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 Chance

(et, surtout, bon amusement)

DemoIO.java

```
/*
Programme de démo des E/S pour le concours de programmation EPFC 2016.
Ce programme utilise le même format d'E/S que celui de votre énoncé.
Vous ne devez pas modifier le format de ces E/S.

Ce programme lit deux nombres entiers positifs nbR et nbS suivis de
nbR lignes formées chacunes de nbS 1 ou 0 et affiche le nombre total
de 1.
*/

import java.util.*;

public class DemoIO {
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    int nbR = scan.nextInt();
    int nbS = scan.nextInt();
    int cpt = 0;
    for(int r = 0; r < nbR; ++r)
      for(int s = 0; s < nbS; ++s) {
        int x = scan.nextInt();
        if(x == 1)
          ++cpt;
      }
    System.out.print(cpt);
  }
}
```

DemoIO.cpp

```
/*
Programme de démo des E/S pour le concours de programmation EPFC 2016.
Ce programme utilise le même format d'E/S que celui de votre énoncé.
Vous ne devez pas modifier le format de ces E/S.

Ce programme lit deux nombres entiers positifs nbR et nbS suivis de
nbR lignes formées chacunes de nbS 1 ou 0 et affiche le nombre total
de 1.
*/

#include <iostream>

using namespace std;

int main() {
  unsigned nbR, nbS, cpt = 0;
  cin >> nbR >> nbS;
  for(unsigned r = 0; r < nbR; ++r)
    for(unsigned s = 0; s < nbS; ++s) {
      unsigned x;
      cin >> x;
      if(x == 1)
        ++cpt;
    }
  cout << cpt;
}
```