Solutions and Evaluation Criteria for Exam #3 (9/5/2016)

Professors of PRO1

1 General comments

As a general rule, for the exercises with no manual evaluation, the submission will be valid unless the program deliberately "cheats" or violates the requirements given in the statement (e.g., the input or substantial parts of it are stored in internal memory, includes libraries other than those specifically allowed, etc.)

2 Exercises

2.1 X53954: Sublist of multiples

```
#include <iostream>
#include <vector>
using namespace std;

// Returns a vector of size n > 0 with n integer values
// read from cin (there are at least n+1 integers in the cin)
vector<int> read_vector(int n) {
  vector<int> x(n);
  for (int i = 0; i < n; ++i)
    cin >> x[i];
  return x;
}

// Prints the sublist of multiples of x of a vector of size
// n > 0 in the cout; see the
// statement for details
void print_multiples(const vector<int>& v, int x) {
  // the boolean variable 'first' is used to avoid
  // printing a blank space before the first element to be output
  int n = v.size();
  bool first = true;
  for (int i = 0; i < n; ++i) {
```

```
    // we avoid division by 0 checking first if v[i] == x
    if (v[i] == x or (x != 0 and v[i] % x == 0)) {
      if (first) { cout << v[i]; first = false; }
      else { cout << " " << v[i]; }
    }
  }
  cout << endl;
}

int main() {
  int n;
  while (cin >> n) {
    vector<int> v = read_vector(n);
    int x;
    cin >> x;
    print_multiples(v, x);
  }
}
```

2.2 X12559: Reorganize a vector

First solution: Using auxiliary memory

```
#include <iostream>
#include <vector>

using namespace std;

// Returns a vector of size n > 0 with n integer values read from cin.
vector<int> read_vector(int n) {
  vector<int> v(n);
  for (int i = 0; i < n; ++i)
    cin >> v[i];
  return v;
}

// Prints a vector of size n > 0 in the cout.
void print_vector(const vector<int>& v) {
  for (int i = 0 ; i < int(v.size()) ; ++i) {
    if (i>0) cout<<" ";
    cout<<v[i];
  }
  cout<<endl;
}

// see the statement of the problem
vector<int> reorganize_vector(const vector<int>& v,int a,int b)
```

```
{
  int n = v.size();
  vector<int> small; // subvector of v[i]'s < a
  vector<int> medium; // subvector of v[i]'s between a and b
  vector<int> large;   // subvector of v[i]'s > b
  for (int i = 0; i < n; ++i) {
    if (v[i] < a) small.push_back(v[i]);
    else if (v[i] <= b) medium.push_back(v[i]);
    else large.push_back(v[i]);
  }
  int s = small.size(); int m = medium.size(); int l = large.size();
  vector<int> result;
  for (int i = 0; i < s; ++i)
    result.push_back(small[i]);
  for (int i = 0; i < m; ++i)
    result.push_back(medium[i]);
  for (int i = 0; i < l; ++i)
    result.push_back(large[i]);
  return result;
}

int main() {
  int n;
  while (cin >> n) {
    vector<int> v = read_vector(n);
    int a, b;
    cin >> a >> b;
    print_vector(reorganize_vector(v, a, b));
  }
}
```

Second solution: Scanning three times

```
#include <iostream>
#include <vector>

using namespace std;

// Returns a vector of size n > 0 with n integer values read from cin.
vector<int> read_vector(int n) {
  vector<int> v(n);
  for (int i = 0; i < n; ++i)
    cin >> v[i];
  return v;
}
```

```
// Prints a vector of size n > 0 in the cout.
void print_vector(const vector<int>& v) {
  for (int i = 0 ; i < int(v.size()) ; ++i) {
    if (i>0) cout<<" ";
    cout<<v[i];
  }
  cout<<endl;
}

// see the statement of the problem
vector<int> reorganize_vector(const vector<int>& v,int a,int b)
{
  int n = v.size(); vector<int> result(n);
  int k = 0;
  for (int i = 0; i < n; ++i)
    if (v[i] < a) { result[k] = v[i]; ++k; }
  for (int i = 0; i < n; ++i)
    if (a <= v[i] and v[i] <= b) { result[k] = v[i]; ++k; }
  for (int i = 0; i < n; ++i)
    if (b < v[i]) { result[k] = v[i]; ++k; }
  return result;
}

int main() {
  int n;
  while (cin >> n) {
    vector<int> v = read_vector(n);
    int a, b;
    cin >> a >> b;
    print_vector(reorganize_vector(v, a, b));
  }
}
```

2.3 X44670: Arches

In order to obtain a correct and efficient solution to this problem it is very useful to abstract one level and consider that the input is a sequence of *blocks*, where a block is a sequence of integers delimited by 0's (or the end of the sequence) and might or might not be an *arch*. Thus at this level of abstraction our task is to find the first non-arch in the sequence of blocks, counting how many arches precede such block, and we must use a sequential search scheme. So we will define a Boolean function `read_arch()` that reads a block (or part of it) and returns **true** if and only if the block is an arch. The function assumes that the initial 0 has already been read (except for the first block, the initial 0 is the ending 0 of the previously read block).

One level of abstraction below, the design of `read_arch()` also is based on

a sequential search scheme. We are looking for an element of the (sub)sequence of integers such that it is negative or the absolute value of the difference with its predecessor is strictly larger than 1. If such element were found we can stop reading and return **false**. If no such element is found and the search ends because we reach the end of the block then we return **true** because we have read the closing 0, or we retun **false** because we have exhausted all the input without finding the closing 0.

Recall that general evaluation criteria apply as penalty factors that multiply the prescore. The specific criteria here apply to a correct (or almost correct) solution:

- ▷ If the two levels of abstraction described above are not clearly identified using suitable function or procedure definitions or a clean and clear documentation, the prescore is between 7 or 8
- ▷ If the sequential search scheme is not applied in one or the two levels of abstraction, substract 1.5 points from the prescore for each omission.

```
#include <iostream>
using namespace std;

// returns the absolute value of a
int abs(int a) {
  return a >= 0 ? a : -a;
}

// Pre: the cin contains a sequence x_2,.... of integers, the previously
// read element was = 0
// Post: read_arch reads elements from the cin until it finds a 0 closing
// a correct arch and returns true, or if the initial subsequence in the cin
// is not an arch then it returns false

bool read_arch() {
  bool is_arch = true;
  int prev = 0;
  int curr = 0;
  while (cin >> curr and curr != 0 and is_arch) {
    is_arch = curr > 0 and abs(curr-prev) <= 1;
    prev = curr;
  }
  // the expression "return cin" returns false if all the
  // input has been read; if we end the input without finding
  // the closing 0 we don't have an arch
  return cin and is_arch and curr == 0;
}

int count_initial_arches() {
```

```
  int nr_arches = 0;
  int x;
  cin >> x;
  if (x == 0)
    while (read_arch())
      ++nr_arches;

  return nr_arches;
}

int main() {
  cout << count_initial_arches() << endl;
}
```

2.4 X12415: Average of consecutive subvectors

The clue for an efficient solution to this problem is to notice that the sum $S_{i+1,k}$ of the segment of size k starting at position $i+1$ can be easily computed if we already know the sum $S_{i,k}$ of the segment of size k starting at position i. Indeed

$$S_{i+1,k} = v[i+1] + v[i+2] + \cdots + v[i+k] = S_{i,k} - v[i] + v[i+k]$$

There is thus only one specific criterion that applies in this problem. A correct solution that fails to benefit from the above observation (and hence performs a lot of redundant computations) will have a prescore of 7. Thereafter general evaluation criteria apply as penalty factors that multiply the prescore.

```
#include <iostream>
#include <vector>
using namespace std;

// reads and returns the contents of a vector of n reals from cin, n > 0
vector<int> read_vector(int n) {
  vector<int> v(n);
  for (int i = 0; i < n; ++i)
    cin >> v[i];
  return v;
}

// see the stament of the problem
void print_average_of_segments(const vector<int>& v, int k) {
  // print the average of the initial segment (v[0]+...+v[k-1])/k
  int curr_seg_sum = 0; int i;
  for (i = 0; i < k; ++i)
    curr_seg_sum += v[i];
  cout << double(curr_seg_sum) / k;
```

```
  i = 1;
  while (i + k <= int(v.size())) {
    // Inv: curr_seg_sum == (v[i-1] + ... + v[i+k-2])
    curr_seg_sum = curr_seg_sum - v[i-1] + v[i+k-1];
    cout << " " << double(curr_seg_sum) / k;
    ++i;
  }

  cout << endl;
}

int main() {
  cout.setf(ios::fixed);
  cout.precision(4);
  int n,k;
  while (cin >> n >> k) {
    vector<int> v = read_vector(n);
    print_average_of_segments(v,k);
  }
}
```

Sublist of multiples X53954_en

Given a list of n natural numbers $v_0, v_1, \ldots, v_{n-1}$ and a natural number x, we want to write the sublist of v_i's that are multiple of x (recall that a natural number a is multiple of a natural number b if there exists a natural number c such that $a = c * b$).
For example, if $v = [12, 3, 4, 2, 0, 5, 23, 4, 18, 32]$ and $x = 4$, then the output must be $[12, 4, 0, 4, 32]$, as those are the only multiples of 4 of the given list.
As another example, if $v = [5, 0, 2, 8, 3, 0, 4, 13]$ and $x = 0$, then the output must be $[0, 0]$. Note that 0 is the only possible multiple of 0.

==========
CASTELLANO
==========

Dada una lista de n números naturales $v_0, v_1, \ldots, v_{n-1}$ y otro número natural x, queremos escribir la sublista de v_i's que son múltiplos de x (recuerda que un número natural a es múltiplo de un número natural b si existe algún número natural c tal que $a = c * b$).
Por ejemplo, si $v = [12, 3, 4, 2, 0, 5, 23, 4, 18, 32]$ y $x = 4$, entonces la salida ha de ser $[12, 4, 0, 4, 32]$ ya que son los únicos múltiplos de 4 en la lista dada.
Otro ejemplo: si $v = [5, 0, 2, 8, 3, 0, 4, 13]$ y $x = 0$, entonces la salida es $[0, 0]$. Observa que 0 es el único múltiplo de 0.

==========
CATALÀ
==========

Donada una llista de n nombres naturals $v_0, v_1, \ldots, v_{n-1}$ i un altre nombre natural x, volem escriure la subllista de v_i 's que són múltiples de x (recorda que un nombre natural a és múltiple d'un nombre natural b si hi ha algun nombre natural c tal que $a = c * b$).
Per exemple, si $v = [12, 3, 4, 2, 0, 5, 23, 4, 18, 32]$ i $x = 4$, llavors la sortida ha de ser $[12, 4, 0, 4, 32]$ ja que són els únics múltiples de 4 a la llista donada.
Un altre exemple: si $v = [5, 0, 2, 8, 3, 0, 4, 13]$ i $x = 0$, llavors la sortida és $[0, 0]$. Observa que 0 és l'únic múltiple de 0.
Exam score: 2.5 **Automatic part:** 100%

Input

The input has several cases, and each case is described with three lines. In the first line there is an integer $n > 0$. In the second line there are n natural numbers $v_0, \ldots, v_{n-1}$ (the list). In the third line there is a natural number x.

==========
CASTELLANO
==========

La entrada consiste en varios casos, cada caso se describe en tres líneas. La primera línea contiene un entero $n > 0$. La segunda línea contiene n números naturales $v_0, \ldots, v_{n-1}$ (la lista dada). La tercera línea contiene un número natural x.

==========
CATALÀ
==========

L'entrada consisteix en diversos casos, cada cas es descriu en tres línies. La primera línia

conté un enter $n > 0$. La segona línia conté n nombres naturals $v_0, \ldots, v_{n-1}$ (la llista donada). La tercera línia conté un nombre natural x.

Output

For each case, write the corresponding sublist of multiples of x in one line, ended with a line break, and separating each two consecutive elements with a blank space.

==========

CASTELLANO

==========

Para cada caso, escribe la correspondiente sublista de múltiplos de x en una línea, finalizada con un salto de línea, y separando cada par de elementos consecutivos con un espacio en blanco.

==========

CATALÀ

==========

Per a cada cas, escriu la corresponent subllista de múltiples de x en una línia, finalitzada amb un salt de línia, i separant cada parell d'elements consecutius amb un espai en blanc.

Sample input

```
8
6 1 3 1 7 2 4 1
2
4
1 2 0 3
0
2
2 4
1
3
5 7 1
0
3
2 0 4
3
8
6 1 3 1 7 2 4 1
5
1
1
1
```

Sample output

```
6 2 4
0
2 4

0

1
```

Problem information

Author : PRO1
Generation : 2016-05-08 14:25:56

Reorganize a vector — X12559_en

Write a function `reorganize_vector(v,a,b)` that receives a vector $v = (v_0, ..., v_{n-1})$ of integers and two integers $a \leq b$ as parameters, and returns a vector w that contains the same elements as v but reorganized as follows: on the first positions of w there are the elements v_i such that $v_i < a$ (and preserving their original respective order in v between them), next there are the elements v_i such that $a \leq v_i \leq b$ (again preserving the original order in v), and finally the elements v_i satisfying $b < v_i$ (again preserving the original order in v).
For example, for $v = [3,6,2,7,5,1,4,3,1,7,2,5,6,4,3]$, $a = 3$, $b = 5$, the subvector of elements strictly smaller than a is $[2,1,1,2]$, the subvector of elements between a and b is $[3,5,4,3,5,4,3]$, and the subvector of elements strictly greater than b is $[6,7,7,6]$. Thus, the function must return $w = [2,1,1,2,3,5,4,3,5,4,3,6,7,7,6]$.
Your submission must use the following C++ code, modifying only the parts that are explicitly indicated.

```
#include <iostream>
#include <vector>

using namespace std;

// Returns a vector of size n > 0 with n integer values read from cin.
vector<int> read_vector(int n) {
    vector<int> v(n);
    for (int i = 0; i < n; ++i)
       cin >> v[i];
    return v;
}

// Prints a vector of size n > 0 in the cout.
void print_vector(const vector<int>& v) {
    for (int i = 0 ; i < int(v.size()) ; ++i) {
        if (i>0) cout<<" ";
        cout<<v[i];
    }
    cout<<endl;
}

// add functions or procedures here if you need them

// see the statement of the problem
vector<int> reorganize_vector(const vector<int>& v,int a,int b)
{
    // Write your implementation here.
}

int main() {
    int n;
```

```
    while (cin >> n) {
        vector<int> v = read_vector(n);
        int a, b;
        cin >> a >> b;
        print_vector(reorganize_vector(v, a, b));
    }
}
```

==========
CASTELLANO
==========

Escribe una función `reorganize_vector(v, a, b)` que recibe un vector $v = (v_0, \ldots, v_{n-1})$ de enteros y dos enteros $a \leq b$ como parámetros, y devuelve un vector w que contiene los mismos elementos que v pero reorganizados como sigue: las primeras posiciones de w contienen los elementos v_i tales que $v_i < a$ (preservando su orden original respectivo en v), a continuación vienen los elementos v_i tales que $a \leq v_i \leq b$ (nuevamente preservando su orden original en v) y finalmente están los elementos v_i tales que $b < v_i$ (nuevamente preservando su orden original en v).

Por ejemplo, para $v = [3,6,2,7,5,1,4,3,1,7,2,5,6,4,3]$, $a = 3$ y $b = 5$, el subvector de elementos estrictamente menores que a es [2,1,1,2], el subvector de elementos entre a y b es [3,5,4,3,5,4,3], y el subvector de elementos estrictamente mayores que b es [6,7,7,6]. Por lo tanto, la función debe devolver $w = [2,1,1,2,3,5,4,3,5,4,3,6,7,7,6]$.

Tu envío (*submission*) debe utilizar obligatoriamente el código en C++ dado más arriba, modificando sólo las partes explícitamente indicadas.

==========
CATALÀ
==========

Escriu una funció `reorganize _vector(v, a, b)` que rep un vector $v = (v_0, \ldots, v_{n-1})$ d'enters i dos enters $a \leq b$ com paràmetres, i retorna un vector w que conté els mateixos elements que v però reorganitzats com segueix: les primeres posicions de w contenen els elements v_i tals que $v_i < a$ (preservant el seu ordre original respectiu en v), a continuació vénen els elements v_i tals que $a \leq v_i \leq b$ (novament preservant el seu ordre original en v) i finalment hi ha els elements v_i tals que $b < v_i$ (novament preservant el seu ordre original en v).

Per exemple, per $v = [3,6,2,7,5,1,4,3,1,7,2,5,6,4,3]$, $a = 3$ i $b = 5$, el subvector d'elements estrictament menors que a és [2,1,1,2], el subvector d'elements entre a i b és [3,5,4,3,5,4,3], i el subvector d'elements estrictament més grans que b és [6,7,7,6]. Per tant, la funció ha de retornar $w = [2,1,1,2,3,5,4,3,5,4,3,6,7,7,6]$.

El teu enviament (*submission*) ha d'utilitzar obligatòriament el codi en C ++ donat més amunt, modificant només les parts explícitament indicades.

Exam score: 2.5 **Automatic part:** 100%

Input

The input is a sequence of cases. Each case is described with three lines. The first line has an integer $n > 0$. The second line has n integers, that is the contents of a vector v of size n, and the third line has two more integers a and b holding $a \leq b$.

==========
CASTELLANO
==========

La entrada es una secuencia de casos. Cada caso se describe en tres líneas. La primer línea contiene un entero $n > 0$. La segunda línea contiene n enteros, el contenido del vector v de tamaño n, y la tercera línea contiene dos enteros más, a y b, tales que $a \leq b$.

==========

CATALÀ

==========

L'entrada és una seqüència de casos. Cada cas es descriu en tres línies. La primera línia conté un enter $n > 0$. La segona línia conté n enters, el contingut del vector v de mida n, i la tercera línia conté dos enters més, a i b, tals que $a \leq b$.

Output

For each case, the program outputs the contents of the resulting vector from a call `reorganize_vector(v, a, b)` followed by a line break. Two consecutive elements are separated by a white space.

==========

CASTELLANO

==========

Para cada caso, el programa imprime el contenido del vector retornado por la llamada `reorganize_vector(v, a, b)`, seguidos por un salto de línea. Cualquier par de elementos consecutivos se imprimen separados por un espacio en blanco.

==========

CATALÀ

==========

Per a cada cas, el programa imprimeix el contingut del vector retornat per la crida `reorganize_vector(v, a, b)`, seguits per un salt de línia. Qualsevol parell d'elements consecutius s'imprimeixen separats per un espai en blanc.

Sample input

```
15
3 6 2 7 5 1 4 3 1 7 2 5 6 4 3
3 5
10
4 -3 5 7 1 4 8 6 5 -7
3 5
10
4 -3 5 7 1 4 8 6 5 -7
-10 -9
10
4 -3 5 7 1 4 8 6 5 -7
2 3
10
4 -3 5 7 1 4 8 6 5 -7
9 15
10
4 -3 5 7 1 4 8 6 5 -7
-10 3
10
4 -3 5 7 1 4 8 6 5 -7
6 15
```

Sample output

```
2 1 1 2 3 5 4 3 5 4 3 6 7 7 6
-3 1 -7 4 5 4 5 7 8 6
4 -3 5 7 1 4 8 6 5 -7
-3 1 -7 4 5 7 4 8 6 5
4 -3 5 7 1 4 8 6 5 -7
-3 1 -7 4 5 7 4 8 6 5
4 -3 5 1 4 5 -7 7 8 6
```

Problem information

Author : PRO1
Generation : 2016-05-08 14:32:13

Arches X44670_en

A sequence $x_1, x_2, \ldots, x_n$ of integers of length $n \geq 2$ is an *arch* if the following conditions are satisfied:

- The first and the last element of the sequence are 0.
- All elements, other than the first and the last, are strictly positive, that is, for all i, $1 < i < n$, we must have $x_i > 0$.
- The difference between any two consecutive elements in the sequence is +1, 0, or -1, that is, for all i, $1 < i \leq n$, we must have $x_i - x_{i-1} \in \{-1, 0, +1\}$

Consider a sequence of concatenated archs like this one

0 1 2 1 0 1 2 2 2 3 4 3 2 3 2 1 0 0

Notice that the ending 0 of an arch is the initial 0 of the next arch. The example sequence contains 3 arches. The first arch is 0 1 2 1 0, the second arch is 0 1 2 2 2 3 4 3 2 3 2 1 0 and the last arch is 0 0.
Write a function `int count_initial_arches()` that reads a sequence of integers from the standard input (`cin`) and returns the number of arches found before we encounter a non-arch.
For instance, for a sequence that starts like this:

0 1 2 1 0 1 0 0 2 2 2 3 4 3 2 3 2 1 0 0 ...

the answer is 3; we find the three arches 0 1 2 1 0, 0 1 0, and 0 0, then followed the subsequence 0 2 2 3 ... which is not an arch.
Use the following C++ structure. Note that in general it is not necessary to read the whole input to obtain the answer, and not taking this into account could give you a "Time Limit Exceeded" verdict.

```
#include <iostream>

using namespace std;

// Add here other function definitions if you need them.

// Read problem statement.
int count_initial_arches()
{
  // Write here your code
  ...
}

int main()
{
  cout<<count_initial_arches()<<endl;
}
```

==========
CASTELLANO
==========

Una secuencia $x_1, x_2, \ldots, x_n$ de enteros de longitud $n \geq 2$ es un *arco* si satisface las siguientes condiciones:

- El primer y el último elementos de la secuencia son 0.
- Todos los elementos, exceptuados el primero y último, son estrictamente positivos, esto es, para todo i, $1 < i < n$, se cumple que $x_i > 0$.
- La diferencia entre cualesquiera dos elementos consecutivos de la secuencia es +1, 0 ó -1, es decir, para todo i, $1 < i \leq n$, se cumple que $x_i - x_{i-1} \in \{-1, 0, +1\}$.

Considera ahora una secuencia de arcos concatenados como la siguiente:

0 1 2 1 0 1 2 2 2 3 4 3 2 3 2 1 0 0

Observa que el 0 al final de un arco es el 0 inicial del siguiente arco. La secuencia del ejemplo contiene 3 arcos. El primer arco es 0 1 2 1 0, el segundo arco es 0 1 2 2 2 3 4 3 2 3 2 1 0 y el último arco es 0 0.
Escribe una función `int count_initial_arches()` que lee una secuencia de enteros de la entrada estándar (`cin`) y retorna el número de arcos encontrado antes de encontrar un no-arco. Por ejemplo, para una secuencia que comienza de la siguiente manera:

0 1 2 1 0 1 0 0 2 2 2 3 4 3 2 3 2 1 0 0 ...

la respuesta es 3; hallamos los tres arcos 0 1 2 1 0, 0 1 0, y 0 0, y a continuación la subsecuencia 0 2 2 3 ... que no es un arco.
Usa el código C++ dado más arriba. Observa que en general no es necesario leer toda la entrada para obtener la respuesta, y no tener esto en cuenta puede conducir a un veredicto "Time Limit Exceeded".

==========
CATALÀ
==========

Una seqüència $x_1, x_2, \ldots, x_n$ d'enters de longitud $n \geq 2$ és un *arc* si satisfà les següents condicions:

- El primer i l'últim elements de la seqüència són 0.
- Tots els elements, exceptuats el primer i últim, són estrictament positius, és a dir, per a tot i, $1 < i < n$, es compleix que $x_i > 0$.
- La diferència entre qualssevol dos elements consecutius de la seqüència és +1, 0 o -1, és a dir, per a tot i, $1 < i \leq n$, es compleix que $x_i - x_{i-1} \in \{-1, 0, +1\}$.

Considera ara una seqüència d'arcs concatenats com la següent:

0 1 2 1 0 1 2 2 2 3 4 3 2 3 2 1 0 0

Observa que el 0 al final d'un arc és el 0 inicial del següent arc. La seqüència de l'exemple conté 3 arcs. El primer arc és 0 1 2 1 0, el segon arc és 0 1 2 2 2 3 4 3 2 3 2 1 0 i l'últim arc és 0 0.
Escriu una funció `int count _initial _arches ()` que llegeix una seqüència d'enters de l'entrada estàndard (`cin`) i retorna el nombre d'arcs trobat abans de trobar un no-arc. Per exemple, per a una seqüència que comença de la següent manera:

0 1 2 1 0 1 0 0 2 2 2 3 4 3 2 3 2 1 0 0 ...

la resposta és 3; trobem els tres arcs 0 1 2 1 0, 0 1 0, i 0 0, i tot seguit la subseqüència 0 2 2 3 ... que no és un arc.
Utilitza el codi C ++ donat més amunt. Observa que en general no cal llegir tota l'entrada per obtenir la resposta, i no tenir això en compte pot conduir a un veredicte "Time Limit Exceeded".
Exam score: 2.5 **Automatic part:** 30%

Input

A sequence of integers.
==========
CASTELLANO
==========
Una secuencia de enteros.
==========
CATALÀ
==========
Una seqüència d'enters.

Output

The program prints in the cout the number of arches read until a non-arch occurs in the input.
==========
CASTELLANO
==========
El programa imprime en la salida (`cout`) el número de arcos leídos hasta el primer no-arco en la entrada.
==========
CATALÀ
==========
El programa imprimeix a la sortida (`cout`) el nombre d'arcs llegits fins al primer no-arc a l'entrada.

Sample input 1	**Sample output 1**
`0 0 0 0 1 1 0 0 1 2 4 3 2 1 0 0 1 0`	`5`

Sample input 2	**Sample output 2**
`1 0 1 2 1 2 2 3 4 5 5 6 5 4 5 4 4 4 4 3 2 1 0 0`	`1`

Sample input 3	**Sample output 3**
`0 1 0 1 2 1 2 2 3 4 5 5 6 5 4 5 4 4 4 4 3 2 1 0`	`2`

Sample input 4	**Sample output 4**
`0 1 2 2 1 1 0 1 0 -1 0 1 2 1 0 2 0 1 0`	`2`

Problem information

Author : PRO1
Generation : 2016-05-08 14:45:17

Average of consecutive subvectors X12415_en

Given a vector $v[0], v[1], \ldots, v[n-1]$ of integer values, the average of the segment starting at position i and of size k is $(v[i] + v[i+1] + \cdots + v[i+k-1])/k$. Note that i and k must satisfy $0 \leq i \leq n-k$ and $0 < k$ in order to have a correct definition. For example, for the vector $v = [3, 2, 5, 1, 4]$, the average of the segment starting at position 1 of size 3 (i.e. the average of the subvector $[2, 5, 1]$) is $(2+5+1)/3 = 2.6667$ written with four digits after the decimal point.

Write a function `print_average_of_segments(v,k)` that, given a vector v of integers and a positive natural number k, prints in the standard output (`cout`) the average of all segments of v of size k from left to right. Note that for the above example of v and $k = 3$ we have the following consecutive subvectors (segments) of size k: $[3, 2, 5]$, $[2, 5, 1]$, $[5, 1, 4]$. Thus, in this case the function must print 3.33332.66673.3333.

Think of a solution that avoids unnecesary computations. Solutions that do not take this into account will likely be rejected by the Jutge with a TLE error ("time limit exceeded") when n and k are large numbers.

Use the following C++ code, completing the missing parts. Recall that the operator / makes integer division if both of its operands are integers.

```
#include <iostream>
#include <vector>
using namespace std;

// reads and returns the contents of a vector of n reals from cin, n > 0
vector<int> read_vector(int n) {
     vector<int> v(n);
     for (int i = 0; i < n; ++i)
         cin >> v[i];
     return v;
}

// add here functions or procedures if you need them

// see the stament of the problem
void print_average_of_segments(const vector<int>& v, int k) {
         ...
}

int main() {
    cout.setf(ios::fixed);
    cout.precision(4);
    int n,k;
    while (cin >> n >> k) {
        vector<int> v = read_vector(n);
        print_average_of_segments(v,k);
    }
}
```

==========
CASTELLANO
==========
Dado un vector $v[0], v[1], \ldots, v[n-1]$ de enteros, la media del segmento de tamaño k que comienza en la posición i es $(v[i] + v[i+1] + \cdots + v[i+k-1])/k$. Observa que i y k deben satisfacer $0 \leq i \leq n-k$ y $k > 0$ para que la definición sea correcta. Por ejemplo, para el vector $v = [3, 2, 5, 1, 4]$, la media del segmento de tamaño 3 comenzando en la posición 1 (es decir, la media del subvector $[2, 5, 1]$) es $(2+5+1)/3 = 2.6667$ cuando se escribe con cuatro dígitos tras el punto decimal.
Escribe un procedimiento `print_average_of_segments(v,k)` que, dado un vector v de enteros y un entero positivo k, imprime en la salida estándar (`cout`) la media de todos los segmentos de tamaño k en v, de izquierda a derecha. En el ejemplo anterior de vector con $k = 3$ tenemos los siguientes subvectores (segmentos) de tamaño k: $[3, 2, 5]$, $[2, 5, 1]$, $[5, 1, 4]$. Por lo tanto en este caso el procedimiento imprime 3.3333 2.6667 3.3333.
Piensa una solución que evite cómputos innecesarios. Las soluciones que no tengan esta observación en cuenta tendrán muy probablemente un veredicto de error TLE ("time limit exceeded") del Jutge cuando n y k son números grandes.
Usa el código C++ dado más arriba, completando las partes que faltan. Recuerda que el operador / hace la división entera si sus dos operandos son ambos enteros.
==========
CATALÀ
==========
Donat un vector $v[0], v[1], \ldots, v[n-1]$ d'enters, la mitjana del segment de mida k que comença a la posició i és $(v[i] + v[i+1] + \cdots + v[i+k-1])/k$. Observa que i i k han de satisfer $0 \leq i \leq n-k$ i $k > 0$ perquè la definició sigui correcta. Per exemple, per al vector $v = [3, 2, 5, 1, 4]$, la mitjana del segment de mida 3 començant en la posició 1 (és a dir, la mitjana del subvector $[2, 5, 1]$) és $(2+5+1)/3 = 2.6667$ quan s'escriu amb quatre dígits després del punt decimal.
Escriu un procediment `print_average_of_segments(v, k)` que, donat un vector v d'enters i un enter positiu k, imprimeix a la sortida estàndard (`cout`) la mitjana de tots els segments de mida k a v, d'esquerra a dreta. En l'exemple anterior de vector amb $k = 3$ tenim els següents subvectores (segments) de mida k: $[3, 2, 5]$, $[2, 5, 1]$, $[5, 1, 4]$. Per tant en aquest cas el procediment imprimeix 3.3333 2.6667 3.3333.
Pensa una solució que eviti còmputs innecessaris. Les solucions que no tinguin aquesta observació en compte tindran molt probablement un veredicte d'error TLE ("time limit exceeded") del Jutge quan n i k són nombres grans.
Utilitza el codi C ++ donat més amunt, completant les parts que falten. Recorda que l'operador / fa la divisió entera si els seus dos operands són tots dos enters.
Exam score: 2.5 **Automatic part:** 30%

Input

The input consists of several cases. Each case has a first line with two integers n, k such that $0 < k \leq n$, and a second line with n integer numbers $v_0, \ldots, v_{n-1}$.
==========
CASTELLANO
==========
La entrada consiste en varios casos. Cada caso comienza con una línea con dos enteros n y k tales que $0 < k \leq n$, seguida de una segunda línea con n números enteros $v_0, \ldots, v_{n-1}$.

==========
CATALÀ
==========
L'entrada consisteix en diversos casos. Cada cas comença amb una línia amb dos enters n i k tals que $a0 < k \leq n$, seguida d'una segona línia amb n nombres enters $v_0, \ldots, v_{n-1}$.

Output

For each case, the program outputs the average of all segments of $v_0, \ldots v_{n-1}$ of size k from left to right, followed by a line break. The averages are printed with four digits of precision and each two consecutive values are separated by a blank space.

==========
CASTELLANO
==========
Para cada caso, el programa imprime la media de todos los segmentos de tamaño k del vector $[v_0, \ldots, v_{n-1}]$, de izquierda a derecha, finalizando con un salto de línea. Las medias se imprimen con cuatro dígitos de precisión y cada dos valores consecutivos se separan mediante un espacio en blanco.

==========
CATALÀ
==========
Per a cada cas, el programa imprimeix la mitjana de tots els segments de mida k del vector $[v_0, \ldots, v_{n-1}]$, d'esquerra a dreta, finalitzant amb un salt de línia. Les mitjanes s'imprimeixen amb quatre dígits de precisió i cada dos valors consecutius es separen mitjançant un espai en blanc.

Sample input

```
5 3
3 2 5 1 4
10 3
1 2 3 4 5 6 7 8 9 10
10 5
2 3 4 3 4 2 1 1 0 0
8 3
-1 -2 -2 -1 -1 0 0 1
3 3
1 1 1
6 2
1 -1 1 -1 1 -1
5 1
1 2 3 4 5
```

Sample output

```
3.3333 2.6667 3.3333
2.0000 3.0000 4.0000 5.0000 6.0000 7.0000 8.0000 9.000(
3.2000 3.2000 2.8000 2.2000 1.6000 0.8000
-1.6667 -1.6667 -1.3333 -0.6667 -0.3333 0.3333
1.0000
0.0000 0.0000 0.0000 0.0000 0.0000
1.0000 2.0000 3.0000 4.0000 5.0000
```

Problem information

Author : PRO1
Generation : 2016-05-08 14:58:17