Grado en Ingeniería Informática y Grado en Estadística
Fundamentos de Programación

Examen convocatoria ordinaria. 11 de enero de 2019.

Apellidos _____

Nombre _____ **Grupo** _____

DNI y Firma

Duración del examen: 3,5 horas.	Se valorará la presentación y la claridad en la exposición.
Empezar cada problema en una cara distinta.	Se valorará la adecuación de las estructuras utilizadas al problema a resolver.
Poner nombre y apellidos en todas las páginas.	No se calificarán las respuestas escritas a lápiz.
Las hojas del enunciado también deben entregarse.	

1. Para almacenar las claves de acceso de un determinado sistema se utilizan cadenas de 5 letras minúsculas del alfabeto inglés (no hay tildes, ni diéresis, ni ñ).

a) **[1 pto]** Elaborar un método Java con las siguientes características:

Parámetros de entrada: una cadena de caracteres.

Valor devuelto: la primera letra minúscula del alfabeto inglés que no esté en la cadena de entrada.

Precondición: la cadena de entrada no contiene todas las letras minúsculas del alfabeto inglés.

```
public static char primer_minus (String s){
      char c = 'a';
      while (s.indexOf(c) >= 0)
            c++;
      return c;
}
```

b) **[1 pto]** Elaborar un método Java con las siguientes características:

Parámetros de entrada: una cadena de caracteres.

Valor devuelto: una letra minúscula aleatoria del alfabeto inglés que no esté en la cadena de entrada.

Precondición: la cadena de entrada no contiene todas las letras minúsculas del alfabeto inglés.

```
public static char caract_aleat (char a, char b){
      int rango = b - a + 1;
      int n =(int)(Math.random() * rango) + a;
      return (char)n;
}
public static char aleat_minus (String s){
      char c;
      do
            c = caract_aleat ('a', 'z');
      while (s.indexOf(c) >= 0)
      return c;
}
```

```
//otra forma de hacerlo
public static char aleat_minus (String s){
      char c;
      do
            c = (int)(Math.random() * ('z'-'a'+1)) + a;
      while (s.indexOf(c) >= 0)
      return c;
}
```

c) **[1 pto]**. Elaborar un método Java con las siguientes características:

Parámetros de entrada: un vector de (**n**) cadenas de caracteres.

Valor devuelto: una cadena, de longitud **n**, de letras minúsculas el alfabeto inglés, en la que el primer carácter no pertenece a la primera cadena del vector, el segundo carácter no pertenece a la segunda cadena del vector, y así sucesivamente.

Precondición: ninguna de las cadenas del vector contiene todas las letras minúsculas del alfabeto inglés.

```
//Esta solución usa la función del apartado a
public static String Cadena (String [] v){
      String cad = "";
      for (int i=0; i < v.length; i++)
            cad = cad + primer_minus(v[i]);
      return cad;
}

//También se puede hacer usando la función del apartado b
public static String Cadena (String [] v){
      String cad = "";
      for (int i=0; i < v.length; i++)
            cad = cad + aleat_minus(v[i]);
      return cad;
}
```

2. **[1 pto]** Crear un programa Java que, dado el fichero de texto **`entrada.txt`** que contiene números enteros, escriba en pantalla cuántas veces aparece en el fichero un número dado que se pide al usuario. Supóngase que el fichero de entrada existe y que están importados los paquetes necesarios.

RECORDATORIO: Para declarar y abrir en modo lectura un fichero de texto:

`Scanner <`*`id_fich`*`> = new Scanner (new File(<`*`nombre_fich`*`>));`

Al abrir el fichero se puede producir la excepción **`FileNotFoundException`**

```
public static void main (String[] args){
      Scanner in = new Scanner (System.in);
      System.out.println ("Teclee el número a contar: );
      int num = in.nextInt();
      int cont = 0;
      try {
        Scanner f = new Scanner (new File ("entrada.txt"));
        int n;
        while (f.hasNextInt()){
            n = f.nextInt();
            if (num == n)
                  cont++;
        }
        f.close();
      }
      catch (FileNotFoundException e) {
          System.out.println ("No se puede abrir el fichero");
          return;
      }
      System.out,println ("El número "+num +" aparece " +cont + " veces");
}
```

3. **[1 pto]** Dada **`list`** (de tipo **`Nodo`**), una referencia a una **lista dinámica** que contiene números enteros en su campo **`dato`**, crear un método Java que devuelva cuántas veces aparece en esa lista un número dado (de tipo **`int`**). Supóngase definida la clase Nodo como:

```
public class Nodo {
      int dato;
      Nodo sgte;
      // constructores, etc.
}
```

```
public static int nveces (int num, lista list){
   lista q = list;
   int cont = 0;
   while (q != null){
         if (q.dato == num)
               cont++;
         q = q.sgte;
   }
   return cont;
}
```

4. **[1 pto]** La sucesión Q de Hofstadter se define de la siguiente manera:

Q(1) = 1

Q(2) = 1

Q(n) = Q (n-Q(n-1)) + Q (n-Q(n-2)) si n>2

Escriba una función **recursiva** en Java que devuelva el elemento n-simo de la sucesión Q, con la precondición n>=1

```
public static int Q (int n){
//Precondición: n>=1
      if ((n == 1) || (n == 2))
            return 1;
      else
            return (Q(n-Q(n-1)) + Q(n-Q(n-2));
}
```

5. Para almacenar los valores de un histograma se utiliza un vector de enteros en el que cada posición almacena la altura correspondiente a ese valor, siempre entero positivo.

Por ejemplo, el siguiente vector podría ser un histograma:

0	1	2	3	4	5	6	7	8	9	10
12	4	2	3	4	6	15	19	9	10	12

a) **[2 ptos]** Codificar un método Java que, a partir de un vector con esas características, dibuje en pantalla el histograma horizontal utilizando asteriscos.

Para el ejemplo anterior, la salida mostrada en pantalla debería ser:

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

```
public static void imprimir (int [] hist){
      for (int i=(hist.length - 1); i>=0; i--){
            System.out.print (i);
            if (i<10)
                  System.out.print (" ");
            System.out.print (" ");

            for (j=1; j<hist[i]; j++)
                  System.out.print ("*");

            System.out.println ();
      }
}
```

b) **[1 pto]** Codificar un método Java que, a partir de un vector con esas características, calcule el valor máximo almacenado en ese vector.

```
public static int maximo (int [] hist){
      int max = hist[0];
      for (int i=1; i<hist.length; i++)
            if (hist[i] > max)
                  max = hist[i];
      return max;
}
```

```
//Como todos los valores del histograma son positivos, también sería válido
public static int maximo (int [] hist){
      int max = -1;
      for (int i=0; i<hist.length; i++)
            if (hist[i] > max)
                  max = hist[i];
      return max;
}
```

c) **[2 ptos]** Codificar un método Java que, a partir de un vector con esas características, almacene en una matriz de caracteres blancos y asteriscos de tal forma que represente el histograma vertical.

Para el ejemplo anterior, la matriz generada debería ser:

	0	1	2	3	4	5	6	7	8	9	10
0								*			
1								*			
2								*			
3								*			
4							*	*			
5							*	*			
6							*	*			
7	*						*	*			*
8	*						*	*			*
9	*						*	*		*	*
10	*						*	*	*	*	*
11	*						*	*	*	*	*
12	*						*	*	*	*	*
13	*					*	*	*	*	*	*
14	*					*	*	*	*	*	*
15	*	*			*	*	*	*	*	*	*
16	*	*		*	*	*	*	*	*	*	*
17	*	*	*	*	*	*	*	*	*	*	*
18	*	*	*	*	*	*	*	*	*	*	*

```
public static char[][] histVertical (int [] hist){
      int fil = maximo(hist);
      int col = hist.length;
      char [][] matriz = new char [fil][col];
      int i;

      //Inicialización de la matriz a blancos.
      //Se podría hacer en otro método independiente
      for (int i=0; i<fil; i++)
            for (int j=0; j<col; j++)
                  matriz [i][j] = ' ';

      for (int j=0; j<col; j++){
            i = fil-1;
            for (int cont=1; cont<=hist[j]; cont++){
                  matriz [i][j] = '*';
                  i--;
            }
      }
      return matriz;
}
```