Universidad Simón Bolívar
Departamento de Computación y T.I.
CI2691 – Laboratorio Algoritmos y Estructuras I
Septiembre - Diciembre 2009

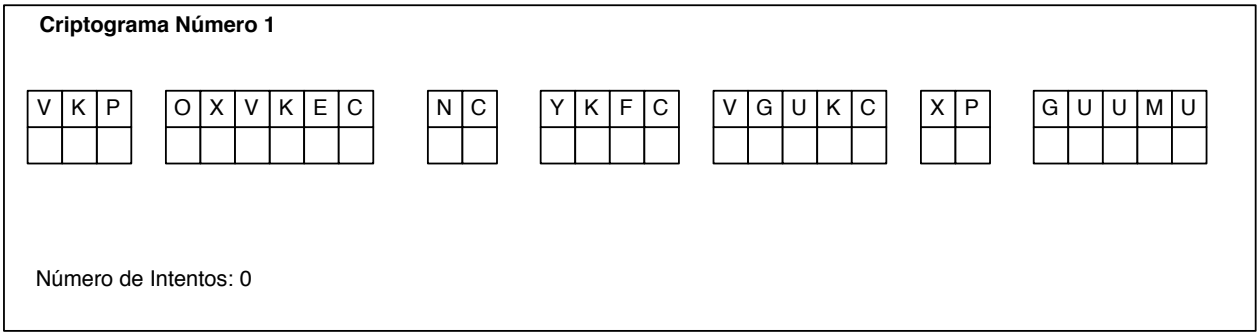
Proyecto
Criptogramas

Se desea que usted diseñe e implemente un programa en Gacela que permita a una persona descubrir mensajes cifrados.

1. Descripción y Reglas del Juego

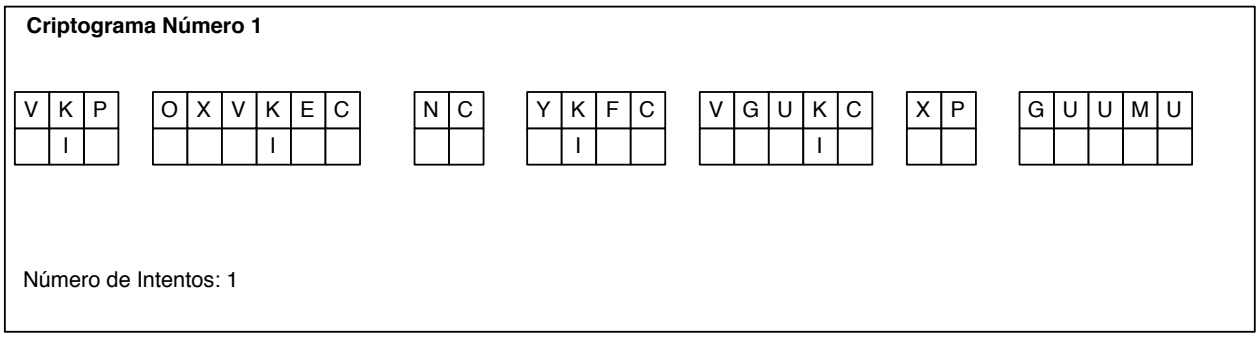
Según http://es.wikipedia.org/wiki/Criptograma, un criptograma es un "mensaje cifrado cuyo significado resulta ininteligible hasta que sea descifrado. Generalmente, el contenido del mensaje original es modificado siguiendo un determinado patrón, de manera que sólo es posible comprender el significado original tras conocer el patrón seguido en el cifrado."

En esta implementación una persona jugará a descifrar uno o más mensajes, cada uno de longitud N incluyendo espacios en blanco, con un máximo de 80 caracteres. El mensaje puede haber sido cifrado por la computadora usando el método que se detallará más adelante. Cuando el jugador inicia el juego aparecerá la siguiente ventana:

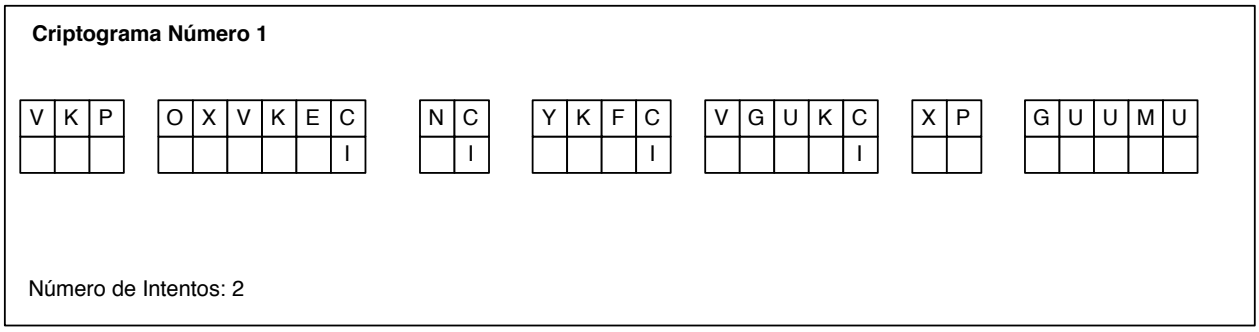


La persona puede descifrar el mensaje, determinando el patrón de cifrado, o simplemente adivinando lo que dice el mensaje luego de que tenga varias letras descifradas.

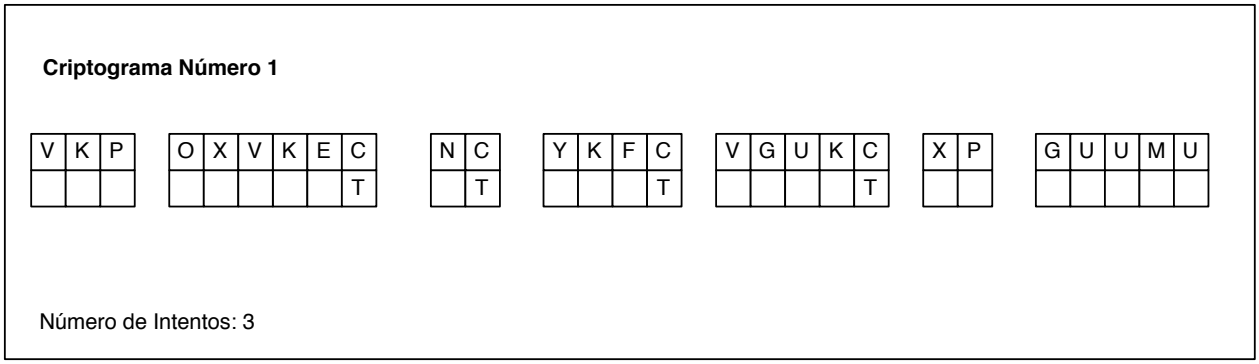
Para cada jugada se indica la posición del mensaje y la letra identificada, suponiendo que el mensaje tiene longitud N (incluyendo los espacios en blanco). En el ejemplo, el mensaje tiene longitud 33. Según la posición (entre 0 y N-1) y letra jugada (no necesariamente la letra correcta), el programa debe rellenar todas las posiciones en donde se encuentra la misma letra. En este ejemplo, si el jugador indica para la posición 7, una I, la ventana se refrescaría de la siguiente manera:



Cabe destacar que el jugador puede jugar una letra ya jugada anteriormente, pero en otra posición. En ese caso se dehace la asignación anterior y se coloca la nueva asignación. Si por ejemplo, se juega ahora la I en la posición 9, se tendría la siguiente ventana:



Igualmente, el jugador puede jugar una posición ya jugada anteriormente, en ese caso se deshace la asignación anterior, y se sustituye por la nueva asignación:



En cada jugada se tienen las siguientes opciones:

- Jugar una posición y letra
- Adivinar el mensaje
- Rendirse

El descifrado termina si el jugador decide adivinar el mensaje, si ha llenado todas las posiciones o si se rinde. En todo caso el programa muestra la solución.

Cada vez que se termine el descifrado de un mensaje, se podrá descifrar un nuevo mensaje o salir del juego completamente.

El método de cifrado es un Cifrado de Sustitución Simple en donde existe una matriz de encriptamiento de dimensiones Mx27 en donde cada fila corresponde a una permutación del alfabeto. El proceso de cifrado se realiza generando un número aleatorio entre 0 y M-1, y escogiendo de la matriz de encriptamiento la permutación (fila) correspondiente al número generado. El cifrado se realizará entonces según esa permutación.
Sin embargo, cabe aclarar que en su proyecto, solo tendrá que realizar el proceso de descifrado de la forma que le indicaremos más adelante.

El programa debe tener las siguientes funcionalidades:

a) Al comenzar, debe preguntar el nombre de la persona.
b) Debe permitir descifrar varios mensajes .
c) Al finalizar un descifrado, la persona puede decidir si desea continuar jugando.
d) El mensaje cifrado debe estar desplegado como un tablero de 2xN en donde N es la longitud del mensaje incluyendo espacios en blanco.
e) El mensaje cifrado debe ser obtenido de un archivo llamado *mensajesCifrados.txt*, y el programa lo debe descifrar utilizando la matriz de encriptamiento dada en el archivo *MatrizEnc.txt*.
f) Cada jugada de la persona deberá reflejarse gráficamente en la ventana. También debe reflejarse el número de intentos.
g) La persona puede decidir pedir la solución y abortar el descifrado de ese mensaje.
h) Al finalizar el juego se debe imprimir el número de mensajes jugados, y el número de mensajes descifrados correctamente por la persona.

2. Proceso de lectura de mensajes y descifrado

Para la lectura y despliegue de mensajes, y el proceso de descifrado se utilizarán archivos de texto que usted podrá manejar utilizando las funciones predefinidas para archivos indicadas en la página Wiki. Se utilizarán dos archivos:

- Los mensajes cifrados se obtendrán del archivo *mensajesCifrados.txt* que se encuentra en la página Wiki. La primera línea del archivo contendrá el número total de mensajes T, y luego para cada mensaje se tendrán dos líneas:
  - Entero entre 0 y M-1 que corresponde a la permutación del alfabeto utilizada (fila en la matriz de encriptamiento Mx27).
  - Mensaje propiamente dicho.

  Cada vez que el usuario quiera descifrar un nuevo mensaje deberá leerlo del archivo y desplegarlo en la pantalla. Debe suponer que el archivo ha sido formateado correctamente.
- La matriz de encriptamiento se obtendrá de un archivo llamado *matrizEnc.txt*. La primera línea del archivo contendrá el número de permutaciones M, y cada línea subsiguiente contendrá un número entero correspondiente a un elemento *matrizE[i,j]* de la matriz. La matriz será almacenada por filas. La matriz de encriptamiento es una matriz de enteros debido a que el proceso de descifrado se simplifica si se utilizan números enteros y no letras como se detallará a continuación con un ejemplo. Debe suponer que el archivo ha sido formateado

correctamente.

Suponga que se tiene un alfabeto de cinco letras [a,b,c,d,e], y la matriz de encriptamiento con M=4 es:
1 0 4 3 2
3 1 0 2 4
0 1 4 3 2
1 3 2 0 4

Suponga que se tiene el siguiente mensaje cifrado : "cbde", y el número de la permutación utilizado es el correspondiente a la fila 3 de la matriz de encriptamiento, es decir, la permutación utilizada para el cifrado es [1 3 2 0 4]. El descifrado se obtiene de la siguiente manera:

- Se convierte el mensaje cifrado de letras a números de acuerdo a la posición de la letra en el alfabeto (a es el número 0), de esta manera el mensaje "cbde" corresponde a 2 1 3 4.
- Luego se toma la fila 3 de la matriz [1 3 2 0 4], y se intercambia el contenido con la posición, así la fila 3 de de *matrizE* se convierte a [3 0 2 1 4] (el 0 está en la posición 3 y se coloca un 3 en la posición 0, el 1 en la posición 0 y se coloca un 0 en la posición 1, y así sucesivamente).
- Se usa esta nueva permutación para descifrar, así se tiene el mensaje descifrado en números 2 0 1 4.
- Se convierte el mensaje de número a letras, y se obtiene el mensaje descifrado "cabe".

3. Solución General

El proyecto involucra dos componentes: uno de lógica y uno de interfaz. Comenzamos con el componente lógico. Lo primero que debemos hacer es tener una visión general de la solución. Podemos aplicar análisis descendente para obtener una primera versión:

```
cargar matriz encriptamiento
do descifrar otro mensaje →
   Obtener mensaje cifrado;
   Descifrar el mensaje;
   Inicializar ventana mensaje cifrado;
   do el juego no ha terminado →
                obtener la opcion;
                if opcion es letra → jugar la letra y actualizar numero intentos
                [] opcion es adivinar → obtener la adivinanza y terminar
                [] opcion es rendirse → terminar
                [] opcion no valida →  Error
                fi
   od;
     if ganadorJuego → contabilizar mensajes ganadores
     [] ¬ ganadorJuego → presentar solucion
od;
Desplegar resultados de conjunto de mensajes
```

A partir del análisis descendente, se pueden identificar los sub-problemas más importantes:
cargar la matriz de encriptamiento, determinar si se desea descifrar otro mensaje, obtener el mensaje

cifrado, inicializar la ventana, determinar si el juego ha terminado, obtener la opción, jugar la letra, obtener la adivinanza, saber si el usuario ganó el juego, contabilizar los mensajes ganadores y presentar la solución.

4. Estructuras de datos

Se tendrán tres arreglos de longitud N para representar el mensaje cifrado, el mensaje descifrado y el mensaje "adivinado"

- *mensajeC array[] of String* para el mensaje cifrado.
- *mensajeD array[] of String* para el mensaje descifrado.
- *mensajeA array[] of String* para el mensaje adivinado. Cada casilla de *mensajeA* podrá tener como valores:
  - la letra (correcta o no) que vaya adivinando la persona
  - espacio en blanco para los espacios en blanco entre palabras
  - "&" para las letras que aún no hayan sido adivinadas.

La matriz que será leida del archivo *matrizEnc.txt* debe ser cargada en una matriz Mx27 llamada *matrizE*. Esta estructura será modificada durante el procedimiento de descifrado. Cada vez que se tenga un mensaje que ha sido cifrado con la permutación correspondiente a una determinada fila de la matriz, ésta sera sustituida por su fila "inversa", es decir aquella que resulta de convertir de contenido a posición tal como se ilustró en el ejemplo.

Si se descifra un mensaje mediante una permutación que ya haya sido usada anteriormente en el descifrado de algún otro mensaje, la fila correspondiente ya contiene la permutación "inversa", por lo cual no hay que invertirla. Para tener control de cuales permutaciones ya se han utilizado anteriormente, se mantendrá un arreglo de booleanos *permUtilizada* de longitud M, y que contiene valor verdadero si la permutación ya ha sido utilizada.

Se tendrá un arreglo *alfabeto* de longitud 27, que contendrá todas las letras del alfabeto, y que permitirá la conversion de números a letras y viceversa y adicionalmente un arreglo de enteros *jugada* de longitud 2 que contendrá la posición y letra de la jugada, note que la letra se representa como un entero de acuerdo a su posición en *alfabeto*. Además se debe tener una variable *intentos* para el número de intentos en cada juego.

5. Sub-problemas

A continuación se describirán los subproblemas del proyecto y se darán las definiciones de los procedimientos y funciones que deben ser utilizados en su desarollo.

5.1. Cargar la matriz de encriptamiento

Consiste en leer el archivo *MatrizEnc.txt* y cargar los datos en la estructura *matrizE*. La primera dimensión de *matrizE* se leerá de la primera línea del archivo. La segunda dimensión es 27.

proc *obtenerMatriz:(****out*** *matrizE:array[][]* ***of int****)*

5.2. Determinar si desea descifrar otro mensaje

Debe verificar si el jugador desea descifrar otro mensaje mediante un mensaje en pantalla.

5.3. Obtener el mensaje cifrado

Obtener el mensaje cifrado *mensajeC* del archivo *mensajesCifrados.txt*, y determinar el número de la permutación (fila de la *matrizE* utilizada para cifrarlo. El *idArchivo* es un identificador que se le asigna al archivo *mensajesCifrados.txt* luego de que se abre. Se definen una función y un procedimiento para resolver este subproblema:

- Obtener el número de la permutación (fila de la *matrizE* utilizada para cifrarlo).
  func** obtenerFilaP:(**in** idArchivo:**String**)→ **int

- Cargar el mensaje en el arreglo *mensajeC.*
  ***proc** obtenerMensaje:(**in** idArchivo:**String , out** mensajeC:**array[] of String**)*

5.4. Descifrar el mensaje

Descifrar el mensaje cifrado *mensajeC* y guardar el mensaje descifrado en *mensajeD* utilizando el proceso descrito anteriormente en donde se utilizan el *alfabeto* para realizar la conversión de letra a número y viceversa y *filaP* para la permutación de la matriz de encriptamiento *matrizE* que fue utilizada para encriptar el mensaje. *M* es el número de permutaciones distintas (número de filas) de la matriz de encriptamiento y *N* es la longitud del mensaje. Se definen los siguientes procedimientos y funciones:

- Si la fila de la matriz de permutación no ha sido "invertida" anteriormente, realizar la "inversión" (intercambio de posición por contenido)
  ***proc** invertirP:(**in** M**:int**, **in** filaP:int, **in out** matrizE:**array[][][] of int**)*

- Convertir al mensaje cifrado *mensajeC* de letras a números, descifrar el mensaje ya convertido a números utilizando la matriz de encriptamiento, convertir el mensaje descifrado de números a letras y colocarlo en *mensajeD*.

  ***proc** descifrar:(**in** N:**int, in** M**:int, in** matrizE:**array[][][] of int, in** filaP:**int**,*
  in** alfabeto:**array[] of String, in** mensajeC:**array[] of String
  ***out** mensajeD:**array[] of String**)*

5.5. Determinar si el juego no ha terminado

Debe determinar si se han llenado todas las posiciones del mensaje, o si el jugador decidió rendirse o si trató de adivinar el mensaje, esto último se puede conocer mediante la *opcion* introducida por el usuario.

func** juegoTerminado:(**in** opcion:**int**, **in** N**: int, in** mensajeA:**array[] of String**) → **boolean

5.6. Obtener la opción

Debe obtener la opción de juego indicada por el usuario

5.7. Jugar Letra

Se divide en los siguientes subproblemas

- Obtener la posición y letra de la jugada. La letra debe convertirse a número utilizando

alfabeto antes de actualizar la jugada.
func *obtenerJugada:(****in*** *alfabeto:array[] of* ***String)*** *→* ***array[] of int****;*

- Si la jugada es válida, reflejarla en *mensajeA* y reflejar la jugada en pantalla (esto último se tratará en la próxima sección). El reflejo de la jugada en *mensajeA* debe incluir colocar la letra en todas las posiciones en donde se encuentre la misma letra y deshacer una asignación anterior en caso de que el jugador juegue una posición o letra ya jugada.
  proc *reflejarMensaje:(****in*** *N:* ***int, in out*** *mensajeA:****array[] of String,***
  in *alfabeto****:array[] of String, in*** *jugada****:array[] of int)***

- Si no es válida dar un mensaje de error y permitir ingresar nuevamente la jugada.

5.8. Obtener la adivinanza

Debe pedirle al usuario su adivinanza del mensaje, y colocarla en *mensajeA*. Se debe validar que la longitud del mensaje introducido sea *N*, en caso de que no sea válido se le pedirá nuevamente.
proc *obtenerAdivinanza:(****in out*** *mensajeA:****array[] of String)***

5.9. Saber si el usuario ganó el juego

Se gana el juego si *mensajeD* y *mensajeA* coinciden.
func *gano:(****in*** *mensajeD* ***array[] of String, in*** *mensajeA* ***array[] of String)*** *→* ***boolean***

5.10. Contabilizar mensajes ganadorss

Debe acumular el número de mensajes que se descifraron.

5.11. Presentar solución

Debe presentar el mensaje descifrado en pantalla debajo del “tablero” del mensaje.

5.12. Desplegar resultados

Se debe mostrar en pantalla los totales de mensajes descifrados y no descifrados.

NOTA IMPORTANTE
Usted podrá desarrollar mejoras y extensiones al esqueleto general y los subproblemas indicados anteriormente siempre y cuando se describan y documenten con claridad.

6. Interfaz

Con las actividades anteriores hemos diseñado un programa que captura la parte lógica o de cálculo del problema. Necesitamos ahora instrucciones que manejen la parte de interfaz. Para facilitar la programación vamos a limitar las operaciones de entrada de información por teclado al modo texto. Para el despliegue de información utilizaremos dos modos de manera conjunta: modo texto y modo

gráfico.

Para la salida gráfica del programa se utilizará la Maquina de Trazados, con la que es posible dibujar el "tablero" del mensaje.

El primer procedimiento necesario es *inicializarVentana* , que dibuja la ventana inicial con el mensaje cifrado. Tiene la siguiente forma:

```
proc inicializarVentana:(in out pantalla:Screen,
                         in xsi : int,
                         in ysi : int,
                         in N:int,
                         in mensajeC : array[] of String)
```

donde *N* es la longitud del mensaje, *pantalla* es la máquina de trazados y *xsi,ysi* las coordenadas de la esquina superior izquierda del "tablero" del mensaje.

Cada vez que se realice una jugada se debe modificar gráficamente la ventana correspondiente.

```
proc reflejarPantalla:(in out pantalla: Screen,
                       in xsi : int,
                       in ysi : int,
                       in N:int,
                       in intentos:int,
                       in mensajeA : array[] of String
                       in mensajeC : array[] of String )
```

Note que con este procedimiento se debe "refrescar" la pantalla completa, es decir debe borrar el contenido de la pantalla y escribir nuevamente el contenido de *mensajeC*, las letras que contiene *mensajeA*, y el número de intentos de *intento.*

7. Entregas

El proyecto se desarrollará de forma incremental. Para cada una de las entregas debe especificar las precondiciones y las postcondiciones así como los invariantes representativos de los programas y subprogramas que se soliciten.

7.1. Primera entrega

Las siguientes funciones deben ser desarrollados como programas para esta entrega y debe **probarse su correctitud**. Los parámetros de entrada serán declarados como constantes y los parámetros de salida como variables:

- *juegoTerminado*
- *gano*

Fecha entrega: Martes de la semana 9 antes de la 1:30.
Forma entrega: Listado impreso y envío por correo electrónico.

7.2. Segunda entrega

En esta entrega deberá realizar lo siguiente:

- Los programas de la entrega anterior deben ser ahora desarrollados como funciones
- Desarrollar todos los procedimientos y funciones restantes excepto las referentes a la obtencion de los mensajes y de la matriz de encriptamiento mediante la lectura de archivos (subproblemas 5.1 y 5.3) y los de la interfaz (sección 6).
- Los procedimientos de las secciones 5.1, 5.3 y 6. serán "dummy", es decir estarán definidos pero no se desarrollarán aún.
- Desarrollar el esqueleto general del programa con las llamadas a los procedimientos y funciones.

Fecha entrega: Martes de la semana 10 antes de la 1:30.
Forma entrega: Listado impreso y envío por correo electrónico.

7.3. Entrega final

Para la entrega final usted deberá completar el componente de lectura de archivos y de interfaz del proyecto e integrarlo con el componente lógico.
Usted deberá llevar una versión operativa del programa en Gacela según las especificaciones indicadas. Es importante que el equipo trabaje de manera integrada. Durante la revisión se verificará el funcionamiento del programa y se hará un corto interrogatorio particular a ambos miembros del equipo. Además, deberá entregar un informe utilizando el siguiente formato:

Portada. Una hoja que contenga:
Arriba a la izquierda el nombre de universidad, carrera y materia.
Centrado: nombre del proyecto.
Abajo a la izquierda: nombre del profesor de laboratorio.
Abajo a la derecha: nombre y carnet de los integrantes del equipo

Introducción:
Breve descripción del problema atacado en el proyecto.
Objetivos planteados.
Alcance de la solución.
Contenido del informe.

Diseño:
Resultado del análisis descendente del problema.
Extensiones a los subproblemas planteados.
Estructuras de datos utilizados.
Estado actual.

Operatividad del programa: decir si funciona perfectamente o no. En caso negativo, describir las anomalías.

Manual de operación: nombre del archivo a ejecutar y modo de operar el programa.

Conclusiones.
Resultados obtenidos.

Experiencias adquiridas.
Dificultades presentadas.

Recomendaciones.

Bibliografía. Libros, revistas o cualquier recurso bibliográfico consultado. Una referencia bibliográfica debe incluir la siguiente información:

Libro: autor(es), nombre, editorial y año.
Artículo: autor(es), nombre, revista, volumen, número y año.
Página Web: autor(es), nombre, url, fecha última visita.

Los siguientes son ejemplos de referencias bibliográficas:

Libro:
Blair, D.: Language and Representation in Information Retrieval. Elsevier Science Publisher, 1990.
Revista:
Isaacson, M.: What is SMDI?, Electronic Musician 9(6). p.79-82
Página Web:
Sun Microsystems: The Java Tutorial. http://java.sun.com. Noviembre 2000.

Fecha entrega: Martes de la semana 11 antes de la 1:30
Forma entrega: Informe según la estructura sugerida anteriormente, listado impreso debidamente documentado e indentado, y CD debidamente identificado.