Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos
Proyecto del curso, 2015-10
Primera Parte

1 Descripción del problema

En general, la información genética de un organismo se encuentra contenida en su genoma. Para manipular la información del genoma de un organismo cualquiera, se usan secuencias de información conocidas como códigos genéticos. Estos códigos genéticos están compuestos de letras que representan las cuatro bases nitrogenadas del ácido desoxirribonucleico (ADN) o del ácido ribonucleico (ARN). Mientras en el caso del ADN las bases son adenina (A), timina (T), guanina (G) y citosina (C); para el ARN las bases son adenina (A), uracilo (U), guanina (G) y citosina (C).

Aunque hay muchos avances relacionados con el descubrimiento de los códigos genéticos de los organismos del planeta, mucho trabajo aún queda por hacer. Para guardar la información que se conoce (y señalar la que se desconoce) se usa un formato de códigos genéticos conocido como FASTA.

2 Descripción del proyecto

El objetivo del presente proyecto es construir un sistema que permita algunas manipulaciones sencillas sobre archivos que contienen códigos genéticos

2.1 Formato FASTA

Los archivos en formato FASTA (extensión .fa) son basados en texto. Pueden contener uno o varios códigos genéticos que componen un genoma. Cada código genético empieza con una línea de texto que tiene el formato:

```
>[descripcion de la secuencia]
```

Debe notarse que siempre empieza con el caracter '>' y justo después se presenta una cadena no vacía que describe la secuencia. Las siguientes líneas (hasta la siguiente secuencia o hasta el final del archivo) contienen los datos que describen el código genético. Estas líneas están justificadas y todas tienen un ancho constante, con la posible excepción de la última línea. Un ejemplo de un archivo FASTA puede ser:

```
>Full_SEQUENCE
CTCCGGTGAGAAATTTTGGGATGTATCAAATCACGGTCCTACTAC
TTACCGCCAACACGAGCCGGAACCCCCTAGATCAATTCATGCTTT
TCCCTTCACGCGAAGGAGTCGGAAGTGATCTGTATGAAGCTATTA
CCCTAGGTGGCCACACCTAC
>Incomplete_sequence
URDNSHVNKSCUANADDDVMMDVHRSRGNUNTSBTTMCGNBUUBTUMNAYKSTRYMVBWVNSC
SUUDDYVBCGNDRUURUBDHVGTYAVAVVSAKHUTSWCUBUUMSMDVDKBWRHHBHDBDWUDC
CAUWBRNYSTVTBBCKGDCMSNTKBNTRTNYRNWNWNVCSCNDDWHUHWRRTUMWNKHUBDWA
DUNYUVKHNKSDCYAVARTUWDVSTDVVMMVUHVBCDGVBSKKBHKSVHHGKSAADDVBRKSS
UKURTKTNAWYBKVRRBGKGBMGKUCGSMDTDTKCKVUYNVDWRWNBRCGKDWWUNBYMTGBN
YTAHCUTAGNBKWWGNDRKMNCDVDTKNRGVBRVMKGBKUBGBRHVUSTBCGDV
```

El signifcado de cada letra se explica en la Tabla 1.

2.2 Primera entrega

A continuación se describen los componentes individuales que conforman la primera entrega del presente proyecto. El sistema se implementará como una aplicación que recibe comandos textuales.

Código	Significado
A	Adenina
C	Citosina
G	Guanina
T	Timina
U	Uracilo
R	A o G
Y	C, T o U
K	G, T o U
M	A o C
S	C o G
W	A, T o U
B	C, G, T o U
D	A, G, T o U
H	A, C, T o U
V	A, C o G
N	A, C, G, T o U
X	Máscara
-	Espacio de longitud indeterminada

Table 1: Relación de cada código del formato FASTA con su significado biológico (tomada de http://www.ncbi.nlm.nih.gov/blast/fasta.shtml).

2.2.1 Componente 1: Resumen de la información de un genoma

Objetivo: A partir de un archivo FASTA, mostrar por pantalla la información que el usuario requiera. Este componente se implementará con los siguientes comandos:

- **comando:** `load <filename>`
  salida en pantalla:
  (Archivo vacío) ``filename'' does not contain any sequence.
  (Una sola secuencia) 1 sequence successfully loaded from ``filename''.
  (Varias secuencias) <n> sequences successfully loaded from ``filename''.
  descripción: Carga en memoria los datos contenidos en el archivo identificado por <filename>.

- **comando:** `count`
  salida en pantalla:
  (No hay secuencias cargadas) No loaded sequences.
  (Una sola secuencia) 1 sequence loaded.
  (Varias secuencias) <n> sequences loaded.
  descripción: Imprime por pantalla la cantidad de secuencias cargadas en memoria.

- **comando:** `list_sequences`
  salida en pantalla:
  (No hay secuencias cargadas) No loaded sequences.
  (Secuencia completa de archivo no vacío) Sequence ``sequence_description'' contains <b> bases.
  (Secuencia incompleta de archivo no vacío) Sequence ``sequence_description'' contains at least <b> bases.
  descripción: Imprime en n líneas (una para secuencia) la información básica (cantidad de bases) de cada secuencia. Si la secuencia es completa (i.e. no tiene el código '-') imprime el segundo mensaje, si no, el tercero.

- **comando:** `histogram <sequence_description>`
  salida en pantalla:
  (La secuencia no existe) Invalid sequence.
  (La secuencia existe) A : <frequence_A> \n C : <frequence_C> \n ...

descripción: Imprime el histograma de una secuencia, en caso de que exista. El histograma se define como el conteo (frecuencia) de cada código en la secuencia. Por cada línea ('\n' es el caracter de salto de línea) se escribe el código y la cantidad de veces que aparece en la secuencia. El ordenamiento del histograma está dado por la Tabla 1.

- **comando:** `is_subsequence <sequence>`
  salida en pantalla:
  (No hay secuencias cargadas) `No loaded sequences.`
  (La secuencia no existe) `The given sequence does not exists.`
  (Varias secuencias) `The given sequence is repeated <s> times.`
  descripción: Determina si una secuencia, dada por el usuario, existe dentro de las secuencias cargadas.

- **comando:** `mask <sequence>`
  salida en pantalla:
  (No hay secuencias cargadas) `No loaded sequences.`
  (No se enmascararon subsecuencias) `No masked sequences.`
  (Una subsecuencia enmascarada) `1 sequence has been masked.`
  (Varias subsecuencias esmascaradas) `<s> sequences have been masked.`
  descripción: Enmascara una secuencia dada por el usuario, si existe. Los elementos que pertenecen a la subsecuencia se enmascaran cambiando cada código por el código 'X'.

- **comando:** `save <filename>`
  salida en pantalla:
  (No hay secuencias cargadas) `No loaded sequences.`
  (Escritura exitosa) `Sequences saved successfully in ''filename''.`
  (Problemas en archivo) `Error writing to ''filename''.`
  descripción: Guarda en el archivo <filename> las secuencias cargadas en memoria. Se debe tener en cuenta la justificación (de líneas) del archivo inicial.

- **comando:** `exit`
  salida en pantalla:
  (No tiene salida por pantalla)
  descripción: Termina la ejecución de la aplicación.

2.3 Interacción con el sistema

La interfaz de la aplicación a construir debe ser una consola interactiva donde los comandos correspondientes a los componentes serán tecleados por el usuario. El indicador de línea de comando debe ser el caracter '$'. Para cada comando, se debe incluir una ayuda de uso que indique la forma correcta de hacer el llamado (i.e. el comando '`help <command>`' debe existir, además de los descritos anteriormente). El comando debe presentar en pantalla los mensajes de resultado especificados antes. Los comandos de los componentes deben ensamblarse en un único sistema (es decir, funcionan todos dentro de un mismo programa, no como programas independientes por componentes).

3 Evaluación

Las entregas se harán en la correspondiente actividad de Uvirtual, hasta la media noche del día indicado. Se debe entregar un archivo comprimido (únicos formatos aceptados: .zip, .tar, .tar.gz, .tar.bz2, .tgz) que contenga documentos (único formato aceptado: .pdf) y código fuente (.h, .hxx, .hpp, .c, .cxx, .cpp). Si la entrega contiene archivos en cualquier otro formato, será descartada y no será evaluada, es decir, la nota definitiva de la entrega será de 0 (cero) sobre 5 (cinco).

3.1 Entrega (lunes 23 de marzo de 2015)

Componente 1 completo y funcional. Esta entrega se compone de:

- (30%) Documento de diseño. El documento de diseño debe seguir las pautas de ingeniería que usted ya conoce: Diagrama de clases (si ya vio ADOO o POO) o diseño precondiciones, poscondiciones e invariantes (si no ha visto ADOO). Además, se exigirá un(os) esquemático(s) que resuma(n) el(los) problema(s) y su(s) solución(ones) gráficamente.
- (70%) Código fuente compilable en el compilador gnu-g++ (versión 4.0.0, como mínimo). Este porcentaje de la entrega será un promedio de la evaluación de cada comando.