

GENOME REARRANGEMENTS

In molecular biology, an organism is distinguished and identified by the sequences of genes found in its chromosomes. The set of chromosomes for a particular organism or species is called the genome of that species. Genes are unique within a genome—there are no duplicate genes in a genome even across chromosomes. Species that are closely related often have identical genes but differ in the placement and order of genes in their chromosomes. Mutations that may have occurred during evolution for these species could then be viewed as *rearrangements* of genes. The most common rearrangements are reversals (a subsequence of genes in a chromosome is inverted) and translocations (two subsequences in the genome are swapped).

Write a C++ program that models and simulates the rearrangements of genes in a genome. The program is called **mutate** and is invoked through the Unix shell command line in the following manner:

mutate <n> <inputfilename> <outputfilename>

Here, <n> stands for the number of chromosomes in the genome and <inputfilename> is the name of the text file containing a sequence of commands that specify the chromosomes in a genome or the mutations that will occur. Five types of commands are possible, and these are described below:

chromosome <i> <gene-string>

assigns (or reassigns) <gene-string> to the <i>th chromosome of the genome. <i> is a value from 1 to <n>. <gene-string> is a string of upper and lower case letters (52 possible genes). Recall that genes do not duplicate so the string should not contain two of the same character; in addition, if the character already exists in some other chromosome, it should not be present in <gene-string>.

reverse <i> <end-gene-1> <end-gene-2>

reverses, in chromosome <i>, the gene subsequence bounded by the characters <end-gene-1> and <end-gene-2>. The characters should, of course, exist in the chromosome, but they could be listed in the command in any order; e.g., **reverse 3 x C** will produce the same effect as **reverse 3 C x**.

translocate <i> <end-gene-1> <end-gene-2> <i'> <end-gene-1'> <end-gene-2'>

swaps the subsequence bounded by <end-gene-1> and <end-gene-2> in chromosome <i> with the subsequence bounded by <end-gene-1'> and <end-gene-2'> in chromosome <i'>. The orders of the bounding characters are important as they specify the orientation of the subsequences in the resulting genome; e.g., **translocate 2 x B 3 A Z** is equivalent to **translocate 2 B x 3 Z A** but not the same as **translocate 2 B x 3 A Z**. Also, note that it is possible for two subsequences in the same chromosome to be translocated, as long as the subsequences do not overlap.

print chromosome <i>

prints the gene sequence in chromosome <i>

print genome

prints the gene sequences for all chromosomes in the genome

A single space separates the arguments in a command.

The program will process all the commands in the input file, in sequence, and provide output after each command is carried out. Output will be sent to a text file named **<outputfilename>** (as indicated in the Unix shell command line). For the **chromosome**, **reverse**, and **translocate** commands, one-line messages that describe the command performed will be printed. Refer to the following example for output formats:

Sample Input:	Resulting Output:
chromosome 1 QuIckLY	1: QuIckLY
chromosome 2 BrowN	2: BrowN
chromosome 3 FOxeS	3: FOxeS
reverse 1 c L	ckL reversed
print chromosome 1	1: QuILkcY
translocate 2 r N 3 x O	rowN and xO swapped
print genome	1: QuILkcY
chromosome 1 nicE	2: BxO
reverse 3 e o	3: FNworeS
print genome	1: nicE
translocate 1 i c 3 w o	ero reversed
print chromosome 1	1: nicE
print chromosome 3	2: BxO
	3: FNweroS
	ic and wero swapped
	1: nweroE
	3: FNicS

Follow the output formats closely. Note that the orientation of the subsequences as they are printed depends on an operation on the order of the bounding characters specified in the operation.

Logical errors in commands should be detected and reported on a per line basis; in such a case, the command is not carried out. Parsing errors (incomplete arguments, invalid characters, badly formed numbers) need not be detected—you may assume that each command has the correct keywords, spacing, and format for its arguments. The following are the errors that should be detected and reported:

- Non-existent chromosome sequence (**<i>** not within the range 1 to **<n>**)
- Genes should be unique across chromosomes (for the **chromosome** command)
- Gene does not exist in sequence (when specifying subsequences for **reverse** and **translocate**)
- Sequences overlap (for the **translocate** command)

This assignment is an exercise in parsing simple input, using an elementary linear data structure (the linked list), proper OO software design and use of C++ classes (as you have learned from CS 2704), and logical error handling. You are required to use lists, either singly-linked or doubly-linked lists, when implementing the sequence of genes in a chromosome (a gene character is stored in a linked list node). However, you may use an array to hold the **<n>** chromosomes of a genome. You are also required to maintain a free list of nodes for garbage collection (important for the **chromosome** command, since a redefinition discards the sequence previously assigned to the chromosome). You may not use the STL or similar libraries but you may pattern your code according to the code provided in the textbook.

A longer test input file as well as the corresponding output file will be available on the course website by Monday, 7 July 2003. The program is due on 9 July 2003 5:00 pm. You are to submit a tarred, gzipped file (.tar.gz) containing all source code and a makefile. Make sure that your submission successfully compiles and executes in the Mandrake Linux environment. Consult the course web site for Curator system instructions, program documentation guidelines, honor code policies, and any further clarifications for this assignment.

Pledge: Your project submission must include a statement pledging your conformance to the Honor Code requirements for this course. Specifically, you must include a pledge statement in the header comment preceding the main() function of your program. The text of the pledge will be posted on the website.