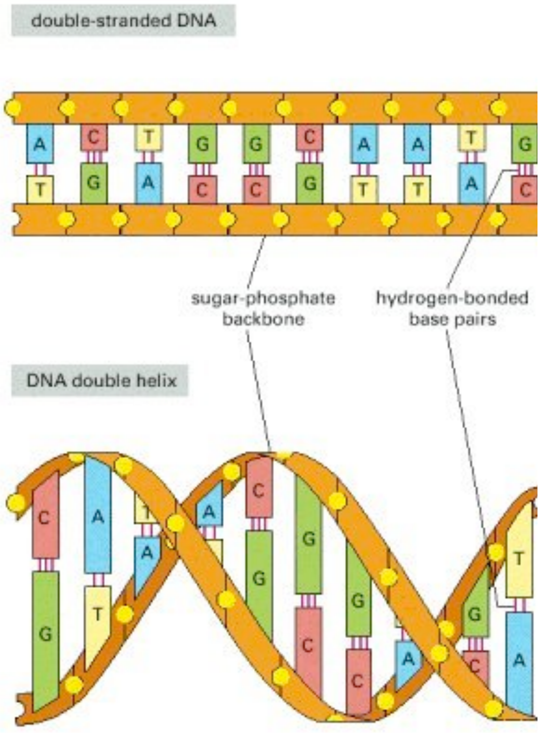
DNA - The Strands of Life

String and Array Manipulation Assignment

Bacteria are everywhere. Some of them are very nasty! Pick your favourite bacteria, access its DNA from the database, complement it and reverse it.



DNA Backgrounder …

A DNA strand has an *orientation*: Its two ends are called the 5 prime end and the 3 prime end (written 5' and 3'), respectively. The translation machinery of the cell, which turns DNA into protein, always reads the strand from the 5'-end. When two strands are joined to form a double helix, their orientations are always opposite.

Within each strand, neighboring nucleotides are held together by strong, chemical covalent bonds. In a double helix, two nucleotides on opposite strands are joined together by a weaker hydrogen bond. Between strands, A's join with T's, and C's join with G's. The two strands in a double helix are *complementary* in the sense that from one of them, the other can be deduced: One simply reverses the sequence of nucleotides and *complements* it, i.e. substitutes all A's with T's, T's with A's, C's with G's and G's with C's. See the top part of the figure.

Adapted from:
www.daimi.au.dk/~besen/PiB/exercises/reversecomplement.html

Assignment Instructions:

1. **Go to National Center for Biotechnology Information (NCBI) and download a bacterial DNA file. Below is an example for salmonella.**

ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/Salmonella_enterica_serovar_Typhimurium_LT2_uid57799/NC_003197.fna

2. **Write a program that:**

    a. **Reads in the DNA file**
    b. **Complement each letter (conditionals) – ex. A<->G C<->G**
    c. **Reverse the letters of each line**
    d. **Write out the reverse complement file**
    e. **Code should be efficient (e.g., no memory leaks)**
    f. **Employ good coding practices throughout**
    g. **Add a timer to your program to print out processing time**
    h. **Your output file should be 69393 lines with 4926947 characters**

    I will be running a Unix diff command to find differences between your file and my checked answer to determine correctness, so check your answers using a reverse complement generator on the Internet.

3. **Write a ½ page summary of how computer science is applicable to genetics**

Rubric: String and Array Manipulation Assignment

	Level 1	Level 2	Level 3	Level 4
Knowledge (10)	No write-up provided	Write-up provided but does no meaningful link – clear cut and pasting evident	Write-up provides a good link between CS and genetics, students own voice evident	Write-up is excellent. Student demonstrates superior knowledge of CS and genetics
Inquiry (5)	Very needy – totally lost	Asks many questions of classmates and teacher.	Student asks a few difficult questions – but is mostly self-sufficient.	Student is self-sufficient and a self-starter. Needs no encouragement.
Communication (5)	No comments, variables are poorly named, camelling for longer variable names is not used, indenting not used, inefficient code, not modular	Some good programming practices are demonstrated.	Most good programming practices are demonstrated, but not flawless.	Student demonstrates a complete grasp of good programming practices. Code is efficient and quick.
Application (10)	Finished product does not meet requirements	Some requirements met – some not.	All requirements met at a good level.	All requirements met at a high level.

Expectations

A1.1 use constants and variables, including integers, floating points, strings, and Boolean values, correctly in computer programs;
A1.3 use assignment statements correctly with both arithmetic and string expressions in computer programs;
A1.6 write programs that declare, initialize, modify, and access one-dimensional arrays.
A2.2 use sequence, selection, and repetition control structures to create programming solutions;
A2.3 write algorithms with nested structures (e.g., to count elements in an array, calculate a total, find highest or lowest value, or perform a linear search).
A4.1 demonstrate the ability to identify and correct syntax, logic, and run-time errors in computer programs;
A4.2 use workplace and professional conventions (e.g., naming, indenting, commenting) correctly to write programs and internal documentation;
A4.3 demonstrate the ability to interpret error messages displayed by programming tools (e.g., compiler, debugging tool), at different times during the software development process (e.g., writing, compilation, testing);
B1.2 demonstrate the ability to solve problems independently and as part of a team;
B3.1 design simple algorithms (e.g., add data to a sorted array, delete a datum from the middle of an array) according to specifications;
C3.2 work independently, using support documentation (e.g., IDE Help, tutorials, websites, user manuals), to design and write functioning computer programs;
d2.2 demonstrate an understanding of an area of collaborative research between computer science and another field (e.g., bioinformatics, geology, economics, linguistics, health informatics, climatology, sociology, art);