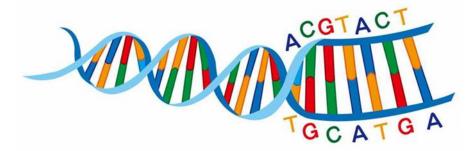
Assignment Set 3

File name: <serial number>_<name>_week<week number>_asg<assignment number>.c

 DNA Sequences. A DNA strand can be represented as a large string built with the four character symbols, A, T, G, and C. Each symbol represents an amino acid. The DNA has a double helix structure that consists of two such strands -- one is the compliment of the other. The symbols A and T are compliments of each other (these amino acids bind) -- similarly G and C are compliments of each other. A DNA strand is read in a particular direction. In the DNA the two complimentary strands are read in the opposite directions. For example, the *compliment* of the DNA subsequence ACGTACT is AGTACGT.



A DNA subsequence is *self-complimentary* if it is identical to its compliment.

If we locate the compliment AGTACGT in a given DNA strand, then we know that the sequence ACGTACT is present in the complimentary strand. When we search for a subsequence in a DNA, we can search only one of the strands, and use the compliment to determine whether the subsequence appears in the complimentary strand.

Write a C program that does the following:

- (a) It reads a DNA subsequence, Z, of length at most 10 from the user, and stores it in an array, seq[], as a string. It then finds the compliment of Z and stores it in another array, comp[], as a string. It then prints the compliment.
- (b) It determines whether Z is self-complimentary.
- (c) It reads another DNA subsequence, Q, of length at most 100. It finds the number of occurrences of Z in Q and the number of occurrences of Z in the compliment of Q. There may be overlapping occurrences, for example there are two occurrences of ATAT in GCATATATC. You may use string library functions such as strncmp() if you wish. For using string library functions, you need to include string.h

Week-4

[Note that Z belongs to the compliment of Q iff the compliment of Z belongs to Q, so you need not find the compliment of Q]

2. **Inverse permutation.** An inverse permutation is a permutation in which each number and the index of the place which it occupies are exchanged. For example, the following are inverse permutations:

a[] = { 2, 7, 4, 9, 8, 3, 5, 0, 6, 1 } b[] = { 7, 9, 0, 5, 2, 6, 8, 1, 4, 3 }

In general, if the permutation is in an array a[], then its inverse is the array b[] such that a[b[i]] = b[a[i]] = i.

Write a C program that reads in a permutation of the integers 0 to n-1 from the user and prints the inverse permutation. Be sure to check that the input is a valid permutation.

3. Decimal expansion of rational numbers. Given two integers p and q, the decimal expansion of p/q has an infinitely repeating cycle. For example, 1/33 = 0.03030303.... We use the notation 0.(03) to denote that 03 repeats indefinitely. As another example, 8639/70000 = 0.1234(142857). Write a C program that reads in two integers p and q and prints the decimal expansion of p/q using the above notation.

[Hint: Sequence of steps for 3/13:

	Quotient	Remainder
Step-1	3/13 = 0	3%13 = 3
Step-2	30/13 = 2	30%13 = 4
Step-3	40/13 = 3	40%13 = 1
Step-4	10/13 = 0	10%13 = 10
Step-5	100/13 = 7	100%13 = 9
Step-6	90/13 = 6	90/13 = 12
Step-7	120/13 = 9	120%13 = 3
Step-8	30/13 = 2	30%13 = 4

Step-8 is the same as Step-2. Hence the decimal expansion is 0.(230769).]