All DNA is made up of sequences of four nucleotides, Adenine, Cytosine, Guanine, Thymine. Each nucleotide is represented by the initial letter of its name (A,C,G,T). A *motif* is a short sequence of DNA that appears to have some particular biological function.

Given a DNA strand, we are often interested in processing its sequence to answer certain questions, such as how many times a particular motif occurs in the DNA strand. In this program, you will process a sequence of DNA, performing several tasks, detailed below.

- Your program will read in a sequence of DNA from a file into a single C++ string. The input file will have lines of DNA, each line containing 60 characters followed by a newline. You will read in one line at a time and concatenate that line with the string.
- Your program will then call a function to verify that the string represents a sequence of DNA. A valid string contains the characters A,C,G, T, and N (where N represents an unknown nucleotide). Your function will receive a string as a parameter and will return 1 if the string represents a DNA strand, and will return 0 otherwise. The function should not change the parameter.
- The third task is to read in a motif from the keyboard, which is a short sequence of DNA. You will write a function that will find all occurrences of the motif in the entire input sequence. For example, if the DNA sequence is ACGTTTACGTTACGTC and the motif is ACG, your program should report that the motif is found 3 times in the DNA sequence, at positions 0, 6, and 11 of the DNA sequence. You will write a function called **print_occurs** that will receive two parameters. The first parameter is a string representing a longer DNA sequence, the second parameter is a string containing a shorter motif. The function should print the positions of each occurrence of the motif in the longer sequence. If the motif is not found at all, the function should print an appropriate message. The function should not change either parameter.

Your Main Program

Your program should use files for both input and output.

1. The program should use a loop that continues until there is no more input in the input file. Read in a string representing a DNA sequence. Since there will not be any spaces within each line of the input file, you can use **cin >>.** However, you may also use **getline**. Each method has a different way of removing the newline character at the end of each line.
2. Print "Original string for DNA sequence:" followed by the original string.
3. Read in a character string representing a DNA motif.
4. Print "Original string for DNA motif:" followed by the string just read in.
5. Call **isvaliddna** to verify that the DNA sequence represents a valid strand of DNA. Call **isvaliddna** again, to verify that the motif represents a valid strand of DNA. If either one is invalid, skip step 6.
6. Call **print_occurs** to print out the positions of each occurrence of the motif in the DNA sequence.

Testing your program

I will give you some test data for your program, but you must also run it on your own test cases. Make up at least 5 sets of input. Make sure that some contain invalid DNA, some do not contain any occurrences of the motif, etc.