1. **(Palindromes)** A palindrome is a string that’s spelled the same way forward and backward. Some examples of palindromes are: “radar,” “able was I ere I saw elba,” and, if you ignore blanks, “a man a plan a canal panama.” Write a recursive function testPalindrome that returns 1 if the string stored in the array is a palindrome and 0 otherwise. The function should ignore spaces and punctuation in the string. Write a test program that prompts the user to enter a string and display whether the string is a palindrome.
2. **(Print a String Backward)** Write a recursive function stringReverse that takes a character array as an argument, prints it back to front and returns nothing. The function should stop processing and return when the terminating null character of the string is encountered.
3. **(Tokenizing Telephone Number)** Write a program that inputs a telephone number as a string in the form (02)2733-3141. The program should use function strtok to extract the area code as a token, the first four digits of the phone number as a token and the last four digits of the phone number as a token. The eight digits of the phone number should be concatenated into one string. The program should convert the area code string to int and convert the phonenumber string to long. Both the area code and the phone number should be printed. (Put a 0 in front of the conversion specifier will make the program to print the leading zeros in front of a number. For example: printf(“%08d”, num);)
4. **(Counting the Number of Words in a String)** Write a program that inputs 4 lines of text and uses strtok to count the total number of words. Assume that the words are separated by either spaces or newline characters.
5. **(Displaying a Sentence with Its Words Reversed)** Write a program that inputs a line of text, tokenizes the line with function strtok and outputs the tokens in reverse order.
6. **(Alphabetizing a List of Strings)** Use the string-comparison functions and the techniques for sorting arrays to write a program that alphabetizes a list of 6 strings.

Here is a sample run:

Input strings:

Ryan Brown DuMond Larson Thorpe Elizabeth”

The result in alphabetizing order:

“Brown”, “DuMond”, “Elizabeth”, “Larson”, “Ryan”, “Thorpe”

1. **(Searching for Substrings)** Write a program that inputs a line of text and a search string from the keyboard. Using function strstr, locate the first occurrence of the search string in the line of text, and assign the location to variable searchPtr of type char \*. If the search string is found, print the remainder of the line of text beginning with the search string. Then, use strstr again to locate the next occurrence of the search string in the line of text. If a second occurrence is found, print the remainder of the line of text beginning with the second occurrence. [Hint: The second call to strstr should contain searchPtr + 1 as its first argument.]
2. **(Random Sentences)** Write a program that uses random number generation to create sentences. The program should use four arrays of pointers to char called article, noun, verb and preposition. The program should create a sentence by selecting a word at random from each array in the following order: article, noun, verb, preposition, article and noun. As each word is picked, it should be concatenated to the previous words in an array large enough to hold the entire sentence. The words should be separated by spaces. When the final sentence is output, it should start with a capital letter and end with a period. The program should generate 20 such sentences. The arrays should be filled as follows: The article array should contain the articles "the", "a", "one", "some" and "any"; the noun array should contain the nouns "boy", "girl", "dog", "town" and "car"; the verb array should contain the verbs "drove", "jumped", "ran", "walked" and "skipped"; the preposition array should contain the prepositions "to", "from", "over", "under" and "on".