C Programming Assignment List

Strings

1. Write a program to convert lower case string to upper case string and vice versa.

2. Write a program to reverse a string using recursive functions

3. Write a program to read n number of strings using two-dimensional character array, sort them and display the sorted list of strings on the screen.

4. Write a program to read n number of strings and display them on the screen. Use array of pointers and dynamic memory allocation techniques.

5. Write a C program with a function any (s1, s2). This function returns the first location (index of location) in the string s1 which matches with any string in s2 otherwise.

6. Write a C program with a function delete (s1, c). This function deletes each character in s1 which matches character c.

7. Write a Program to implement ***strtok*** library function.

8. Write a C program with a function deletes2 (s1, s2). This function deletes each character in string s1 which matches any character in string s2.

9. Write a function expand (s, t) which converts characters like newline and tab into visible escape sequences like \n and \t as it copies the string s to t. Use switch statement and also display both s and t at the end.

10. Write a function expand (s1, s2) which expands shorthand notations of s1 like a-d into abcd and 0-9 to 0123456789 in s2. For example if the string in s1 is 0123a-e1-4 then s1 is expanded in s2to 0123abcde1234.

11. Write a program to print out all rotations of a string typed in. For eg:if the input is “Space”, the output should be: space paces acesp cespa espac.

12. Implement string library functions. strrev, strcpy, strcat, strcmp with same return values and all error handling features using pointers.

Structures, Unions and Enumeration

1. Write a program to represent time of the day in hrs, mins and secs. Use structures.

2. Define structure with two members (one int and other char). Also define s union with two members (one int and other char). Print the sizes of structure and union in number of bytes.
3. Define a structure declaration for each of the following situations. Assume a 16-bit integer word
   a) Define three bit fields, called a, b and c, whose widths are 6-bits, 4-bits and 6-bits, respectively
   b) Declare a structure-type variable v having the composition defined in part (a) above. Assign

initial values 3, 5 and 7 respectively, to the three bit fields. Are the bit fields large enough to accommodate these values?

c) What are the largest values that can be assigned to each of the bit fields defined in part (a) above?

d) Define three bit fields, called a, b and c, whose widths are 8 bits, 6 bits and 5 bits, respectively. How will these fields be stored within the computer's memory?

e) Define three bit fields, called a, b and c, whose widths are 8 bits, 6 bits and 5-bits respectively.

Separate a and b with 2 vacant bits.

4. Develop a program to generate marks sheet of C-DAC, Hyderabad Students (DSSD, DESD and DAC courses). Modules are different for each course. Implement this using structures, unions, arrays, loops and variables.
5. Write a program to search for a given element in a list of elements using Linear Search. Use flag to represent the status of search. Define flag as an enumeration variable whose value is either true or false.
6. Write a menu driven C program to perform operations on Complex numbers. Use enumeration data type to identify the different operations on Complex numbers.

Files, Console I/O and Command line arguments

1. Experiment to find out what happens when printf argument string contains \x, where x is some character (a, b, c, \, ^ etc). What are your observations.

2. Write a program to remove all the comments from a 'C' program.

3. Write a program that will concatenate two files, that is append the contents of one file at the end of another file and write the results into a third file. You must be able to execute command at DOS prompt as follows:
   C > CONCAT Source 1.txt source 2.txt Target.txt.

4. Write a c program to printing the same file on the console.

5. Write a C program to open a file and store text (character type data) in one's complement form. Read the contents from the file and display as it is as well in one's complement form. Use command line arguments to pass file name to your C program.

6. Write a program which reads a line of characters. Each character entered from the keyboard is tested to determine its case, and is then written to the data file in opposite case. Display the contents of the file. Also use ftell and fseek to determine the current file position and to change the file position.
7. Write a program to embed assembly language code in C program

8. Write a program to read a positive integer at least equal to 3, and print out all possible permutations of three positive integers less or equal to than this value.

9. Given as input a floating (real) number of centimeters, print out the equivalent number of feet (integer) and inches (floating, 1 decimal), with the inches given to an accuracy of one decimal place.

   Assume 2.54 centimeters per inch, and 12 inches per foot. If

   the input value is 333.3, the output format should be:

   333.3 centimeters is 10 feet 11.2 inches.

10. Given as input an integer number of seconds, print as output the equivalent time in hours, minutes and seconds. Recommended output format is something like 7322 seconds is equivalent to 2 hours 2 minutes 2 seconds.

11. Read a positive integer value, and compute the following sequence: If the number is even, halve it; if it's odd, multiply by 3 and add 1. Repeat this process until the value is 1, printing out each value. Finally print out how many of these operations you performed.
    Typical output might be:

    Inital value is 9
    Next value is 28
    Next value is 14
    Next value is 7
    Next value is 22
    Next value is 11
    Next value is 34
    Next value is 17
    Next value is 52
    Next value is 26
    Next value is 13
    Next value is 40
    Next value is 20
    Next value is 10
    Next value is 5
    Next value is 16
    Next value is 8
    Next value is 4
    Next value is 2
    Final value 1, number of steps 19 If the input value is less than 1, print a message containing the word

    Error

    and perform an exit(0);

12. Write a program to count the vowels and letters in free text given as standard input. Read text a character at a time until you encounter end-of-data. Then print out the number of occurrences of each of the vowels a, e, i, o and u in the text, the total number of letters, and each of the vowels as an integer percentage of the letter total.

    Suggested output format is:

    Numbers of characters:
    a 3 ; e 2 ; i 0 ; o 1 ; u 0 ; rest 17
    Percentages of total:
    a 13%; e 8%; i 0%; o 4%; u 0%; rest 73%

    Read characters to end of data using a construct such as

    ```
    char ch;
    while(( ch = getchar() ) >= ) {
     /* ch is the next character */   ....     }
    ```

    to read characters one at a time using getchar() until a negative value is returned.

13. Write a program to read English text to end-of-data (type control-D to indicate end of data at a terminal, see below for detecting it), and print a count of word lengths, i.e. the total number of words of length 1 which occurred, the number of length 2, and so on.

    Define a word to be a sequence of alphabetic characters. You should allow for word lengths up to 25 letters. Typical output should be like this:

    length 1 : 10 occurrences

    length 2 : 19 occurrences

    length 3 : 127 occurrences
    length 4 : 0 occurrences
    length 5 : 18 occurrences
    ....

14. Write a program last that prints the last n lines of its text input. By default n should be 5, but your program should allow an optional argument so that

        last -n

    prints out the last n lines, where n is any integer. Your program should make the best use of available storage. (Input of text could be by reading a file specified from the command or reading a file from standard input)

15. Write a program to list the files given as arguments, stopping every 20 lines until a key is hit.(a simple version of more UNIX utility)