Assignment X

Submit all the programs separately against each assignment in the Moodle System. Provide the result in a separate output file (named, result_<assgn><no>.txt). Use standard output redirection feature to generate the output file.

Hints. If you run the program with the following command

./a.out >result.txt

Output of your program (generated by printf(.) function) will be written in the file result.txt. You need to provide input from your keyboard, by remembering the sequence of inputs to be given or writing them in a text file in the same sequence.

Otherwise you may use the redirection for the standard input file, such as,

./a.out <input.txt

For the above all your printing by printf(.) function would be displayed on your monitor.

For both reading from a file and writing to a file use the following.

./a.out <input.txt >result.txt

If you execute the program multiple times, you may concatenate the outputs in a single file by using the following redirection command:

./a.out >>result.txt

or

./a.out <input.txt >> result.txt

(a) Implement the following fucntions:

(a) Allocate_2D_Array: for dynamically allocating a 2D array of real numbers of size MxN.
(b) Read_2D_Array: for reading an array of real numbers of size MxN.
(c) Print_2D_Array: for printing an array of real numbers of size MxN.
(d) Free_2D_Array: for freeing memory space of a dynamically allocated array.
(e) Multiply_matrices: for multiplying two matrices of size MxN and NxL respectively. The result is stored in a dynamically allocated array of appropriate size and the address of the first element is returned. If the multiplication is not possible, it should return NULL.
(f) Compute_trace: for computing trace (sum of diagonal elements) of a matrix of size MxN.

Write a program which reads two matrices of sizes MxN and NxL, respectively, and perform multiplication. Values of all the matrices (input matrices and the resulting matrix) are printed one after another. Finally traces of all the matrices are printed.

Run your program for the following matrices:

(a) Matrix 1:

1 2 3 4
5 6 7 8
9 10 11 12

Matrix 2:

13 14 15 16 17
18 19 20 21 22
23 24 25 26 27
28 29 30 31 32

(b) Matrix 1:

1 2 3 4
5 6 7 8
9 10 11 12

Matrix 2:

13 14 15
18 19 20
23 24 25
28 9 30

(c) Matrix 1

27 29 30 31 32
1 2 3 4 5
6 7 8 9 10

Matrix 2:

1 0 0
0 1 0
0 0 1
-1 1 1
1 -2 1

(b) Define a structure for nodes of a linked list of characters (called stack of characters), and write a program as follows:

(a) Write a function push(.), which takes a character and the address of the head of the list as arguments, and add the node at the beginning of it.
(b) Write a function pop(.), which removes a node from its head and returns the address of the removed node. If the list is empty, it returns NULL.
(c) Write a main program which reads a line containing an algebraic expression, which uses operators such as *, +, -, /, (, and). For example an expression could be given in the following form:

e*f+(a+(b*c*(d+e)/(e+f)))+d*g

The program checks whether all the parentheses in the expressions are matched, using the linked data structure as described above. It also prints the segments of within the enclosure of each matched parentheses pair. For example, the output of the above expression would be:

(d+e)
(e+f)
(b*c*(d+e)/(e+f))
(a+(b*c*(d+e)/(e+f)))
Matched

(d) Test your program with five expressions and generate the output for them writing in an output file. You should write the input expressions also in the file. You may use *stdout* redirection for generating output. Take some positive and negative examples to demonstrate the correctness of your program.

Hints:

(i) Read each character from the input string and push them in the list till you find ')'.
(ii) Pop all the characters from the list and store it in a string S, till you get '('. If you do not get any '(' and the list becomes empty, the expression is not matched. Quit from the computation. Otherwise, you write the string S in the output. Once again push the same string (S) character by character by replacing every '(' by '[', and ')' by ']' in the list. While writing the string S in the output you should convert corresponding braces, '[' and ']', to '(' and ')', respectively.
(iii) Repeat the above two steps till you come at the end of the input string, and write "Matched" in the output.

Please note that:

(i) Your program may be run on a separate data set.

(ii) If dynamic memory allocation is not used for the second problem and fixed array allocation is used maximum 80% marks will be awarded for otherwise successful completion of the assignment.