



Engineering Code Test

Last Revised: November 26, 2019

Copyright © 2019, DocuPhase Corporation. All rights reserved.

IMPORTANT NOTICE

DocuPhase has made every effort to ensure that this document is accurate;
DocuPhase disclaims liability for any inaccuracies or omissions.

No part of the contents of this publication may be reproduced, transmitted, transcribed,
stored in a retrieval system, or translated into any language (in any form)
without written consent from DocuPhase Corporation.

Information in this document is subject to change without notice and
does not represent commitment on the part of DocuPhase.

If you find information in this document that is incorrect, misleading, or incomplete,
we would appreciate your comments and suggestions.

Instructions

Expectations

- ✓ You have 7 days, from the receipt of this test, to complete and return your response.
- ✓ Using the C# language to answer any two (2) of problems 1 through 5, then also for problem 6: for a total of 3 answers.
- ✓ Please provide source code for each project in its own folder: named for the respective question that is being answered (q1, q2, q3, etc.). Then, compress those folders into a single zipped file, and return that to DocuPhase.
- ✓ Each program should accept a file as input: the full path of which should be a parameter passed into your program.
- ✓ The output should be sent to standard out.
- ✓ Each problem below includes sample input and sample output. Please conform your input and output to the samples provided. Also note that the "****" characters are for delineation only, and should not be included as input or as output.



EXAMPLE

C:\MyProjects\StringManip.exe C:\Temp\Strings.txt



IMPORTANT!

Elegant design and clean coding style are considered to be equally important as a correct solution. This solution should be considered production code: meaning that whatever quality control, unit testing, comments, etc., you would put into production code, should be included in your Code Test solutions in this test.

Problems

#1 – String Manipulation

Write a function to compact a string in place: A. strip whitespace from the string. B. remove duplicate characters if they are next to each other.



EXAMPLE

If the input is:

*** abb cddpdef gh ***

Then, the output should be:

*** abcdpdefgh ***

#2 – Spiral printing

Write a function to print a 2-D array (n x m) in spiral order (clockwise).



EXAMPLE

If the input is:

```
***
1  2  3
4  5  6
7  8  9
***
```

Then, the output should be:

```
***
1 2 3 6 9 8 7 4 5
***
```

#3 - Sub-Trees

- ✓ Write a function to check the following:
 - If binary tree 1 is a subtree of another binary tree 2
 - Tree 1 is a subtree of Tree 2, if (and only if) we can find a subtree in Tree 2 with the same structure as Tree 1; in addition, each node must have the same value.
- ✓ Input for this problem will use a breadth-first heap style representation.



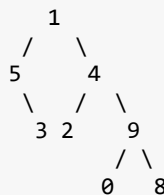
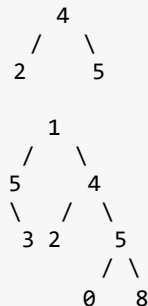
EXAMPLE A

4,2,5

Which is a subtree of:
1,5,4,,3,2,5,,,,,0,8

But it's not a subtree of:
1,5,4,,3,2,9,,,,,0,8

Represents this tree:



EXAMPLE B

If the input is:

```
***
1,5,4,,3,2,5,,,,,0,8
4,2,5
***
```

Then, the output should be:

```
***
Yes
***
```



TIP

To see how to create a binary tree, visit: <http://cslibrary.stanford.edu/110/BinaryTrees.html>

Sample struct and function in C:

```
typedef struct {
    int val;
    Node *left;
    Node *right;
}Node;

Node *subtree(Node *t1, Node *t2);
```

#4 – Min/Max Stack

Write your own implementation of the stack data structure that allows a user to determine the minimum and maximum value in the stack.

- ✓ Include the following methods: Pop(), Push(item), Peek(), IsEmpty(), Min(), Max()
- ✓ You may assume that all data in the stack will be numeric
- ✓ For input, the file may contain each data item on a separate line

#5 – Ordered Number Sequence

Write a function that takes an array of numbers as input, and outputs an array of numbers where

- ✓ There are no duplicates, and
- ✓ The original order of the numbers is preserved



EXAMPLE

If the input is:

[1, 2, 1, 1, 5, 6, 4, 4, 3, 2]

Then, the output should be:

[1, 2, 1, 5, 6, 4, 3, 2]

#6 – Data Processing

Write an application that takes a text file with the following format as input:

TableName Column 1 Name, Column 2 Name, Column 3 Name, ... Column N Name Item1, Item2, Item3, Item5, Item4, some other data, etc.

- ✓ Where the first line of the file has the name of a database table, and
- ✓ The second line has the columns of the table, and
- ✓ Lines 3 through N have the data lines for that table
- ✓ All data is comma separated, and
- ✓ Results in the following
 - Once the file is parsed, it should create a database table with the proper columns and data types, as well as insert the data from the file into those columns.
 - Validation or processing errors must not cause the program to crash: you should be able to log an entry into a log file, and continue processing the file.



NOTE

The connection to the database may be stored in a configuration file, and there may or not be any tables in the database before the application is executed.