Ain Shams University
Faculty of Engineering
Computer and Systems Engineering Department

CSE 323: Programming with Data Structures– 3rd Year CSE
2nd Semester 2020/2021

Project - XML editor

Submitted to:

Eng. Amr Mohamed

Name	Section	Code
Aya Sameh Mazhar	1	1700342
Bassant Yasser	1	1700360
Sarah Mohamed Ahmed Ahmed	2	1700593

Video link: https://drive.google.com/file/d/1IBQnIr1dTEl8i-VPciHpbYS3FZwo-O-X/view?usp=sharing

Repo link: https://github.com/Bassant-Yasser/XML-Editor

Contents

Background:

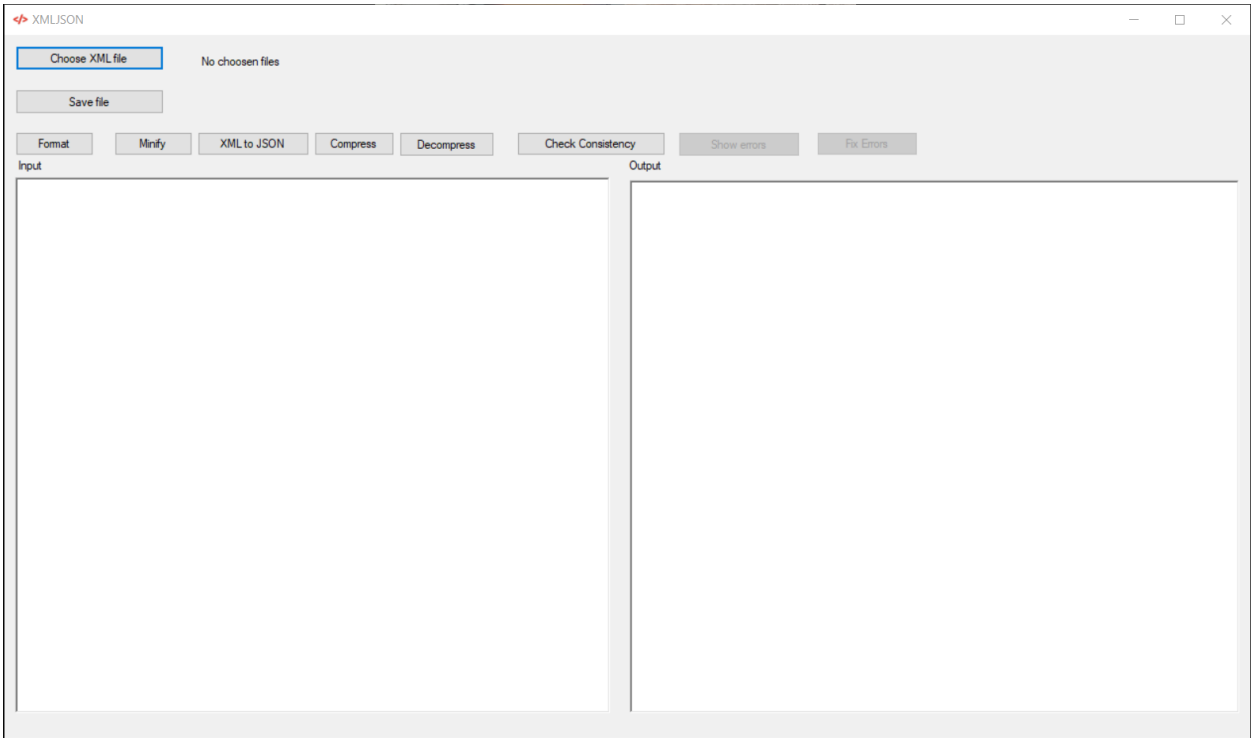
We are using different types of data structures here in the project:

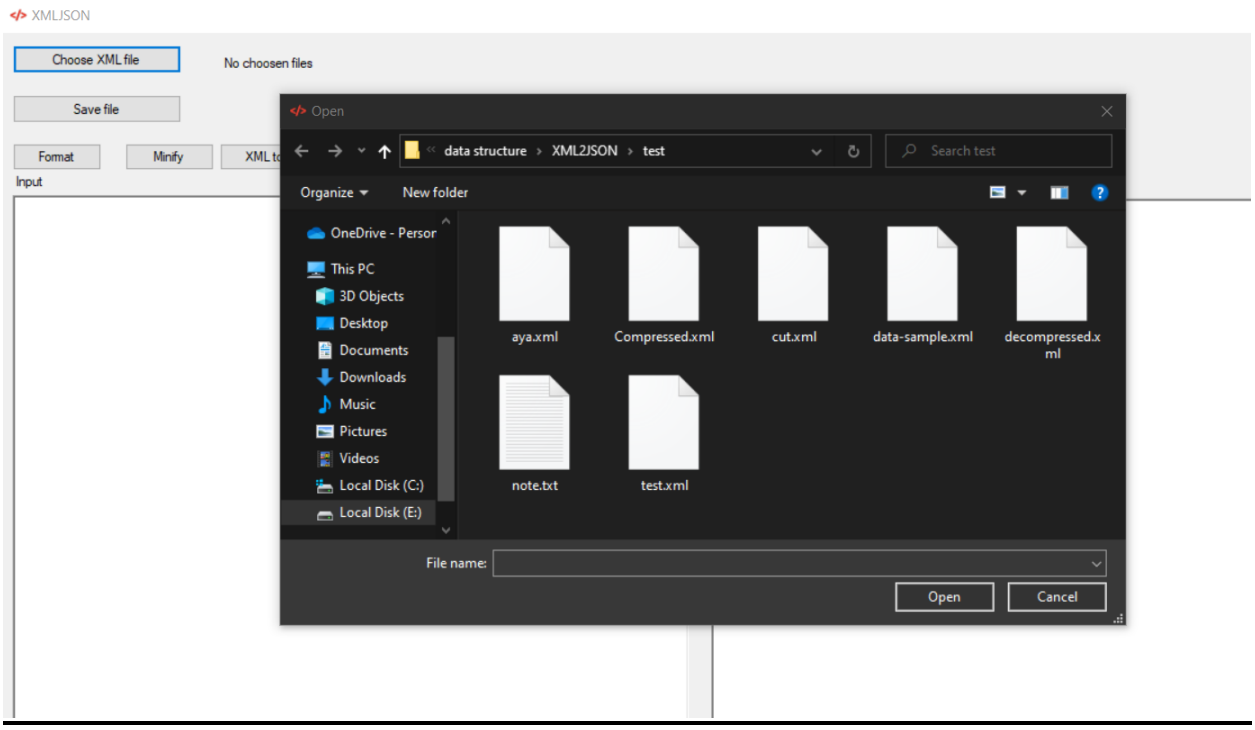
- Tree: used in the insertion the tags name, attributes if existed, and data which later will help in the implementation of the format, minify, and convert to json functions.
- Stack: we are mainly using it to check the consistency of the xml file, and it helps us in fixing the errors.
- Dictionary or map: used in the compress function.

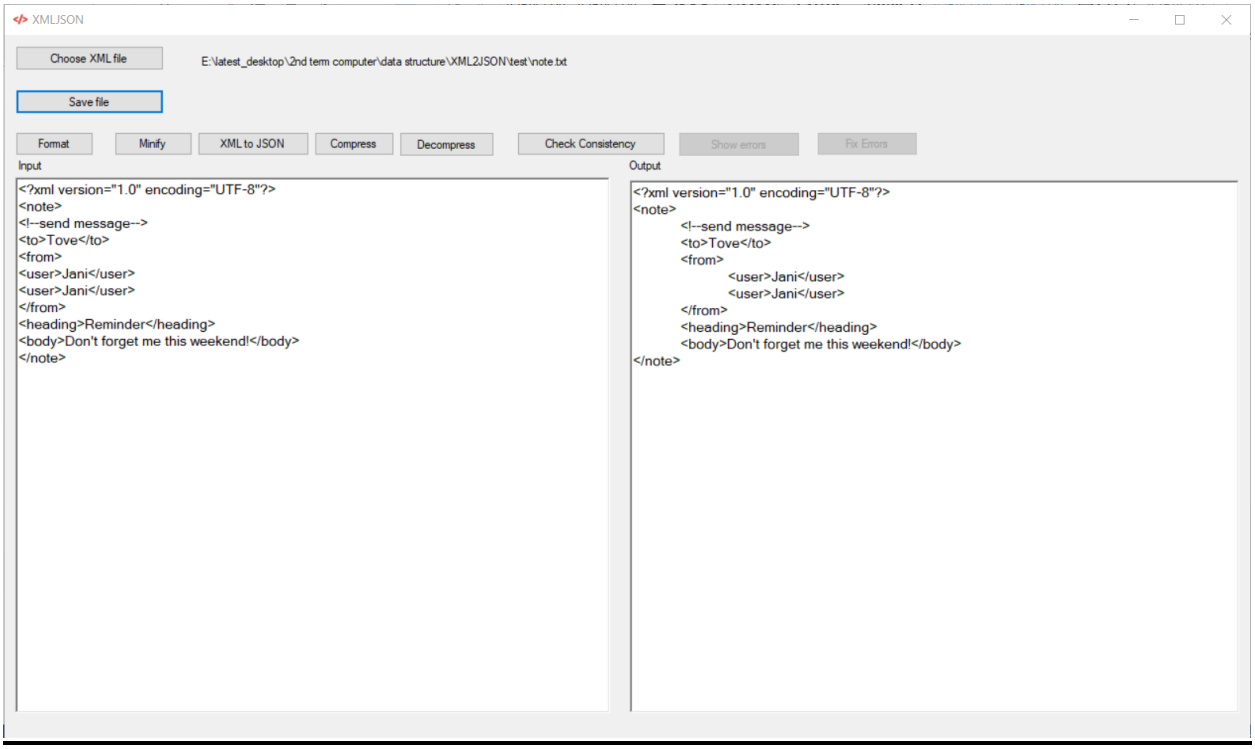
Implementation Details:

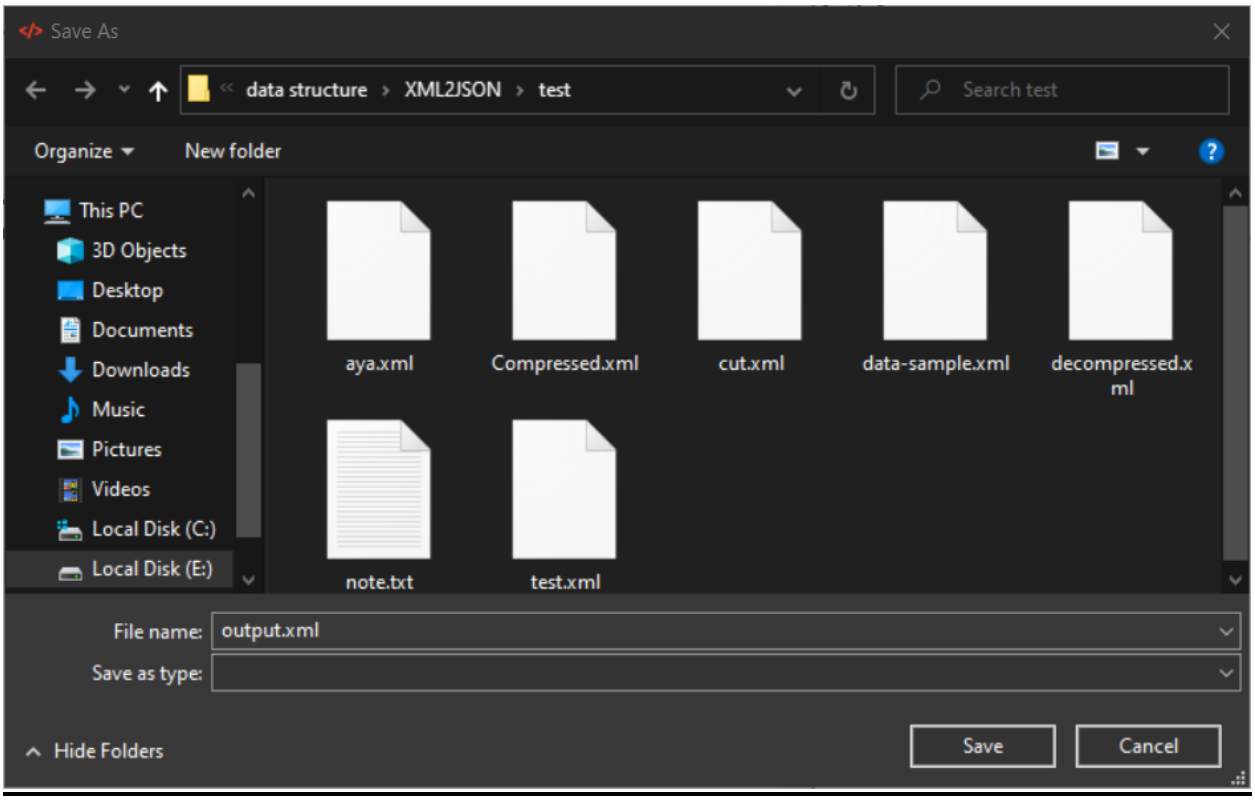
First thing you should do to use our program without facing any errors: you should choose the xml or txt file using choose XML button, and then press the Check Consistency button.

After printing any output, you can save the file on your computer at any desired location on your computer and you are free to choose its' name and extension.




XMLJSON
Choose XML file
No choosen files
Save file
Format
Minify
Input
Open
data structure › XML2JSON › test
Search test
Organize
New folder
OneDrive - Persor
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (E:)
aya.xml
Compressed.xml
cut.xml
data-sample.xml
decompressed.xml
note.txt
test.xml
File name:
Open
Cancel


XMLJSON
Choose XML file
E:\latest_desktop\2nd term computer\data structure\XML2JSON\test\note.txt
Save file
Format
Minify
XML to JSON
Compress
Decompress
Check Consistency
Show errors
Fix Errors
Input
<?xml version="1.0" encoding="UTF-8"?>
<note>
<!--send message-->
<to>Tove</to>
<from>
<user>Jani</user>
<user>Jani</user>
</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
Output
<?xml version="1.0" encoding="UTF-8"?>
<note>
<!--send message-->
<to>Tove</to>
<from>
<user>Jani</user>
<user>Jani</user>
</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>



1.Minify:

- The input to this function is a XML file which everything is in a separate line without any spaces or tabs before or after the data and the output is a string of concatenated strings.

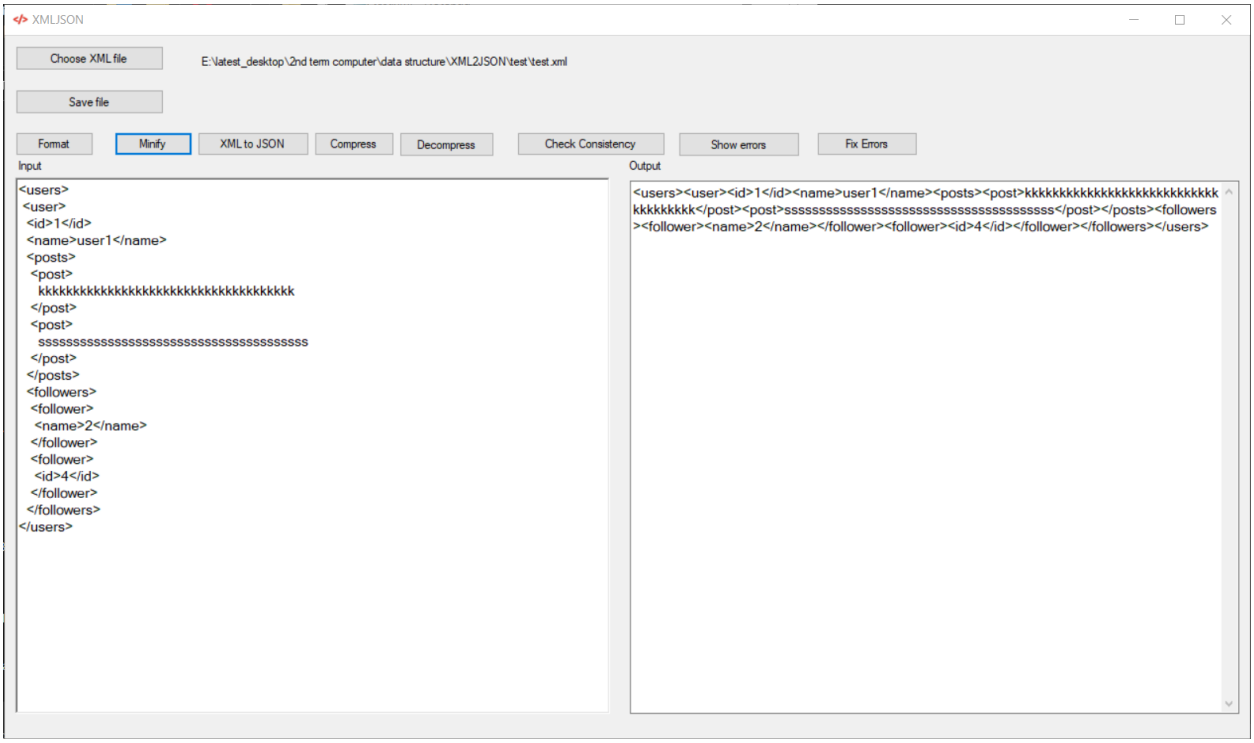
cut.xml - Notepad

File Edit Format View Help

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<!--send message-->
<to>
Tove
</to>
<from>
<user>
Jani
</user>
<user>
Jani
</user>
</from>
<heading>
Reminder
</heading>
<body>
Don't forget me this weekend!
</body>
</note>
```

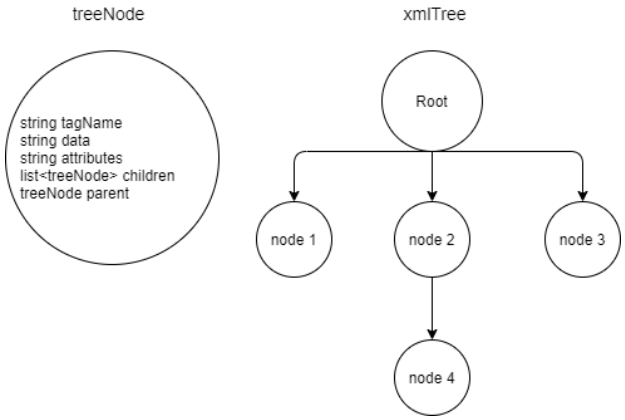
Input XML file example

- Test case:



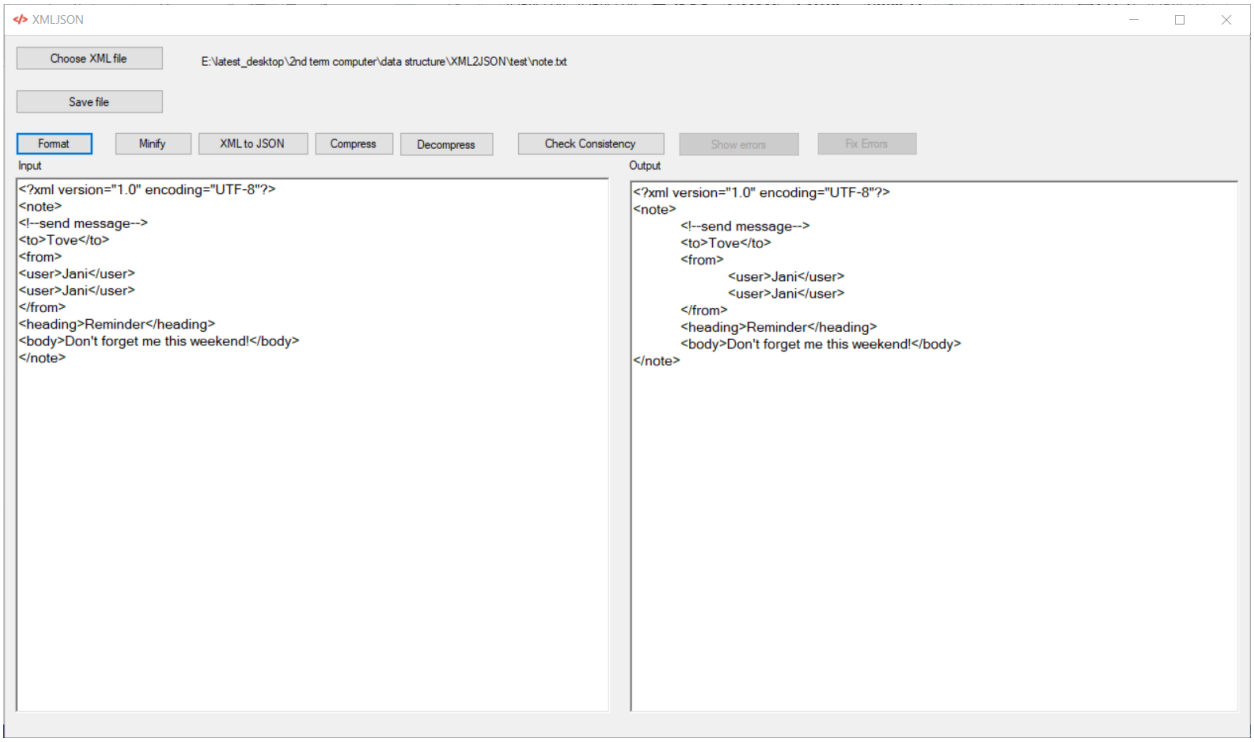
2.Formatting:

- The input to this function is a xmlTree and its root of the chosen XML file which is traversed in pre-order to output a formatted XML file.



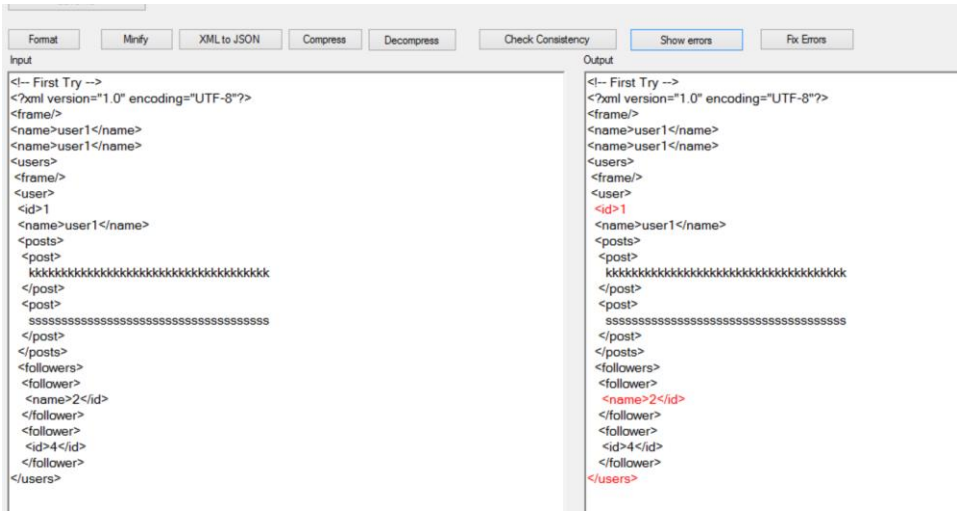
- The tree root is always empty treeNode so, that it can store the comments and self closing tags, when printing each node the tabs number is the height of the node from the root minus one (tabs = height(treeNode) – 1)

- Test case:



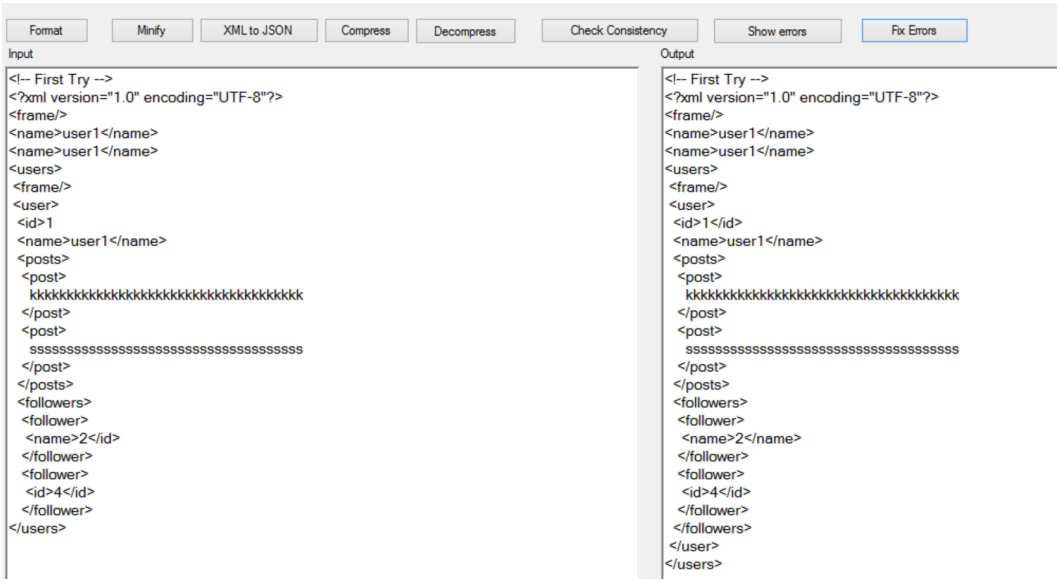
3.Show Errors:

- The Line After an error occurred will be printed in red in the output box.
- We start by counting tabs (or spaces) in the beginning of the line, then we push it on a stack if it is bigger than the top of that stack, or if the stack is empty.
- After that we push the opening tag if existed, and then we search the rest of the line to see if there is a closing tag.
- If there is a closing tag, we pop the stack where we push the spaces, and pop the tags stack, if that closing tag does not match the opening tag then we report an error, and this line will be red colored.
- If not, we move on to the next line.
- We again check the spaces count, now if it is smaller than the top of the spaces stack, and the last tag was not closed (and it is an opening tag line), we report error, and this line will be red colored because of detecting an error.
- The last type of errors we can show and solve is when the number of spaces is equal to the top of the spaces stack, if it is an opening tag, we notice that the last tag was not closed, so again we report an error, and this line will be red colored.
- Other than that, if it is a closing tag line, we check the top of the tags stack, if it matches we pop the stack, if not report an error.



4.Fix Errors:

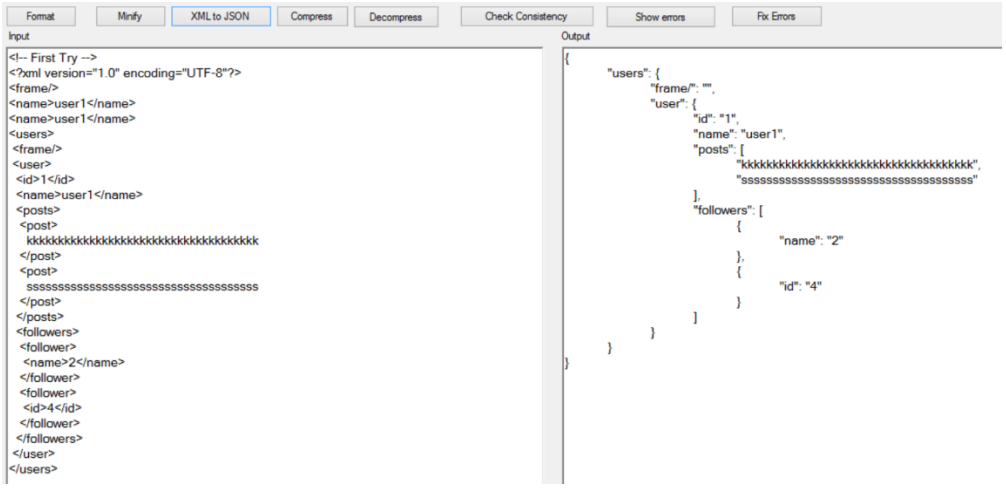
- At the same place where we detected the error, we solve it.
- We add any missing closing tags at its place, and then add it to the box where we write the output.
- We correct any wrong or misspelled closing tag.



5.Convert into Json:

- We use the tree here, so in order to insert correctly in the tree, you should make sure that this is a consistent xml file.
- If it is not, use our feature(Fix Errors), then save the file on your computer, and choose it again to be in the input box where we use to insert in the tree.
- JSON format is started with { and ended with }.

- The function works recursively.
- The base condition of the function is that the tree node has no children, so we print it in the right format.
- If it has one child, so this child will be in [child].
- If it has more than one child, first we check if they are all the same type then they will be in array notation [children], if not so they will be in object notation { children }.
- The program can adjust indentation of the root and all the children according to their level in the tree.
- The program can adjust the location of , after each child if it is not the last child of its parent.



Note: as shown in the picture I entered the consistent xml file as an input.

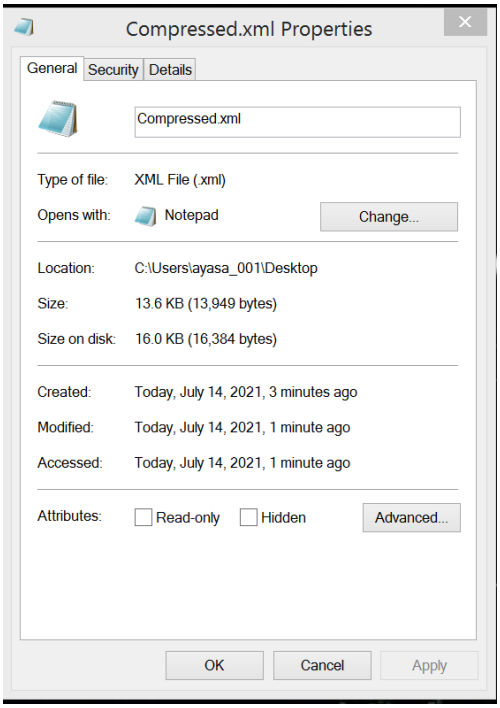
6.Compress:

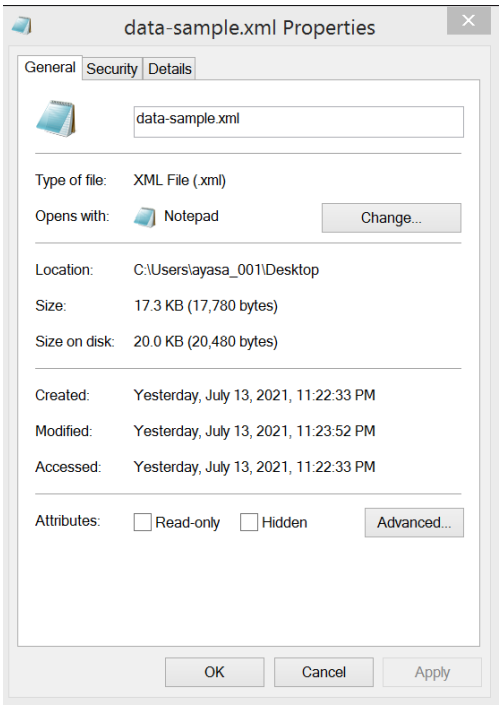
- We are using LZW Data Compression technique.
- After pressing the Compress button, a compressed file will be saved in the same destination of the xml file.
- In the Compress function, I use the minify function to try to decrease the size as much as possible.
- I also remove any spaces, so the compressed file will be a stream of numbers in one line.
- You can decompress it using the decompress button.
- When you press the decompress button, I call the decompress function giving it the string of the compressed file, the output string will be minified, so after that I call the format function to return the file to its first state completely.

```
*    PSEUDOCODE OF COMRESS FUNCTION
1    Initialize table with single character strings
2    OLD = first input code
3    output translation of OLD
4    WHILE not end of input stream
5        NEW = next input code
6        IF NEW is not in the string table
7                S = translation of OLD
```

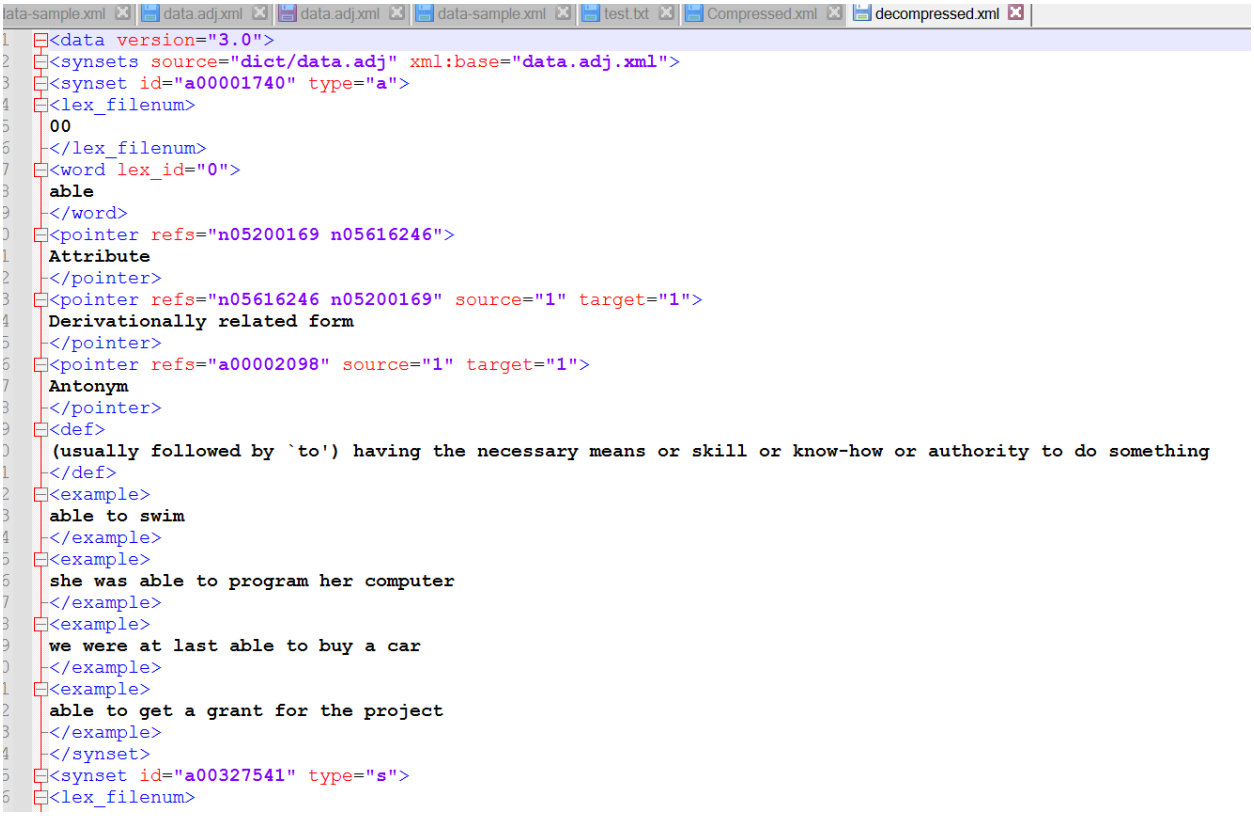
```
8                S = S + C
9        ELSE
10               S = translation of NEW
11        output S
12        C = first character of S
13        OLD + C to the string table
14        OLD = NEW
15    END WHILE
```

Here I used the data-sample file that was given to us.





The decompressed file: which was saved in my computer.



```
<data version="3.0">
<synsets source="dict/data.adj" xml:base="data.adj.xml">
<synset id="a00001740" type="a">
<lex_filenum>
00
</lex_filenum>
<word lex_id="0">
able
</word>
<pointer refs="n05200169 n05616246">
Attribute
</pointer>
<pointer refs="n05616246 n05200169" source="1" target="1">
Derivationally related form
</pointer>
<pointer refs="a00002098" source="1" target="1">
Antonym
</pointer>
<def>
(usually followed by `to') having the necessary means or skill or know-how or authority to do something
</def>
<example>
able to swim
</example>
<example>
she was able to program her computer
</example>
<example>
we were at last able to buy a car
</example>
<example>
able to get a grant for the project
</example>
</synset>
<synset id="a00327541" type="s">
<lex_filenum>
```

Complexity of Operations:

1.Minify:

O(n), where n is the number of lines in xml file.

2.Formatting:

O(n^2), where n is the number of children to the treeNode.

3.Show Errors:

O(n), where n is the number of lines in xml file.

4.Fix Errors:

Same complexity as Show Errors.

5.Convert into Json:

O(n^2),where n is the number of children to the treeNode.

5.Compress:

O(n), where n is the number of characters in the string that contains the xml code.

Reference:

https://www.c-sharpcorner.com/UploadFile/mahesh/how-to-read-a-text-file-in-C-Sharp/

https://www.youtube.com/watch?v=FJ5rNOMTZ9w

https://www.youtube.com/watch?v=ldOeGUto6BMhttps://www.xml.com/pub/a/2006/05/31/converting-between-xml-and-json.html?fbclid=IwAR2NJfnq4s85ilnw5h8Ali7OG4ZDNoiIv81n4JrSv4XJVKbDSOalqoyf1yk

https://www.youtube.com/watch?v=ttibGd34y8Q

https://www.educba.com/json-vs-xml/

https://www.codeproject.com/Articles/37668/Multiple-Colored-Texts-in-RichTextBox-using-C

https://dev.to/niinpatel/converting-xml-to-json-using-recursion-2k4j

https://www.coursera.org/lecture/data-structures-design-patterns/tree-implementation-wxcwp