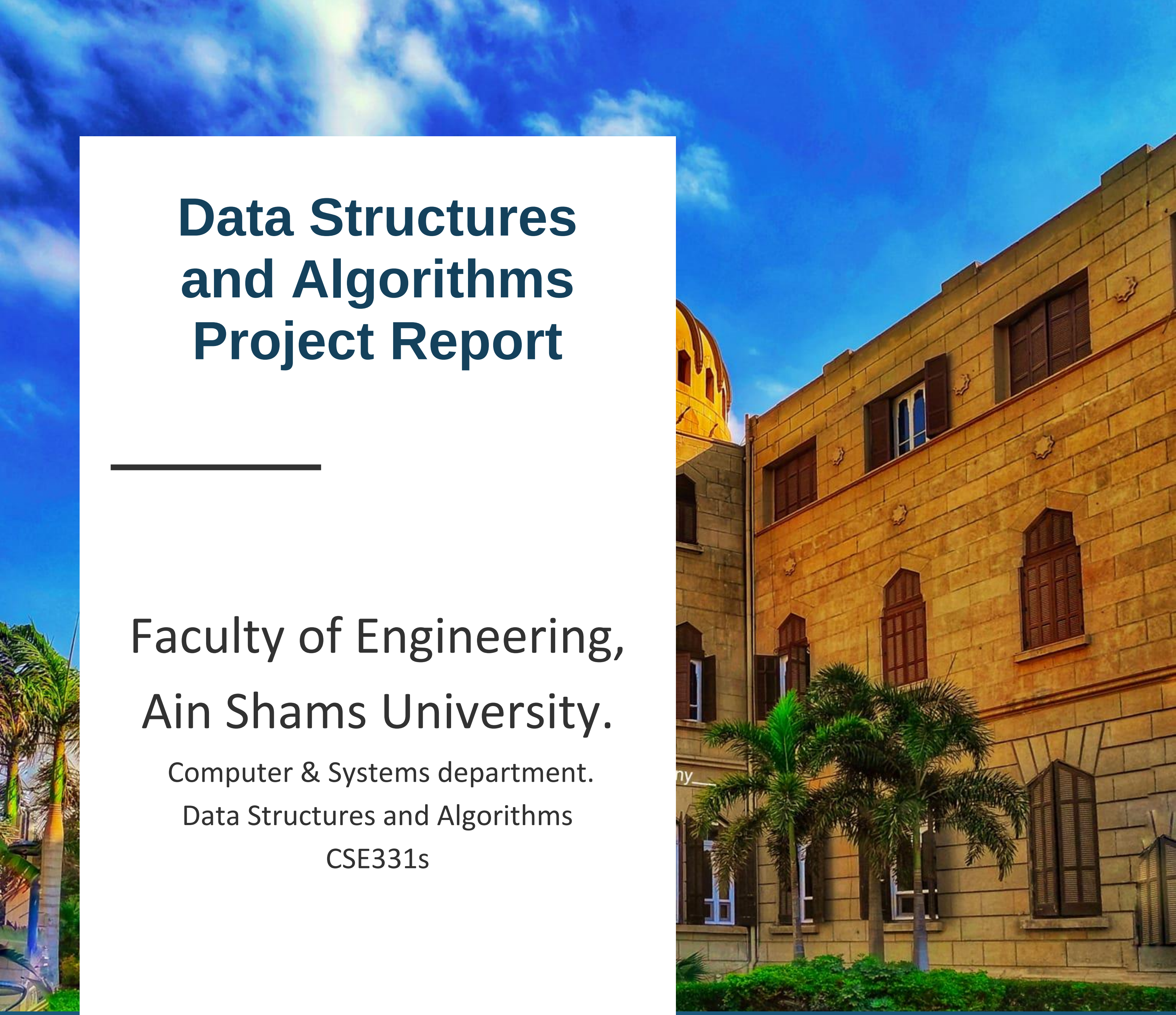


Data Structures and Algorithms Project Report

Faculty of Engineering,

Ain Shams University.

Computer & Systems department.

Data Structures and Algorithms

CSE331s

Under the supervision of.

Dr. Islam Elmadah

Eng. Fady Faragallah

Contents

Who are we?

• Mazen Saaed Farouk	2001080
• Sameh Ossama Nabil	2000379
• Omar Nader Ahmed	2001714
• Hazem Zainhom Abdel-Alim	2000168
• Amgad Sherif abdelrahman	2000121

Code in xmlParser.cpp, graph.cpp & Gui part

GitHub Repository: https://github.com/ElecSpartan/DS_Project
Video Link:
https://drive.google.com/drive/folders/1QX79Ab3qDl9w2bIAgUaaybdsrjgfmTPK

Introduction

We have divided the project into three main sections.

GUI:

1- File location input & text input.

2- Buttons for different operations & Save output to file.

3- Graph representation & graph functionality.

XML:

1- Check consistency (open/close tags and not matching tags).

2- Get location of the error and solve them -> show it in GUI.

4- Formatting the XML.

5- XML to JSON.

6- Minifying XML file.

7- Compressing XML/JSON (tokenization).

8- Decompressing XML/JSON.

Graphs:

1- Represent users as graphs (id, name, list of posts, followers).

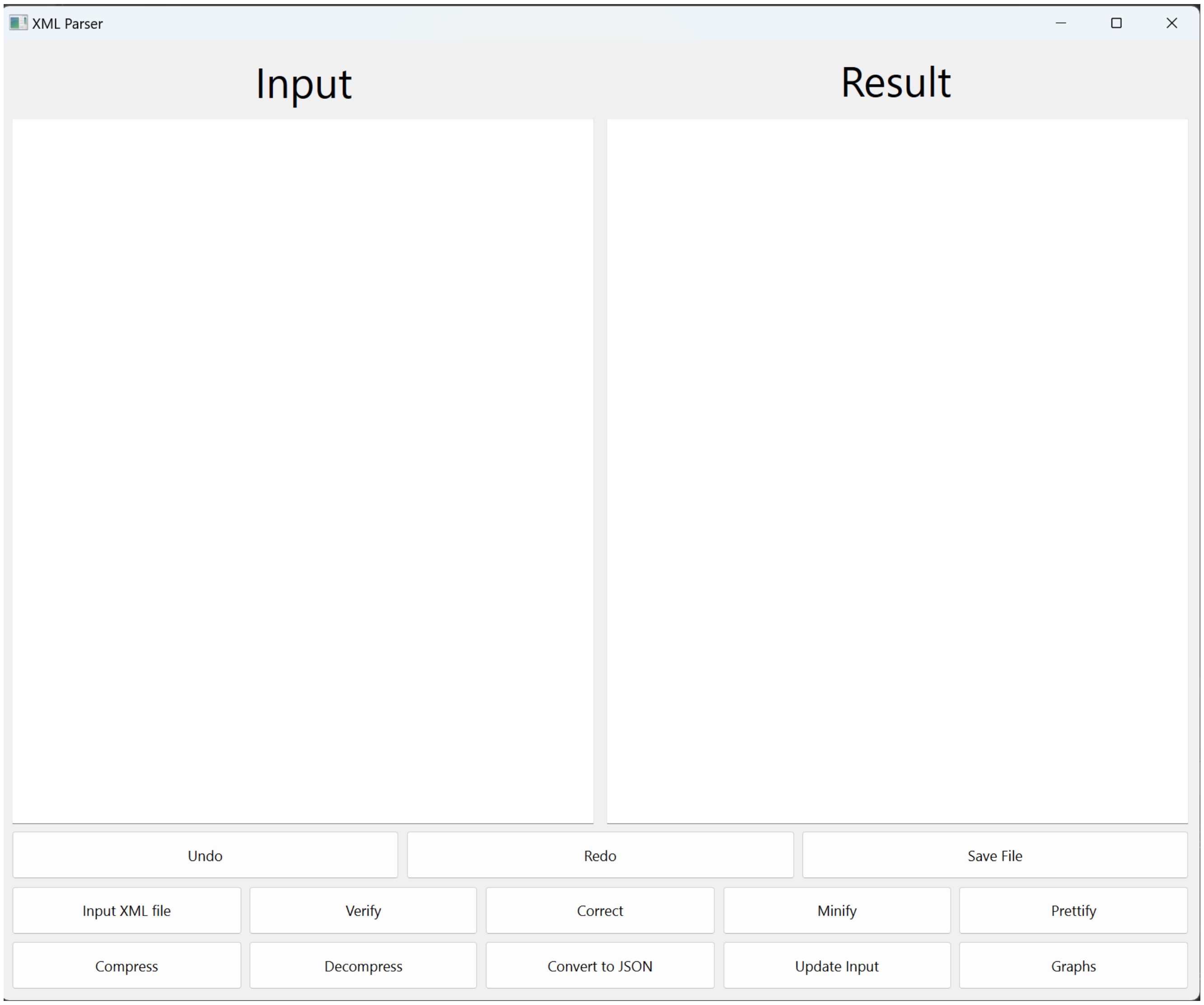
2- Statistics (most followers - most connections - mutual followers - user suggestions).

3- Post search.

Implementation Details

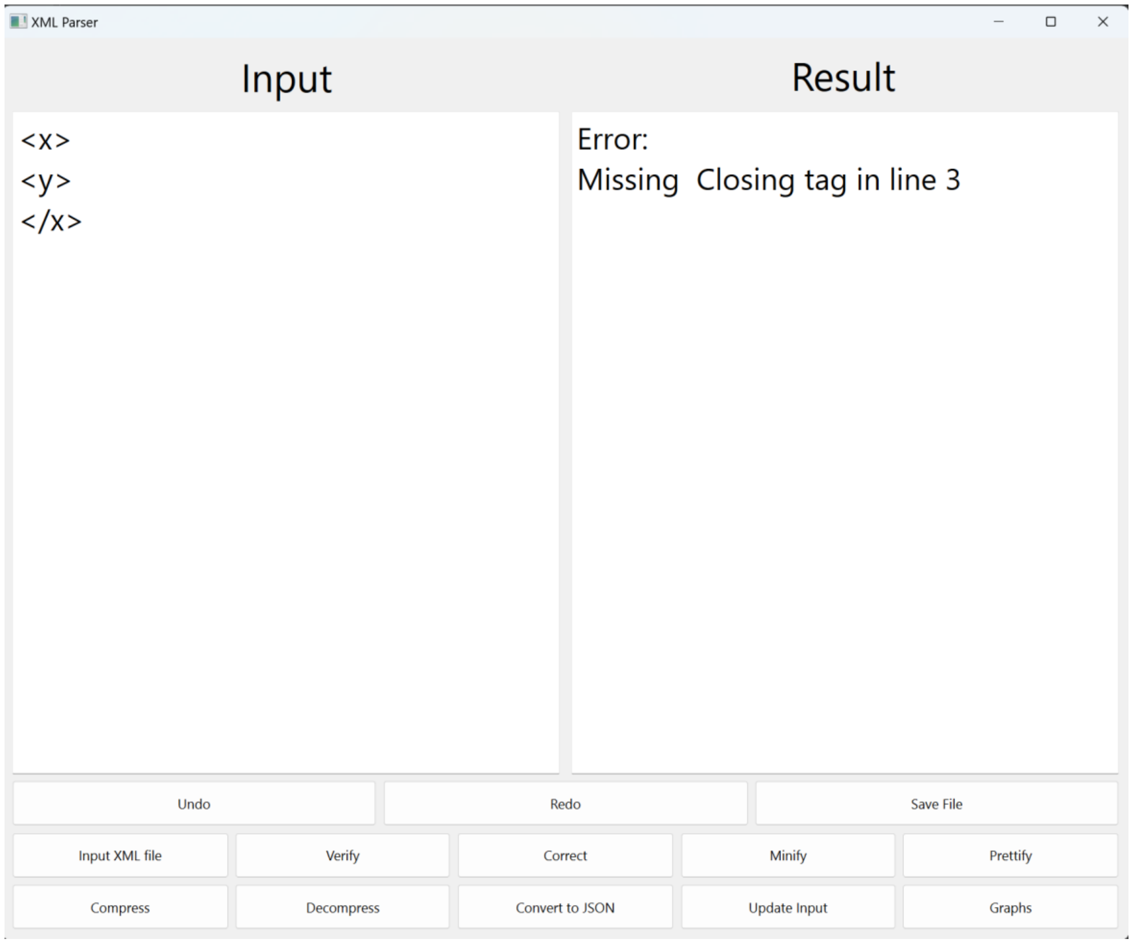
- **Gui Part:**

1- We have used Qt for GUI as we are using C++ for our project. It's the first time for us to work with GUI using C++, so it has been an exploratory experience. First, we installed the open-source version of Qt.
2- We've created two windows. One is called "XML Parser," which can take inputs in two ways: typing or choosing a file. There are a bunch of buttons to minify, prettify, and perform other functions for XML.
3- Then, there is a button that opens a new window called "Graph Representation" that shows the graph and has its functions as buttons.



- **XML Part:**

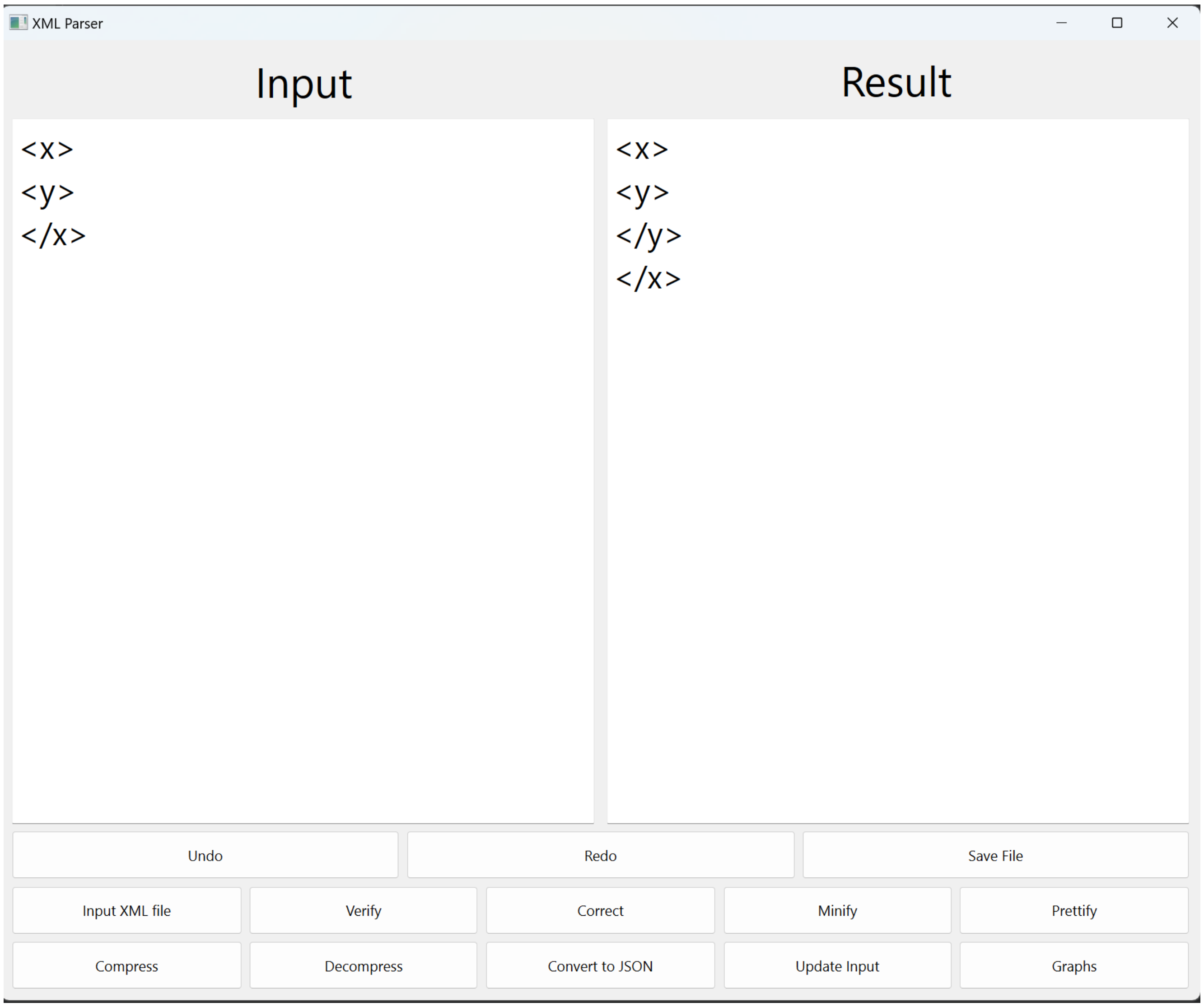
   1- To Verify no errors: we are validating the correctness of XML tags by determining the count of occurrences for a specified XML tag within a given input string. The function utilizes a stack-based approach, mimicking the strategy employed in solving bracket correctness problems. It employs two pointers, one for opening tags and another for closing tags, iterating through the input string. For each encountered opening tag, the function increments a base counter, and for each closing tag, it decrements the counter. The balance of the base counter indicates the correctness of the XML tags.



```
<x>
<y>
</x>
```

```
Error:
Missing  Closing tag in line 3
```

2- To correct errors: The primary function, values_correction, processes a vector of strings representing an XML file, detecting, and correcting errors related to the structure of XML tags. It iterates through the input vector, identifying non-tag lines and evaluating their relationships with adjacent lines. If inconsistencies are detected, such as mismatched opening and closing tags or dummy tags, the code aims to correct them by generating appropriate corrective lines. The errors and corrections are tracked and stored in the 'errors' vector. The second function, divide_string_for_correction, is responsible for dividing a given XML file string into a vector of strings, each representing a distinct tag or content line. It uses a state-based approach, distinguishing between tags and content lines, and extracts individual components while handling potential edge cases.

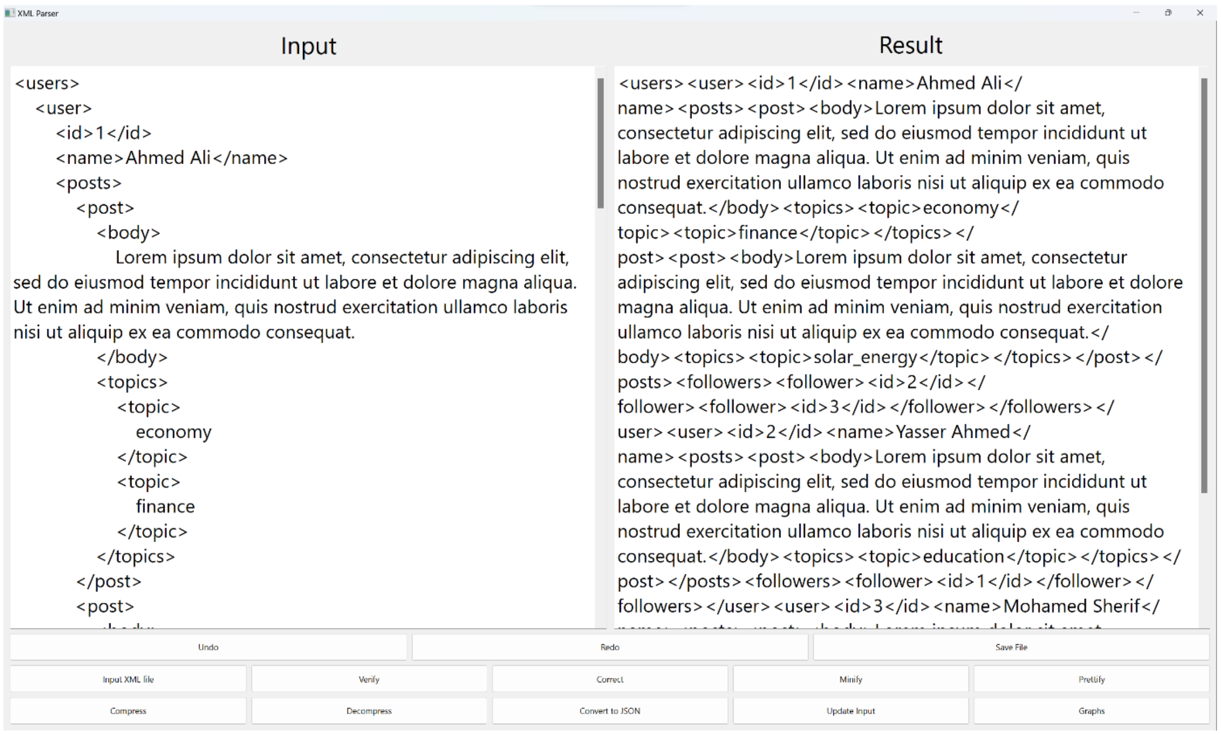


```
<x>
<y>
</x>
```

```
<x>
<y>
</y>
</x>
```

3- Minify:

We path through the file from end to start to remove the excess whitespaces and new lines on the right side of tags, this results in a string with the text being right trimmed and in reverse. Then we pass again from end to start to remove the rest of the whitespaces and new lines from the left side of the string, this results in the string being minified and in the right order again while leaving data between tags intact.



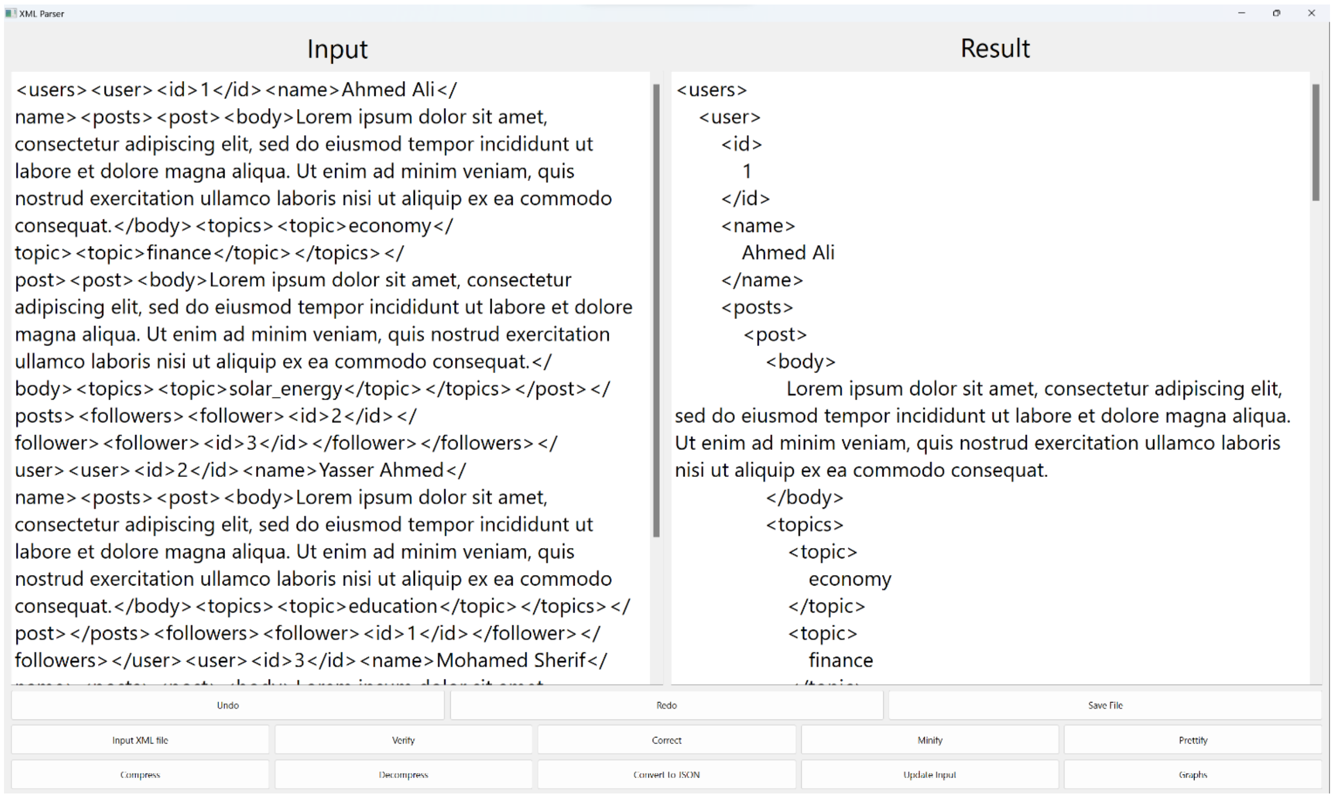
```
<users>
  <user>
    <id>1</id>
    <name>Ahmed Ali</name>
    <posts>
      <post>
        <body>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat.
        </body>
        <topics>
          <topic>
            economy
          </topic>
          <topic>
            finance
          </topic>
        </topics>
      </post>
      <post>
```

```
<users><user><id>1</id><name>Ahmed Ali</
name><posts><post><body>Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.</body><topics><topic>economy</
topic><topic>finance</topic></topics></
post><post><body>Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.</
body><topics><topic>solar_energy</topic></topics></post></
posts><followers><follower><id>2</id></
follower><follower><id>3</id></follower></followers></
user><user><id>2</id><name>Yasser Ahmed</
name><posts><post><body>Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.</body><topics><topic>education</topic></topics></
post></posts><followers><follower><id>1</id></follower></
followers></user><user><id>3</id><name>Mohamed Sherif</
```

4- Prettify:

We pass through the file and try to find the open tags. With each open tag or text content we increase the indentation level before them and with every closing tag we decrease the indentation level before it. It also takes into account whether 2 tags are on the same level or not by detecting if a closing tag is directly followed by an opening tag or not.

The code bypasses all spaces and newlines already available in the file before starting to format it.

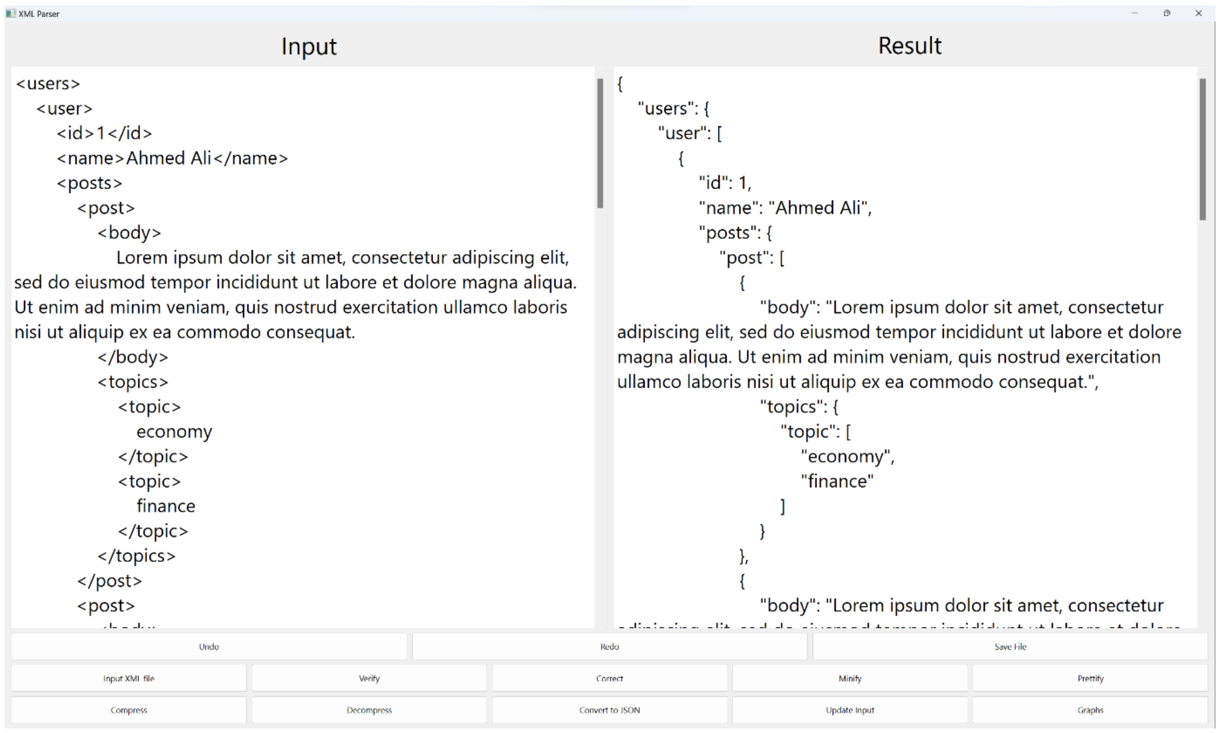


```
<users><user><id>1</id><name>Ahmed Ali</
name><posts><post><body>Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.</body><topics><topic>economy</
topic><topic>finance</topic></topics></
post><post><body>Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.</
body><topics><topic>solar_energy</topic></topics></post></
posts><followers><follower><id>2</id></
follower><follower><id>3</id></follower></followers></
user><user><id>2</id><name>Yasser Ahmed</
name><posts><post><body>Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.</body><topics><topic>education</topic></topics></
post></posts><followers><follower><id>1</id></follower></
followers></user><user><id>3</id><name>Mohamed Sherif</
```

```
<users>
    <user>
        <id>
            1
        </id>
        <name>
            Ahmed Ali
        </name>
        <posts>
            <post>
                <body>
                    Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat.
                </body>
                <topics>
                    <topic>
                        economy
                    </topic>
                    <topic>
                        finance
```

5- Convert To JSON:

We used 2 methods mainly consisting of the same idea:
Check if the tag is an opening tag or closing tag
If the tag contains one or many different children: Then write the name of the child and open a normal bracket (for every child available).
Else if they the children have same name: Then write the name of the child once then open an array and add the children to it.
Else: the child is a string. Then print its value in quotation marks if string or just the number if numeric.
By iterating through the string and determining every one of those conditions on the go with the help of 2 stacks containing the parent node with multiple children and the children count, and every time we detect the end of such tag we decrease the child count by 1 till it gets popped from the stack at 0 count.
By creating a tree where the node has a parent node, vector of children nodes and name string.

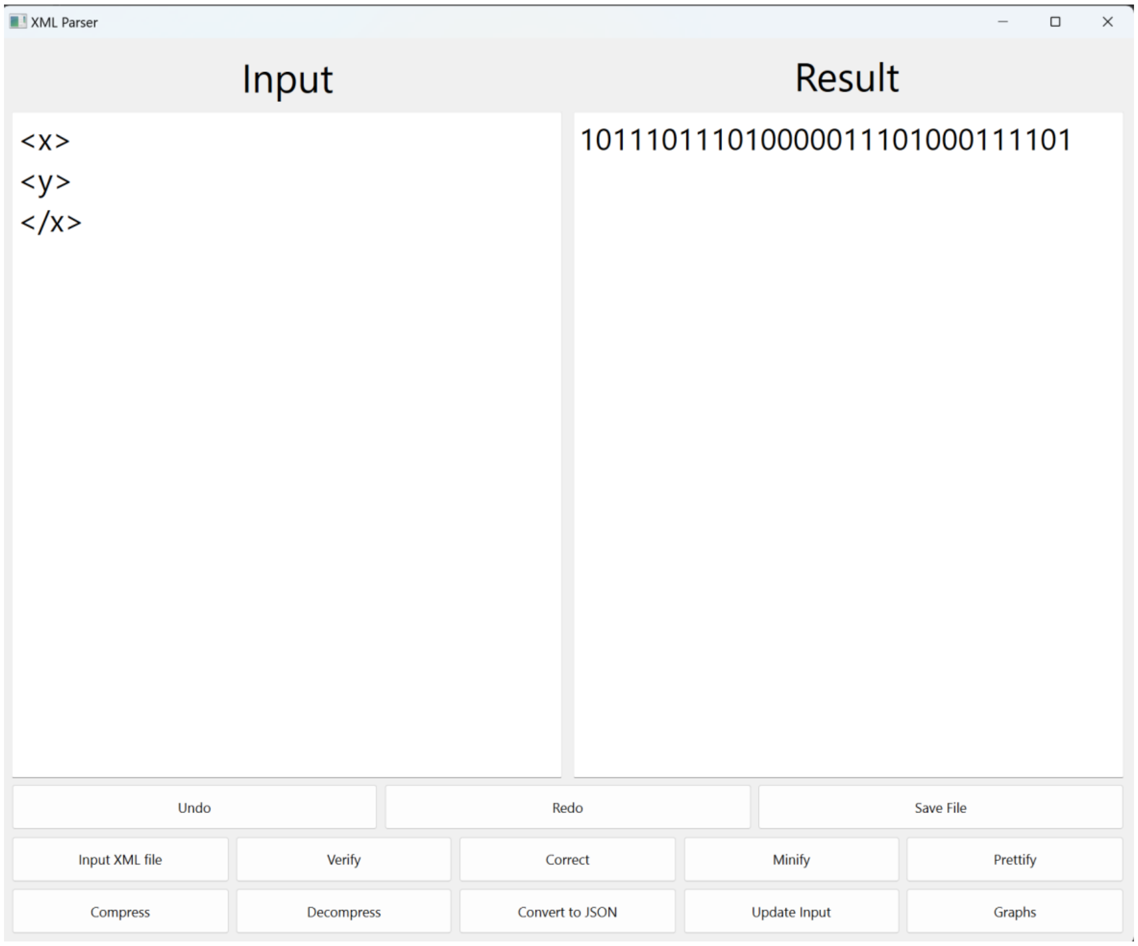


```
<users>
    <user>
        <id>1</id>
        <name>Ahmed Ali</name>
        <posts>
            <post>
                <body>
                    Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat.
                </body>
                <topics>
                    <topic>
                        economy
                    </topic>
                    <topic>
                        finance
                    </topic>
                </topics>
            </post>
            <post>
```

```
{
    "users": {
        "user": [
            {
                "id": 1,
                "name": "Ahmed Ali",
                "posts": {
                    "post": [
                        {
                            "body": "Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.",
                            "topics": {
                                "topic": [
                                    "economy",
                                    "finance"
                                ]
                            }
                        },
                        {
                            "body": "Lorem ipsum dolor sit amet, consectetur
```

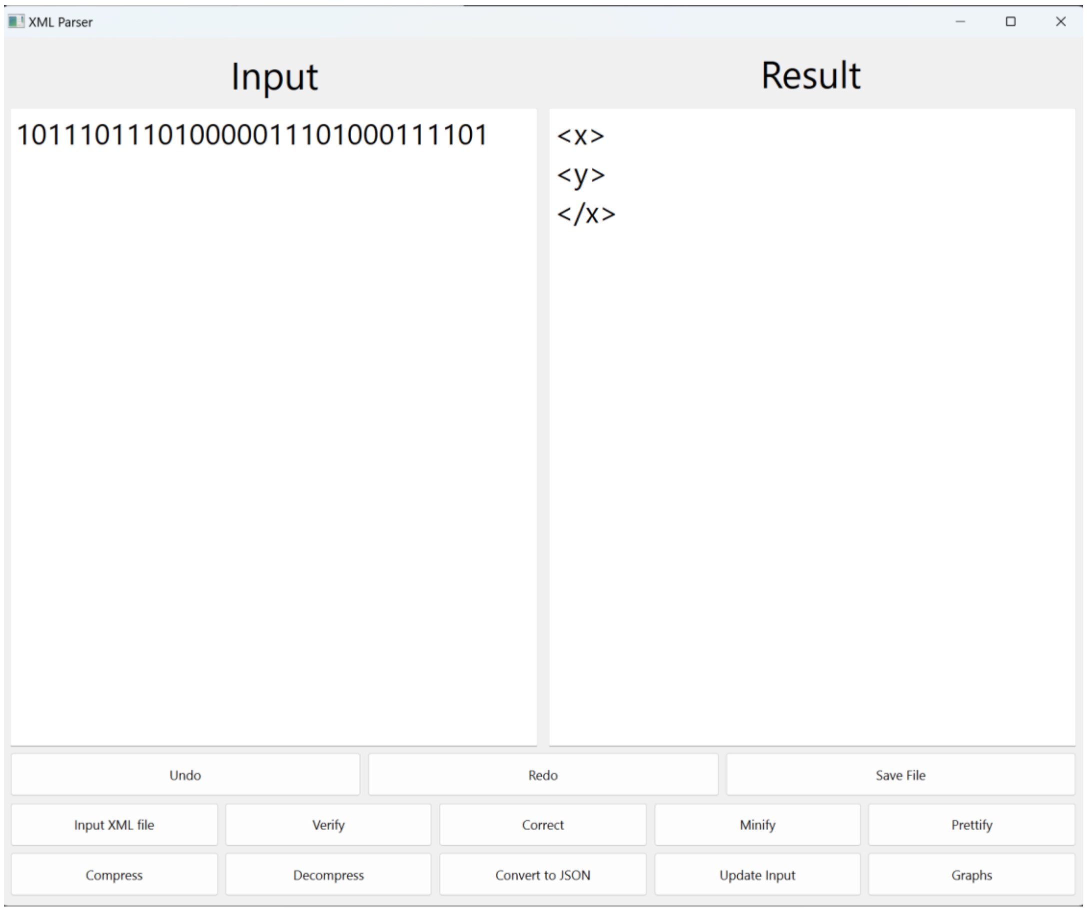
6- Compress Decompress:

Huffman coding is a widely employed technique for lossless data compression, leveraging a binary tree structure known as the Huffman tree. This tree is constructed based on the frequency of each character in the input string, with frequently occurring characters assigned shorter codes and less frequent ones assigned longer codes. During compression, these variable-length codes efficiently represent the original string, resulting in a more compact representation. To decompress, the Huffman tree is utilized to traverse the encoded bitstream, reconstructing the original string with optimal efficiency. This method not only minimizes the overall size of the compressed data but also ensures a unique and unambiguous decoding process, making Huffman coding a versatile and efficient approach for data compression.



```
<x>
<y>
</x>
```

```
101110111010000011101000111101
```

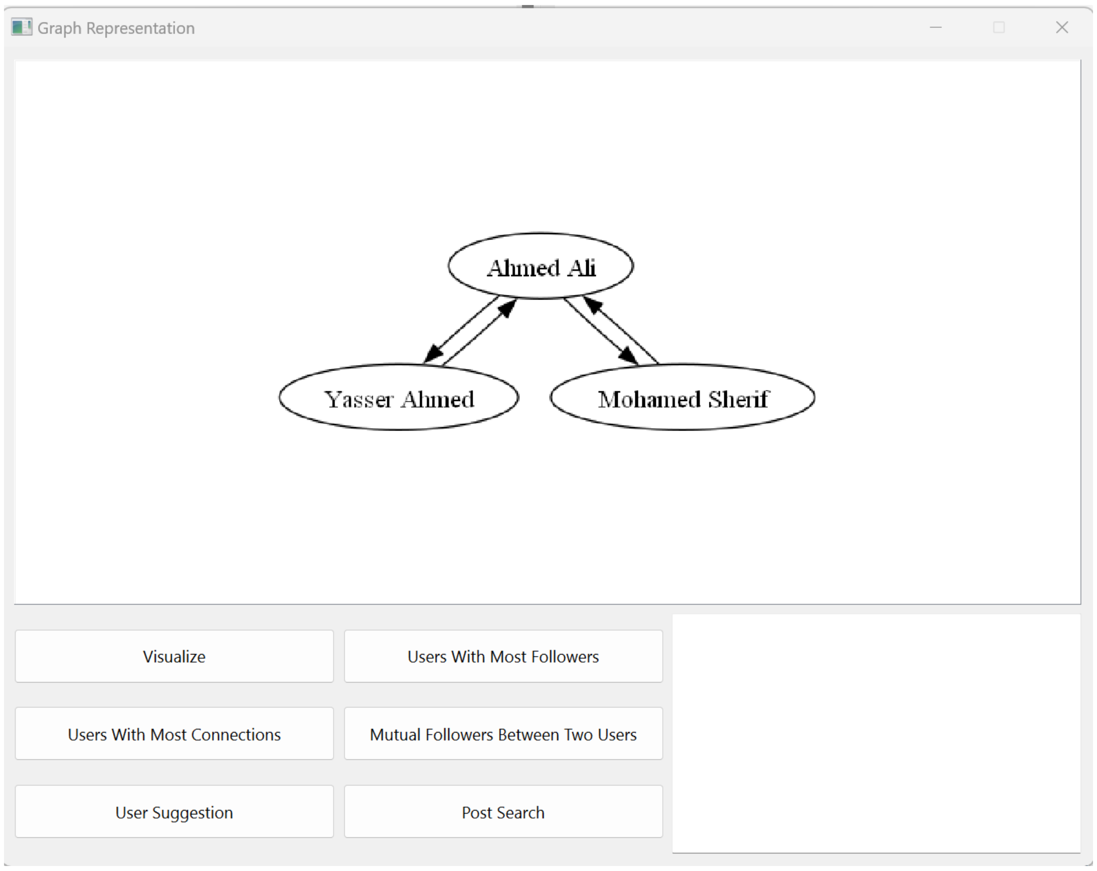


```
1011101110100000111010001111011
```

```
<x>
<y>
</x>
```

- **Graph Part:**

1- We created a social network analysis system designed to model and analyze user interactions within a network.
2- The implementation includes classes for 'Post,' 'User,' and 'Graph,' providing a structured representation of user posts, user profiles, and the relationships between users, respectively.
3- The Graph class incorporates functionalities such as adding users and followers, determining users with the most followers or connections (we made a frequency array so we can make this feature in less time), identifying mutual followers between two users, suggesting connections for a user (using bfs), and searching posts for specific keywords.
4- The code also includes a Network Analysis class responsible for parsing input data, visualizing the network graph, and offering high-level functionalities for network analysis.



Complexity of operations

Function	Time	Space
detect	**O(n)**	**O(n)**
correct	**O(n)**	**O(n)**
minify	**O(n)**	**O(n)**
prettify	**O(n)**	**O(n)**
toJson(String)	**$O(n^2)$**	**O(n)**
toJson(Trees)	**O(n)**	**O(n)**
compress	**O(n)**	**O(n)**
decompress	**O(n)**	**O(n)**

References

https://en.wikipedia.org/wiki/Huffman_coding
https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/