JSON PARSER

Nodes1: It will be storing the components of type within the properties.

Appendl1: It creates a node in the nodes1 linked list which is followed up by the character input through add value function.

Addval\_l1: It takes the input of character variable as well as the position of input. The character helps in entering the code in the sequence after the append is called the position ensures that the character is correctly added to the node.

Display: This function generally helps the cause of printing the linked list for the error correction part.

Del\_l1: This helps in deallocating the linked list again for another input as it stores the linked list into the database.

Nodes2: It will be storing the components of required within the properties.

Appendl3: It creates a node in the nodes2 linked list which is followed up by the character input through add value function.

Addval\_l2: It takes the input of character variable as well as the position of input. The character helps in entering the code in the sequence after the append is called the position ensures that the character is correctly added to the node.

Display: This function generally helps the cause of printing the linked list for the error correction part.

Del\_l2: This helps in deallocating the linked list again for another input as it stores the linked list into the database.

Nodes3: It will be storing the components of description within the properties.

Appendl3: It creates a node in the nodes2 linked list which is followed up by the character input through add value function.

Addval\_l3: It takes the input of character variable as well as the position of input. The character helps in entering the code in the sequence after the append is called the position ensures that the character is correctly added to the node.

Display: This function generally helps the cause of printing the linked list for the error correction part.

Del\_l3: This helps in deallocating the linked list again for another input as it stores the linked list into the database.

Nodes: It will be storing the components of description, type, properties within the properties.

Appendl1: It creates a node in the nodes2 linked list which is followed up by the character input through add value function.

Addval\_l3: It takes the input of character variable as well as the position of input. The character helps in entering the code in the sequence after the append is called the position ensures that the character is correctly added to the node.

Display: This function generally helps the cause of printing the linked list for the error correction part.

Del\_l3: This helps in deallocating the linked list again for another input as it stores the linked list into the database.

Nodes: It will be necessary storage deletion linked list which helps in our program

Append: It creates a node in the nodes linked list which is followed up by the character input through

D: This helps in deallocating the linked list again for another input as it stores the linked list into the database.

Prod (property description): This function takes the file as a character array then as per the quotation and identification of the json or schema files.

Proe (property encrypted type): This function takes the file as a character array then as per the quotation and identification of the json or schema files.

Proi (property ID): This function takes the file as a character array then as per the quotation and identification of the json or schema files.

Prof (reduced part): This function takes the file as a character array then as per the quotation and identification of the json or schema files. It is then asked for checking the essential elemnts of a json formatted files

Proc (common parts of the program): This function takes the file as a character array then as per the quotation and identification of the json or schema files, and deducts the common tittle, signature, description, type and then the property section.

Call: This function is called to ask for encrypted type, description, common parts of the program, property id and then the reduced part one by one.

kv: this is used to store different properties as per different nodes.

Json: It stores the head pointer of the kV as such each json node will consists of complete property set.

createNode: It helps for addition of properties within the node. When derived of json it works for addition of head of kV to json.

Schema: It is the binary search tree which consists of nodes as per the value of signature. It checks along over and then converts the control to the json and then finally to the KV , for deduction of complete Schema.

Long toDecimal: This function helps in converting the extracted hexa decimal signature to integer value for input in the binary search tree (Schema).

insert\_node: The function inserts a node to the kV linked list and with the help of create node.And when json type is asked for then we generally find the similar action but over the Json linked list.

Schema \*insert\_node: This function inserts signature contain json which further contains kV.

searchSchema: It is used to search for the node within the schema files. It is actually a help function for the major part of the code.

getSchemaCount: Asper the name suggest it is used to find the total number of node for further processing of the storage part.

GetjsonCount: It counts the number of json node for error check and addition of further node.

PrintSchema: It prints the elements from the storage data structure as per the original json format. But it only prints the common attributes. The print properties prints all the properties as per the given call from schema.

Get from: this function has two parts one is used to send the head of kv to the print properties part as such to execute it command properly .

KV \* merge : It is used to use two merge two schema types to get a new schema .

Print : The code consists of various print function as like print json is used to print all the different property list present in the storage data structure , print kv prints all the present properties in it , print schema prints the complete binary search tree .

setRequired : to add the extracted is required part to the kv node.

verifyRequired : It checks for the presence of required elements , and checks if the required condition is verified or not .

checkRedendancy : Checks if the merge schema is already present or not .