**Term Project Report**

My code converts all given inputs to given correct outputs.

My Code in General:

body : statements

statements : statement | TAB statement | statements NWLINE statements

statement : assignment | If\_else\_elif | //empty

If\_else\_elif : If Condition COLON | Else Condition COLON | Elif Condition COLON

Condition : assign COMPARISON assign

assignment : VAR EQUAL assign

assign : VAR | INT | STRİNG | FLOAT | assign OP assign

In the assign part I put some flags such as (integer , str , flo ) so that I can decide which vector will I be pushing them and passed the data to the upper caller with $$. In Var section of the assign I check if the variable is in any of the vectors (variable\_int, variable\_float, variable\_ string) and name it, flag it accordingly.

In the assignment part I decide the var’s type by checking assign flags that I previously set and name the var accordingly while pushing the var to the correct vector. In addition to that I check whether the var is already in the vector or not, if it is already in, I do not push the var again. Also, I make the flags 0. Lastly, I pushed the string to $$.

In Condition rule I check If the comparison is a valid comparison by checking the flags. Then if there is a comparison mismatch It prints an error. If there is no error, it pushes the string to $$. Then makes the flags 0.

In If\_else\_elif rule I check if its an if elif or else. If its and if it makes ifopen flag 1 and pushes the string to $$. If it’s an else it makes the ifopen flag 0 and checks whether if there were an if opened before with cheking the ifloopcount variable, if its zero then there were no if before the else so it will give an error if there is no error it pushes the string to $$. If it’s elif then it checks again the ifloopcount to know if there were an if before, if its 0 then it prints an error, if there is no error it pushes the string to $$.

In statement rule I check whether it’s an assignment, if\_else\_elif or an empty line, if its and assignment it pushes the string to $$. And makes the flag Ifelseelif 0. If it’s an If\_else\_elif It pushes the string to $$. And makes the flag Ifelseelif 1 to indicate that the statement is an if else or elif. If it’s an empty line, this line is for the code to not break when there is an empty line in between. This line makes the ıfelseelif flag 0 and pushes the string to $$.

In the statements rule I check whether it’s a statement, TAB statement or statements NWLINE statements. In statement first I check whether the ifloopcount is greater then zero or not. If its greater than zero, then it pushes tab to a temp string. It pushes tab, ifloopcount -1 times. Then it pushes close parenthesis to the temp. Then makes ifloopcount 0. Then checks if it comes from an If\_else\_elif statement if its then increases the ifloopcount and makes ifloop flag 1 and opens a parenthesis which has right number of tabs. Else it makes the ifloop flag 0. In the end it pushes all temps to $$. If it’s a tab statement then it prints the tabs then checks if there is a tab inconsistency by looking to ifloopcount, if there is an inconsistency prints an error. If the ifloopcount is greater then tab count, then it closes parentheses and makes the ifloopcount equal to tab count. And in the last part checks if its from an If\_else\_elif statement then increases the ifloopcount and makes ifloop flag 1 and opens a parenthesis which has right amount of tabs. In the final pushes the temp to $$. If it’s a statements NWLINE statements, it pushes the string to $$. This line is for taking care of lines that includes \n.

In the body rule all code is merged together but first it checks for empty if else elif block error by looking through the $1(statemen) string if there is an open parenthesis and close parenthesis following each other it prints an error. Also calculates the newline for printing error. Then it checks for an elif after else error and else without if error. It does this by iterating over the string. When it finds an if it saves its tab number to an iftabcount vector. Then when a else shows up it looks to the iftabcount vector and compares each element with the current tab number(else) if they are equal pops the element from iftabcount (means that that if closed by else). If no match is found it gives an error. For the elif after else error it looks to iftabcount and in there is a match it continues else it prints an error. After that it prints the main first it iterates over float, int, string vectors and prints those variables then it prints the all string of $1(statement).