Assignment 2: We will build a symbol table here. Symbol table is just a regular table, and it used to store variables and other symbols in a structured way. We will create the symbol table here by omitting many actual details from an actual real-life symbol table in order to reduce the complexity of the assignment. Furthermore, we will consider only the variable declarations only. We will have only four columns in our symbol table. 1) SL 2) NAME 3) TYPE 4) VALUE

SL	NAME	TYPE	VALUE

For example, if a user enters a statement for variable declaration such as, "**int a=5;**". We will have to analyze and split the statement and eventually figure out data that required (SL, NAME, TYPE, and VALUE) . However, we don't need the data for SL since we can give a serial to a row in a first come first serve basis or on other words, make SL autoincrement. OK, look at the statement now, we clearly have the NAME which is "a", we have TYPE which is "int", and finally we have a VALUE "5" as well. Nevertheless, from the statement we can easily understand that there is no SL given. Do we need SL from user? No. As we have said earlier, just give a SL by increment it with previous SL+1. Don't worry check the following instances given below and you'll surely understand (IF YET NOT, WATCH THE CLASS RECORDING).

Input 1: int a;
Output View 1:

SL	NAME	TYPE	VALUE
1	a	int	NULL

Input 2: float karim=5.5;
Output View 2:

SL	NAME	TYPE	VALUE
1	a	int	NULL
2	karim	float	5.5

Input 3: float x,y=7.2;
Output View 3:

SL	NAME	TYPE	VALUE
1	a	int	NULL
2	karim	float	5.5
3	x	float	NULL
4	y	float	7.2

Input 4: int position,initial=0, rate=2;
Output View 4:

SL	NAME	TYPE	VALUE
1	a	int	NULL
2	karim	float	5.5
3	x	float	NULL
4	y	float	7.2
5	position	int	NULL
6	initial	int	0
7	rate	int	2

[Please turn over]

Just like Assignment 1, you have to create an **infinite loop** for the menu so that we can reuse its functionalities in one go. Menu items are:

1) Insert Data (i.e. int a=5;)
2) Remove (use NAME to remove a row i.e. If user puts “a”, the entire row of “a” will be removed)
3) Update (use NAME to remove entire row, user can update TYPE and VALUE only)
4) Display (diplay the full array/map/structure in a table structure;)
5) Exit

**You can add any other feature for practice.

** Think NAME is Unique field. If there is a name “karim” already exists , then should not be another row with the name “karim”.

** Handle basic error and give warning accordingly.

** Make it user-friendly and suggest user what to do, what not to do and show reports if there is any problem during operation such as in insert, remove, and update.(Just print warning, suggestion, report WHERE NECESSARY)

**** IF you do it by yourself and you cannot complete it 100%, don’t worry I will try to give a better mark. By contrast, if you copy, I will have to give you STRAIGHT ZERO without any further considerations.**