**symbol.h**

/\* define the row \*/

typedef struct ROW \*Row;

/\* create a row with pos, value, type and next row\*/

Row createRow(int pos, char \*value, int type, Row next);

/\* set the next row \*/

void setNextRow(Row currentRow, Row next);

/\* get the next row \*/

Row getNextRow(Row currentRow);

/\*create a table\*/

typedef struct TABLE \*Table;

/\*define a table with head, tail, and size\*/

Table createTable(Row head, Row tail, int size);

/\*insert row into table\*/

void insertRow(Table currentTable, int pos, char \*value, int type);

/\* print out the symbol table\*/

void displayTable(Table currentTable);

/\*check if the row already exist in the table\*/

bool isRowExist(Table currentTable, Row newRow);

/\* free the row \*/

void freeRow(Row currentRow);

**proj.h**

/\* lexanAnalyzer analyzer \*/

int lexanAnalyzer();

/\* check if t is equal to looked \*/

void match(int t);

/\* call match and check syntax error\*/

void factor();

/\* check match with ‘\*’ or ‘/’ \*/

void term();

/\* check match with ‘+’ or ‘-’ \*/

void expression();

/\* assign the statement with ‘=’\*/

void assignStmt();

/\* build string array \*/

char \*getWord(char c, char \*word);

/\* build comment array \*/

void getComment(char c, char \*line);

/\* build number array \*/

void getNumber(char c, char \*number);

/\*call all the function and run the program\*/

void runProgram();

/\* check if the identifier if valid \*/

bool valueIdentiferCheck(char \*word);

/\* check if the number if valid \*/

bool numberCheck(char \*number);

/\* assign signs into the table \*/

void assignsign(char ch);

/\*insert the value to the table\*/

void insertValue(char c);

/\*get sentence after ‘=’\*/

void output(char c);

/\*helps to print the output statment\*/

void compiling(char \*sentence);