**CSCI 301 Moustafa Elsayed**

**Computer Science 2**

**Project 11: IMPLEMENTING AND MANIPULATING EXPRESSION TREES**

**Introduction**

Binary Expression trees can be very useful in quickly accessing, manipulating and printing data. In this program we use binary expression trees in order to read data from an input file, then build a binary expression tree based on that expression, then by traversing through the binary expression tree in different orders, we are able to print an expression in three different forms, the infix form, the post order form, and the pre order form. Then the program calculates the value of the expression. The three forms of the expression and its values are printed in the terminal.

**Data Structures**

This program uses 11 data structures:-

* A char type “optr” in the header file: in the structure of the node this variable is used to hold the operators of the expression.
* An int type “operand” in the header file: in the structure of the node this variable is used to hold the operands of the expression.
* A pointer node type “left” in the header file: in the structure of the node this node is used to point towards the left node.
* A pointer node type “right” in the header file: in the structure of the node this node is used to point towards the right node.
* A pointer node type “root” in the header file: this node is used to hold the value of the root of the binary expression tree
* A char type “ch” in the source file: in the r\_build() function this variable is used to hold the characters in the input file.
* A pointer node type “temp” in the source file: in the r\_build() function this node is used to hold the new items and become a part of the binary expression tree.
* An int type “val1” in the source file: in the r\_value() function this variable is used to hold the left integer of an expression
* An in type “val2” in the source file: in the r\_value() function this variable is used to hold the right integer of an expression
* An ifsteam “file” in the main file: in the main function main() this variable is used to hold the input file
* A string “name” in the main file: in the main function main() this variable is used to hold the user input of the name of the file.

**Functions**

This program uses 11 functions:-

* Void value() in the header file: this function prints the returned value of the function r\_value() with the root as its parameter
* Void build() in the header file: this function takes the ifstream variable as its parameter and deletes the previous binary expression tree, then calls the function r\_build() to build another binary expression tree
* Void infix() in the header file: this function calls the function r\_infix() with the root as its parameter
* Void prefix() in the header file: this function calls the function r\_prefix() with the root as its parameter
* Void postfix() in the header file: this function calls the function r\_postfix() with the root as its parameter
* Node\* r\_build() in the header file: this function checks if the parameter node’s data is an integer, if it is an integer then the function will change the data in the node from a character to an integer and set both right and left node to null. If not then the function will recur and return to the left node, assign the sign to the node then recur and return to the right node. Finally the function returns the ndoe.
* Void destroy() in the header file: this function deletes a binary expression tree by, checking if the input node is null, if it is not null then the function recurs to the left node, then recurs to the right node, then deletes the current node.
* Void r\_infix() in the header file: this function prints the expression in the binary expression tree by checking if the parameter node is null, if the node is null then the function prints the operand, if not then the function recurs the left node, then print the operator, then recurs the right node.
* Void r\_prefix() in the header file: this function prints the expression in the binary expression tree by checking if the parameter node is null, if the node is null then the function prints the operand, if not then the function prints the operator, then recurs the left node then recurs the right node
* Void r\_postfix() in the header file: this function prints the expression in the binary expression tree if the parameter node is null, if the node is null then the function prints the operand, if not then the function recurs the left node, then the right node, then prints the operator.
* Int r\_value() in the header file: this function return the value of the expression in the binary expression tree, by checking if the left node is null, if the node is null then the function returns the operands, if not then the function assigns the left integer to the returned value of the recurred function to the left, and assigns the right integer to the returned value of the recurred function to the right, then the function checks the operator of the node, and depending on the operator the function will return the value of its corresponding arithmetic expression.

**Main function**

In the main function the user is asked to enter the name of the input file, if the name is not “input.txt” then the program will print “invalid file name” , if the name is correct the function will open the file, then open a loop that will stop at the end of the input file, calling the build function, then the infix function, then the prefix function then the postfix function and finally the value function.