Name : Aadil Mohamed Puthiyaveetil
Reg No: 22BCE2436
Course Name: Compiler Design Lab
Course Code : BCSE307P
Course Faculty: SABYASACHI KAMILA

DIGITAL ASSIGNMENT-2

Question

Given a program's source code, your task is to construct an Abstract Syntax Tree (AST) for each functional block of the code (assignments, conditions and loops only).
You can write your programs in any language.
The tasks are as follows:

Tasks:

● Input: A program file containing the function for finding the factorial of a
number. It means that the input will be a file named, for example, factorial.c
(which can be written in any language) which will contain the function to
calculate the factorial of a number.
[Note - The program should also run for any other program input]
● Outputs:
○ A set of ASTs (displayed either in pre-order traversal or in other suitable
representations) for each functional block written inside the factorial.c file.

Factorial.c

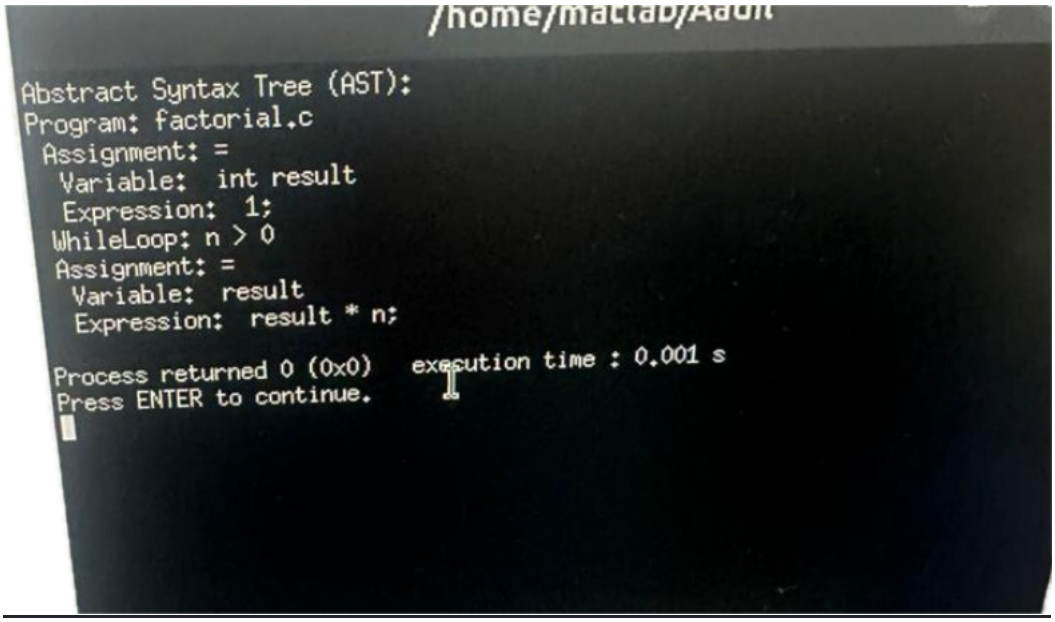
```
int factorial(int n) {
 int result = 1;
 while (n > 0) {
 result = result * n;
 n--;
 }
 return result;
}
```

Main Code

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <sstream>
struct ASTNode {
 std::string type;
 std::string value;
 std::vector<ASTNode*> children;
 ASTNode(std::string type, std::string value) : type(type),
value(value) {}
};
class AST {
public:
 void printAST(ASTNode* node, int depth = 0) {
 if (node == nullptr) return;
 for (int i = 0; i < depth; ++i) std::cout << " ";
 std::cout << node->type << ": " << node->value << std::endl;
 for (ASTNode* child : node->children) {
 printAST(child, depth + 1);
 }
 }
};
ASTNode* parseAssignment(const std::string& line) {
```

```
 // Parse assignment and create corresponding ASTNode
 std::string var = line.substr(0, line.find('='));
 std::string expr = line.substr(line.find('=') + 1);
 ASTNode* assignmentNode = new ASTNode("Assignment", "=");
 assignmentNode->children.push_back(new ASTNode("Variable",
var));
 assignmentNode->children.push_back(new
ASTNode("Expression", expr));
 return assignmentNode;
}
ASTNode* parseWhileLoop(const std::string& condition) {
 ASTNode* whileNode = new ASTNode("WhileLoop", condition);
 return whileNode;
}
int main() {
 std::ifstream infile("factorial.c");
 std::string line;
 AST ast;
 ASTNode* root = new ASTNode("Program", "factorial.c");
 while (std::getline(infile, line)) {
 line = line.substr(0, line.find("//")); // Remove comments
 if (line.find("while") != std::string::npos) {
 std::string condition = line.substr(line.find('(') + 1, line.find(')') -
line.find('(') - 1);
 ASTNode* whileNode = parseWhileLoop(condition);
 root->children.push_back(whileNode);
 } else if (line.find('=') != std::string::npos) {
 ASTNode* assignmentNode = parseAssignment(line);
 root->children.push_back(assignmentNode);
 }
 }
 std::cout << "Abstract Syntax Tree (AST):" << std::endl;
 ast.printAST(root);
 return 0;
}
```

Output


/home/matlab/Aaun
Abstract Syntax Tree (AST):
Program: factorial.c
Assignment: =
Variable: int result
Expression: 1;
WhileLoop: n > 0
Assignment: =
Variable: result
Expression: result * n;
Process returned 0 (0x0) execution time : 0.001 s
Press ENTER to continue.