AVL – Algorithm Visualization Language
Team 1

Qianxi Zhang (qz2204) – Project Manager
Shining Sun (ss4314) – Language Guru
Yu Zheng (yz2583) – System Architect
Qinfan Wu (qw2168) – System Integrator
Jiuyang Zhao (jz2538) – System Tester

May 11, 2014

Contents

1 Introduction

1.1 The Problem

AVL (Algorithm Visualization Language) is an intuitive educational programming language that concentrates on the algorithm teaching and learning in the way of visualization. The output of AVL is a sequence of graphics or video, which describes how an algorithm works. AVL provides varying degrees of flexibility and programmability. It helps beginners understand a complicated algorithm in an intuitive way. It assists professionals to visualize their self-design algorithms, where AVL provides visualization control interface.

AVL breaks the limitation of current similar languages. It is highly focused on the robust algorithm guarantee as a visualization language. The shining breakthrough is to provide a series of complicated low-level drawing operations, which is specially designed for algorithm visualization.

AVL is also an educational language. It provides simple and clear grammar with C++ style. Benefiting from the easy-to-use features of AVL, the target users, teachers and students, are able to easily demonstrate or learn the algorithms.

1.2 Why Use AVL

There are several existing approaches to algorithm visualization. First, in order to visualize algorithms such as Quicksort, we can just write a C++ program that manipulates an array and use third-party libraries to display the animation. Second, there are lots of videos which show different kinds of algorithms and data structures, such as sorting (http://www.sorting-algorithms.com/) and trees (http://www.qmatica.com/DataStructures/Trees/AVL/AVLTree.html). There are also some visualization techniques, such as Animal (http://www.algoanim.info/AnimalAV/). However, these approaches all have certain kinds of drawbacks and limitations.

1.2.1 Complicated low-level drawing operations

Some approaches require users to implement the low-level drawing operations. People have to generate the video frame by frame, take into account where to put the array on the screen and think about what third-party library is available to draw the variables and operations. These low-level operations are complicated, making algorithm visualization difficult and inefficient.

1.2.2 Limitation on available algorithms

Many approaches only allow visualization of a limited number of existing algorithms. They do not allow user to visualize their own algorithms.

1.2.3 Hard to learn and use

Some approaches, such as Animal (http://www.algoanim.info/AnimalAV/), provides a good animation tool for general-purpose animation. However, learning to use these tools is also hard and takes extra time for people who only have a basic understanding of C programming.

1.3 Target Users

The target users of our algorithm visualization language are teachers and students in computer education in high school. This language can also be utilized in basic programming courses in college like Data Structure.

Teachers with rich programming experiences can use our AVL to generate the video for the algorithms they prepare to demonstrate in class. With our language, they do not need to draw the picture describing their algorithm in the old fashioned way. In addition, because our languages are highly flexible, teachers can visualize whatever algorithm they want. For example, when preparing a teaching material, you may easily get the picture for Quick Sort from internet. But its difficult to get one for Randomized Quick Sort. Then our AVL may help you at this time. You just need to write a Randomized Quick Sort liked program. Our language will generate the video for you automatically.

Our language is also a good tool for the students or beginner in coding. Students with no programming background can use our built-in library such as Stack and LinkedList. Operation of these built-in data structures are well encapsulated, but we can generate the video telling you what is happening inside.

1.4 Properties

1.4.1 Visualization

According to research, visual information accounts for more than 70% of all the information human perceive. And that is our origin motivation of inventing AVL. The most significant feature of AVL is, as its name indicates, visualization. Instead of focusing on the outcome of algorithms as normal programming languages do, AVL concentrates on the process of algorithms. With AVL, users can actually see how their programs run, via which they can have a better understanding of their programs and algorithms.

1.4.2 Convenient

The primary goal of AVL is to hide the complex visualization process behind the code for user to write. By using AVL, user can focus on the algorithm and write C-style code without thinking how to visualize it. The visualization process, such as where to draw the array on the screen and how to highlight a part of a array, is generated automatically when the code is compiled.

1.4.3 Flexible

The most important feature of AVL is visualization. But users of AVL may not what to visualize everything they write. Therefore, AVL has some key words to let the uses define which part of their program to visualize. It gives the users a flexible choice to generate their own animations. For example, some user may be curious about the difference between merge sort and insertion sort, and he can create two animations with AVL to compare them. While in other cases, sorting might just be a subroutine of a complex program, and thus can be ignored in the animation.

1.4.4 Simple and Familiar

Since the users of the AVL are professors and students who teach and learn algorithms, respectively, we want to design AVL as a simple language that can be used without extensive training.

We keep AVL as a similar language to C++ which means that the programmers can migrate easily to our language. Focusing on the problem of algorithm visualization, we must add new features such as keywords mentioned in part "Flexible". Moreover, considering the fact that student users may be entry level programmers and the limitation of visualization, we remove some unnecessary complexities of C++ and add some build-in libraries which include general data structures and algorithms for the beginners.

1.4.5 Robust

AVL is designed for creating clearly and understandable animations of algorithms, so it provides compile-time checking of the code to prevent violations such as exceeding array bounds. However, for the purpose of flexibility, AVL cannot check whether the program is "correct". For example, the user may write a sort algorithm whose result is still unsorted, or the user may write an infinite loop which makes the program never end. To solve this problem, we simply create the animation as the user defined in their program and let them to find the error by themselves, which, on the other hand, makes AVL a debug tool for students learning algorithms.

2 Language Tutorial

This section is a short tutorial of the Algorithm Visualization Language. In this section, we will go through some example programs that demonstrate our language and show how to use our language to visualize some simple algorithms.

This tutorial will focus on basic data types, expressions, control flows, inputs and outputs so that users can get started with AVL quickly. In order to provide users a way to visualize algorithms by writing C-style code, the syntax of AVL is very similar to C. Users who have a basic understanding of C programming would be able to write AVL programs after reading this tutorial.

AVL is an intuitive educational programming language that concentrates on the algorithm teaching and learning in the way of visualization. The output of AVL is a sequence of graphics or video, which describes how an algorithm works. AVL provides varying degrees of flexibility and programmability. It helps beginners understand a complicated algorithm in an intuitive way. It assists professionals to visualize their self-design algorithms, where AVL provides visualization control interface.

AVL breaks the limitation of current similar languages. It is highly focused on the robust algorithm guarantee as a visualization language. The shining breakthrough is to provide a series of complicated low-level drawing operations, which is specially designed for algorithm visualization.

AVL is also an educational language. It provides simple and clear grammar with C++ style. Benefiting from the easy-to-use features of AVL, the target users, teachers and students, are able to easily demonstrate or learn the algorithms.

First, let us examine the input and output of the compiler.

2.1 Input and Output

The purpose of AVL is to help user create a program which visualizes a particular algorithm without much effort. The user creates a `.avl` file as input, which contains AVL code that describes the algorithm. Then the `.avl` file is compiled by our compiler and an executable is created. When user runs the executable, they would see a window that displays the algorithm procedure.

We provide a program avl which acts as the compiler.

```
$ avl --help
Usage: avl [-h|-o|-t] file
Options:
    -h --help            Display this information
    -o --output=<file>   Compile and place the executable into <file>
    -t --translate       Translate the source files into c++

    Use at most one option at a time.
```

The input filename should have the extension `.avl`.

Assuming the input file is `test.avl`, the following command will create an executable file with the default output name `a.out`:

```
avl test.avl
```

Then the user can run `./a.out` to display the algorithm procedure.
The user is also allowed to specify the output name of the executable using the -o option:

```
avl -o test test.avl
```

For debugging purpose, if we want to examine whether the compiler works correctly, we could invoke avl with -t option to translate the input file into a C++ file, which is the intermediate result:

```
avl test.avl -t
```

This would generate a C++ file with name test.cpp.

2.2 Hello World

A Hello World program in AVL creates images of the characters "Hello world" on the screen, and images of an integer array on the screen. This program is for demonstration of the basic visualization feature of the program.

```
int main()
{
    <begin_display>
        display char str[] = {'H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd'};
        display int a[] = {1, 2, 3, 4};
    <end_display>

    return 0;
}
```

As you may already notice, the code is similar to C/C++ code. This is actually a feature of AVL: programmers do not need to spend too much time to learn new grammars. As long as a programmer knows C/C++, she or he will have little obstacle picking up AVL.

Now let me briefly explain the program. The first line declares a main function, which is the entrance of any program. On line 2 and 5, there are a `<begin_display>` and a `<end_display>`. This means anything in between this two tags are to displayed on the screen. On line 3 and 4, a char array and a integer array are declared, and `display` keyword indicates these arrays are displayable on the screen. Variables are not displayable by default, unless it has a `display` key word before it. Notice that the `display` keyword does not have to be put before variables while being declared. A variable can be declared as non-display while later changed to display.

2.3 Types and Variables

AVL is a strongly typed language. For the current version, it supports the following types:

```
int
char
bool
index
string
array
```

Among them, array is the only derivative type. There could be array of ints, or chars, etc. Most of them are the same with C/C++, while there are several things to be noticed.

1. If a programmer wants to display a string on the screen, she should use array of chars, instead of `string`. The type `string` are supposed to be used to print things in terminal, e.g. for debug use.

2. The type `index` is used for indexing elements inside an array. While it is similar to the `int` type , the purpose of having this type is to highlight certain elements inside an array. Therefore, whenever we want to index an array, we should use `index` instead of `int`.

3. The type `array` is not similar to C/C++ arrays in a number of ways. First, the length of an array in AVL can be obtained by calling `len(arr)`. Also, subarray of an array can be conveniently obtained by calling `arr[i : j]` , where `i` and `j` are two indices, and the array obtained is the subarray of `arr` that starts at index `i` and ends at index `j-1`. Lastly, if an array is declared without initialization, it is set to be the default value of its type. For example, the following statements creates an array of five 0s:

    ```
    int a[5];
    ```

2.4 Display Hierarchy

As described in the Hello World part above, a variable is set to be displayable only when it is declared to be `display`. This can happen during or after the variable being declared. However, the `display` key word does not actually display things on the screen. What actually displays things on the screen is the pair of `<begin_display>` and `<end_display>`. Things between this two tags are displayed (if they are declared to be `display`).

A intuitive question to ask is, what if the tag pairs are nested? This behaves differently in two cases. The first case is, the tag pairs are nested within one function. For example:

```
int main()
{
    <begin_display>
    <begin_display>
```

```
        display char str[] = Hello world!;
        display int a[] = {1, 2, 3, 4};
    <end_display>
    <end_display>

    return 0;
}
```

In this program, the tag pairs are nested. In this case, the inner pair are ignored. The second case is, the nesting happens in another function. For example:

```
void quicksort(int a[])
{
    if (len(a) <= 1)
        return;

    display index i = -1;
    display index j = 0;
    display index k = len(a) - 1;

    <begin_display>
        while (j < k) {
            if (a[j] >= a[k]) {
                j = j + 1;
            }
            else {
                swap(a, i + 1, j);
                i = i + 1;
                j = j + 1;
            }
        }
        swap(a, i + 1, k);

        quicksort(a[0 : i+1]);
        quicksort(a[i + 2 : k + 1]);
    <end_display>
}

int main()
{
    <begin_display>
        display int a[] = {5, 2, 3, 6, 1, 7, 4, 9, 8};
```

```
        quicksort(a);
    <end_display>

    return 0;
}
```

In this program, there is a display tag pair inside `main` function. In between this tag pair, another function `quicksort` is called, within which there is another display tag pair. In this case, only the things inside the `quicksort`s display tag pair are displayed, while things inside `quicksort` but outside the display tag pair are not displayed. That means, if there were not a display tag pair inside `quicksort`, then nothing inside `quicksort` is displayed.

2.5 Sample Code

One more piece of sample code is listed below:

```
int main()
{
    display int a[] = {5, 2, 7, 3, 6, 8};

    <begin_display>
        for (display index i = 1; i < len(a); i = i + 1)
        {
            display index j = i - 1;

            while (j >= 0 && a[j] > a[j + 1])
            {
                swap(a, j + 1, j);
                j = j - 1;
            }
        }
    <end_display>

    return 0;
}
```

This code above shows a bubblesort for an integer array.

3 Language Reference Manual

3.1 Introduction

This manual introduces abundant features of AVL visualizing algorithms with OpenGL easier. We firstly describe lexical conventions used in this language. Then we introduce the language syntax, and finally end with a grammar to represent AVL.

3.2 Lexical Conventions

Each program is reduced to a stream of tokens. Each token can be one of the following six classes: identifiers, keywords, constants, operators, separators, and whitespace. Whitespace is the collective term used for spaces, tabs, newlines, form-feeds, and comments.

3.2.1 Comments

In AVL we only support multiline comments defined as word between `/*` and `*/`, nested comments are not allowed.

3.2.2 Identifiers

Identifiers are sequences of characters used for naming variables, and functions. Letters, decimal digits, and the underscore character can be included in identifiers. An identifier cannot start with a digit. Any letters used in identifiers are case-sensitive.

3.2.3 Keywords

The following identifiers are reserved as keywords, which cannot be used for any other purpose.

`bool`	`else`	`int`	`void`
`break`	`false`	`len`	`while`
`char`	`for`	`return`	`<begin_display>`
`continue`	`hide`	`string`	`<end_display>`
`display`	`if`	`swap`	`do`
`index`	`true`		

Table 1: Keywords

3.2.4 Reserved

The following characters are reserved for use in the grammar.

3.3 Types

3.3.1 Basic Types

AVL has several primitive types.

`&&`	`!=`	`(`	`]`	`/`
`||`	`;`	`)`	`!`	`<`
`<=`	`,`	`{`	`-`	`>`
`>=`	`:`	`}`	`+`	`++`
`==`	`=`	`[`	`*`	`--`

Table 2: Reserved characters

int The 32-bit signed int data type holds integer values in the range of -2,147,483,648 to 2,147,483,647.

char A char is used for storing ASCII characters, including escape sequences.

bool A bool encapsulates true and false values.

string A string is a sequence of characters surrounded by double quotation marks. In AVL, string is mainly used for outputting debug information to terminal.

index An index is used for identifying an elements position in an array, and range of positions to for a subarray. It is interchangeable with int. In addition to int, index can be displayed as arrow.

3.3.2 Derived Types

array An array is a data structure that let you store one or more elements consecutively in memory. Only int is allowed to be the data type of elements. `len` returns the length of an array. The index of an array ranges from `0` to `len - 1`. To declare an array, you need to either declare its size or directly initialize this array.

There are three ways to initialize an array. One is the assign value to each elements, and the default value is 0. Another one is to initialize an array with a comma-separated array surrounded by `{` and `}`. The other one is to assign another array or subarray to the newly declared array. `a[s:e]` defines as subarray starting at index s and ending at index e, left-closed, right-open, which means `a[s]` is included and `a[e]` is excluded.

3.3.3 Constants

Integer Constants An integer constant is a sequence of digits, representing a decimal number.

Character Constants A character constant is a single character enclosed within single quotation marks.

Boolean Constants Boolean constants are either `true` or `false`.

3.4 Scopes

3.4.1 Lexical Scope

The lexical scope of an identifier for a declared variable starts at the end of its declaration immediately and lasts until its current scope exits. Duplicate variable names within the same scope are not allowed.

3.4.2 Function Scope

The scopes of variables declared inside a function persist through the function and end upon exiting the function.

3.4.3 Statement Block Scope

The scopes of variables declared inside a statement block persist through the block and end upon exiting the block.

3.5 Syntax and Semantics

3.5.1 Expressions

Primary Expressions Primary expressions are identifiers, constants, strings and parenthesized conditional expressions.

```
primary_expression
    : IDENTIFIER
    | CONSTANT
    | TRUE
    | FALSE
    | CHAR_LITERAL
    | STRING_LITERAL
    | '(' conditional_expression ')'
    ;
```

Postfix Expressions Postfix expressions includes function call, array elements, sub arrays, postfix operators. The grammar of postfix expression is:

```
postfix_expression
    : primary_expression
    | IDENTIFIER '[' conditional_expression ']'
    | IDENTIFIER '[' conditional_expression ':' conditional_expression ']'
    | IDENTIFIER '(' ')'
    | IDENTIFIER '(' argument_expression_list ')'
    | postfix_expression INC_OP
```

```
    | postfix_expression DEC_OP
    ;

argument_expression_list
    : conditional_expression
    | argument_expression_list ',' conditional_expression
    ;
```

Subarray is written in python style, for example `a[2:5]` will return a new array `{a[2], a[3], a[4]}`. The postfix operators we support are `++` and `--`. These two postfix operators have the highest precedence.

Unary Expressions Unary Expression include unary operator and keyword len. len is a unary operator like sizeof in C to get the length of an array. The different is len only return the number of elements in the array. Let us see the grammar:

```
unary_expression
    : postfix_expression
    | INC_OP unary_expression
    | DEC_OP unary_expression
    | unary_operator cast_expression
    | LEN '(' unary_expression ')'
    ;

unary_operator
    : '+'
    | '-'
    | '!'
    ;
```

In AVL, unary (prefix) operator includes +, -, !, ++ and –. The have the second highest precedence.

Types The type we support in AVL are:

```
type_specifier
    : VOID
    | CHAR
    | INT
    | STRING
    | INDEX
    | BOOL
    | type_specifier '[' ']'
    ;
```

Binary Operators The binary operators except for the assignment operator are listed according to precedence in the following table. The closer to the top of the table an operator appears, the higher its precedence. Operator with higher precedence are evaluated before operators with relatively lower precedence. Operators on same line have equal precedence and are evaluated from left to right.

Name	Operator
Multiplicative	`* / %`
additive	`+ -`
relational	`< > <= >=`
equality	`== !=`
logical and	`&&`
logical or	`||`

Table 3: Binary operators

Multiplicative Operators `*` and `/` represent multiplication and division operators. Division with int operands results in discarding the fractional portion of the result.

`%` represents remainder operator. It divides one operand by another and returns the remainder as its result.

```
multiplicative_expression
    : unary_expression
    | multiplicative_expression '*' unary_expression
    | multiplicative_expression '/' unary_expression
    | multiplicative_expression '%' unary_expression
    ;
```

Additive Operators `+` represents the summation of the left operand and the right operand.

`-` represents the subtraction operator that returns the result of left operand subtract the right operand.

```
additive_expression
    : multiplicative_expression
    | additive_expression '+' multiplicative_expression
    | additive_expression '-' multiplicative_expression
    ;
```

Relational Operators The relational operators determine if one operand is less than (`<`), greater than (`>`), less than or equal to (`<=`), or greater than or equal to (`>=`) another operand. It returns a bool value true if the relation is true and false if the relation is false.

```
relational_expression
    : additive_expression
    | relational_expression '<' additive_expression
    | relational_expression '>' additive_expression
    | relational_expression LE_OP additive_expression
    | relational_expression GE_OP additive_expression
    ;
```

Equality Operators The relational operators determine if one operand is equal to (`==`), or not equal to (`!=`) another operand. It returns a bool value true if the relation is true and false if the relation is false.

```
equality_expression
    : relational_expression
    | equality_expression EQ_OP relational_expression
    | equality_expression NE_OP relational_expression
    ;
```

Logical AND Operator `&&` operator returns a bool value true if both operands evaluate to true or a bool equivalent. Otherwise, it returns false.

```
logical_and_expression
    : equality_expression
    | logical_and_expression AND_OP equality_expression
    ;
```

Logical OR Operator `||` operator returns a bool value true if either of the two operands evaluate to true or a bool equivalent. Otherwise, it returns false.

```
logical_or_expression
    : logical_and_expression
    | logical_or_expression OR_OP logical_and_expression
    ;
```

Conditional Operator Ternary conditional operator `? :` can be thought of as shorthand for an if-else statement. If the first operand evaluate to true, assign the value of the second operand to result. Otherwise, assign the value of the third operand to result.

```
conditional_expression
    : logical_or_expression
    | logical_or_expression '?' conditional_expression ':'
    conditional_expression
    ;
```

Assignment Expression The left operand must be a postfix expression and the left operand must be a conditional expression which has return value. It assign the value on its right to the operand on its left.

```
assignment_expression
    : postfix_expression '=' conditional_expression
    ;
```

Expression An expression is a construct made up of variables operators and function invocations. There are six kind of expressions.

```
expression
    : conditional_expression
    | assignment_expression
    | DISPLAY IDENTIFIER
    | HIDE IDENTIFIER
    | SWAP '(' IDENTIFIER ',' conditional_expression ',' conditional_expression ')'
    | PRINT print_list
    ;

print_list
    : conditional_expression
    | print_list , conditional_expression
    ;
```

Conditional expression and assignment expression are introduced above.

Display and hide expressions can decide whether to show a variable. If the variable is defined to be displayed, the following execution about it inside the `<begin_display>` and `<end_desplay>` block will be displayed. If it is hide, it will not be shown. The display status of variables can be changed in the program.

Swap expression swaps two elements of one array. The first operand indicates the identifier of the array. The next two operands indicate the index of the two elements. The SWAP key word can help the language to generate specific animation of swapping elements. Users can swap the elements by using their own code, but they cannot get the animation by doing that.

Print expression prints information to the console for debugging. The `print_list` consists of a set of conditional expression separated by commas. The value of the conditional expression will be joined and printed.

3.5.2 Declarations

Variables should be declared inside a function. A example of declaration can be:

```
display int val = 5;
```

Variable declarations are composed of three components:

- Zero or more modifiers, such as display and hide
- The fields type
- The fields name and initializing value

Modifier The modifiers here indicates whether this variable can be displayed inside the `<begin_display>` `<end_display>` block. It is an attribute of every variable. If you does not specify the modifier when declaring, AVL will set hide as the default for it.

Types All variables must have a type. You can use primitive types such as int, char, index. Or you can use reference types such as String.

Variable Names and Initializing Value Variables name should start with a letter and then followed by zero or more number, letter or underscore. This is the same as in C. You can choose not to initialize a variable immediately in declaration.

Return Value Although declaration in AVL is C styled. It does not return a value. Here is the grammar of declaration in AVL:

```
declaration
    : type_specifier init_declarator_list
    | DISPLAY type_specifier init_declarator_list
    | HIDE type_specifier init_declarator_list
    ;

init_declarator
    : declarator
    | declarator '=' initializer
    ;
```

Similar to C, the declarator(left value) can be an identifier or an array. The initializer(right value) can be an expression or a list of value inside a brace. An example is

```
int array[] = { 1 , 2 , 3 }.
```

Grammars for declarator and initializer are as follow:

```
declarator
    : IDENTIFIER
    | IDENTIFIER '[' conditional_expression ']'
    | IDENTIFIER '[' ']'
```

```
    ;

initializer
    : conditional_expression
    | '{' initializer_list '}'
    ;

initializer_list
    : conditional_expression
    | initializer_list ',' conditional_expression
    ;
```

Note that here we does not support 2-dimensional array in AVL.

3.5.3 Statements

A statement forms a complete unit of execution. There are five kind of statements which are expression statement, declaration statement, display statement, control flow statement and compound statement. The first four kind of statements can be nested in compound statement.

```
statement
    : compound_statement
    | expression_statement
    | declaration_statement
    | display_statement
    | selection_statement
    | iteration_statement
    | jump_statement
    ;
```

Expression Statement It is the most basic statement, which is an expression terminated by a semicolon or a single semicolon.

```
expression_statement
    : expression ';'
    ;
```

Declaration Statement It is used to declare variables, which is formed by a declaration and a semicolon.

```
declaration_statement
    : declaration ';'
    ;
```

Display Statement It forms a block of statements between keywords `<begin_display>` and `<end_display>`. The execution of the statement list, which is a group of statements, in the block will be displayed to the user.

```
display_statement
    : BEGIN_DISPLAY statement_list END_DISPLAY
    ;

statement_list
    : statement
    | statement_list statement
    ;
```

Control Flow Statement

Selection Statement The if statement is the most basic control flow statements. It tells the program to execute the certain section of statement only if the expression is true.

The if-else statement evaluates the expression and executes the first statement if it is true. Otherwise it executes the second statement. The dangling else ambiguity is resolved by binding the else with the nearest if.

```
selection_statement
    : IF '(' conditional_expression ')' statement
    | IF '(' conditional_expression ')' statement ELSE statement
    ;
```

Iteration Statement The while statement evaluates expression and executes the statements if it is true. It continues testing the expression and executing its statements until the expression evaluates to false.

The do-while statement evaluates its expression at the bottom of the loop instead of the top. Therefore, the statements within the do block are always executed at least once. In the for statement, the first operand in the parentheses is executed once as the loop begins, which can be an expression or a declaration. The second operand is termination expression, that when it evaluates to false the loop terminates. The third operand is invoked after each iteration of the loop.

```
iteration_statement
    : WHILE '(' conditional_expression ')' statement
    | DO statement WHILE '(' conditional_expression ')' ';'
    | FOR '(' expression ';' conditional_expression ';' ')' statement
    | FOR '(' expression ';' conditional_expression ';' expression ')' statement
    | FOR '(' declaration ';' conditional_expression ';' ')' statement
    | FOR '(' declaration ';' conditional_expression ';' expression ')' statement
    ;
```

Jump Statement The continue statement skips the current iteration to the end of the innermost loops body and evaluates the expression that controls the loop.

The break statement terminates the innermost control flow statement. The return statement exits from the current function and returns to where the function was invoked. It can return a value or nothing.

```
jump_statement
    : CONTINUE ';'
    | BREAK ';'
    | RETURN ';'
    | RETURN conditional_expression ';'
    ;
```

Compound Statement It is a group of zero or more statements between balanced braces and can be used for function definitions and control flow statements.

```
compound_statement
    : '{' '}'
    | '{' statement_list '}'
    ;
```

3.5.4 Function Definition

A function consist of the return type, function name, parameter list surrounded by parentheses and the body between braces. The parameter list can be empty. Parameters of array type are passed in by reference, and others are passed in by value.

```
function_definition
    : type_specifier  IDENTIFIER '(' parameter_list ')' compound_statement
    | type_specifier  IDENTIFIER '('  ')' compound_statement
    ;
```

The parameter list consists of a set of type and declarator pairs which are separated by commas.

```
parameter_list
    : parameter_declaration
    | parameter_list ',' parameter_declaration
    ;

parameter_declaration
    : type_specifier declarator
    ;
```

3.5.5 Program

The translation unit consists of multiple function definition. So our language does not support Macro and Global Variable in C. The whole function itself is a translation unit.

```
translation_unit
    : function_definition
    | translation_unit fucntion_definition
    ;

program
    : translation_unit
    ;
```

3.6 Complete Grammar

```
primary_expression
: IDENTIFIER
| CONSTANT
| TRUE
| FALSE
| CHAR_LITERAL
| STRING_LITERAL
| '(' conditional_expression ')'
;

postfix_expression
: primary_expression
| IDENTIFIER '[' conditional_expression ']'
| IDENTIFIER '[' conditional_expression ':' conditional_expression ']'
| IDENTIFIER '(' ')'
| IDENTIFIER '(' argument_expression_list ')'
| postfix_expression INC_OP
| postfix_expression DEC_OP
;

argument_expression_list
: conditional_expression
| argument_expression_list ',' conditional_expression
;

unary_expression
: postfix_expression
```

```
| INC_OP unary_expression
| DEC_OP unary_expression
| unary_operator unary_expression
| LEN '(' unary_expression ')'
;

unary_operator
: '+'
| '-'
| '!'
;

multiplicative_expression
: unary_expression
| multiplicative_expression '*' unary_expression
| multiplicative_expression '/' unary_expression
| multiplicative_expression '%' unary_expression
;

additive_expression
: multiplicative_expression
| additive_expression '+' multiplicative_expression
| additive_expression '-' multiplicative_expression
;

relational_expression
: additive_expression
| relational_expression '<' additive_expression
| relational_expression '>' additive_expression
| relational_expression LE_OP additive_expression
| relational_expression GE_OP additive_expression
;

equality_expression
: relational_expression
| equality_expression EQ_OP relational_expression
| equality_expression NE_OP relational_expression
;

logical_and_expression
: equality_expression
```

```
| logical_and_expression AND_OP equality_expression
;

logical_or_expression
: logical_and_expression
| logical_or_expression OR_OP logical_and_expression
;

conditional_expression
: logical_or_expression
| logical_or_expression '?' conditional_expression ':' conditional_expression
;

assignment_expression
: postfix_expression '=' conditional_expression
;

type_specifier
: VOID
| CHAR
| INT
| STRING
| INDEX
| BOOL
;

expression
: conditional_expression
| assignment_expression
| DISPLAY IDENTIFIER
| HIDE IDENTIFIER
| SWAP '(' IDENTIFIER ',' conditional_expression ',' conditional_expression ')'
| PRINT print_list
;

print_list
: conditional_expression
| print_list ',' conditional_expression
;

declaration
```

```
: type_specifier init_declarator
| DISPLAY type_specifier init_declarator
| HIDE type_specifier init_declarator
;

init_declarator
: declarator
| declarator '=' initializer
;

declarator
: IDENTIFIER
| IDENTIFIER '[' conditional_expression ']'
| IDENTIFIER '[' ']'
;

initializer
: conditional_expression
| '{' initializer_list '}'
;

initializer_list
: conditional_expression
| initializer_list ',' conditional_expression
;

statement
: compound_statement
| expression_statement
| declaration_statement
| display_statement
| selection_statement
| iteration_statement
| jump_statement
| ';'
;

expression_statement
: expression ';'
;
```

```
declaration_statement
: declaration ';'
;

compound_statement
: '{' '}'
| '{' statement_list '}'
;

statement_list
: statement
| statement_list statement
;

display_statement
: BEGIN_DISPLAY statement_list END_DISPLAY
;

selection_statement
: IF '(' conditional_expression ')' statement
| IF '(' conditional_expression ')' statement ELSE statement
;

iteration_statement
: WHILE '(' conditional_expression ')' statement
| DO statement WHILE '(' conditional_expression ')' ';'
| FOR '(' expression ';' conditional_expression ';' ')'statement
| FOR '(' expression ';' conditional_expression ';' expression ')' statement
| FOR '(' declaration ';' conditional_expression ';' ')' statement
| FOR '(' declaration ';' conditional_expression ';' expression ')' statement
;

jump_statement
: CONTINUE ';'
| BREAK ';'
| RETURN ';'
| RETURN conditional_expression ';'
;

translation_unit
: function_definition
```

```
| translation_unit function_definition
;

function_definition
: type_specifier  IDENTIFIER '(' parameter_list ')' compound_statement
| type_specifier  IDENTIFIER '('  ')' compound_statement
;

parameter_list
: parameter_declaration
| parameter_list ',' parameter_declaration
;

parameter_declaration
: type_specifier declarator
;

program
: translation_unit
;
```

4 Project plan

4.1 Management Process

Our team was formed after the first class. During the early process of the project we met two to three times a week to brainstorm ideas about the new language. We use three weeks to discuss what language we want to implement. Finally, we combined several ideas of different team members and AVL came out.

After we decide the language, we met once a week to discuss what we can do for the next step. We separated the project into two part: the front-end and back-end, and they are developed parallelly. The back-end is the library code which is used after AVL code was translated to C++ code. We began early in implementing the back-end, since it need less knowledge about PLT when we started to write code. The front-end process went on according to the what we learned from the course. The workloads are divided to creating grammar, implementing scanner and parser, building abstract syntax tree, implementing code generator and doing semantic check.

We adopt GitHub to help us control the code version. We require each member push their code to the master node after testing the code locally. At least the code can be compiled at that time. Otherwise, the member need to create a branch on Github. It can make sure that the team members work will not be slowed down by others. We assigned the work to each member according to the modularity rule that each one was not working on a same file at one time. That can greatly enhance the efficiency of the whole process. When someone find the bugs which is not managed by him, he can open an issue on Github and everyone can receive the notify email immediately.

For each file, we did small tests directly by the one who was responsible to it. This step can decrease the number of bugs as soon as possible. After we complete the whole project, we tested them together by using the designed set of test cases. We could found some special bugs, and discussed together to fix it.

4.2 Roles and Responsibilities

Team Member	Role	Responsibility
Qianxi Zhang	Project Manager	parser, AST, code generator
Shining Sun	Language Guru	grammar, semantic check, AvlTypes
Yu Zheng	System Architect	AST, AvlTypes, test cases
Qinfan Wu	System Integrator	symbol table, AvlTypes, AvlVisualizer, AvlUtils, runtime environment
Jiuyang Zhao	System Tester	scanner, parser, test cases

Table 4: Roles and responsibilities

4.3 Implementation Style Sheet

- Keep code neat and clean. Use 4 spaces tab to indent.
- Make sure the code can be compiled when push the code to master node in GitHub.

- Push uncompilable code to new branches if needed.
- Push code regularly to let other team members keep track of the newest version.
- When one find a bug which cannot be fixed by himself, notify team members through GitHub.

4.4 Timeline

Date	Milestone
01/22/2014	Team Formation
02/16/2014	Project AVL Confirmed
02/24/2014	GitHub Opened
02/26/2014	Whitepaper Completed
03/09/2014	Build Environment Initialized
03/23/2014	AvlUtils Implemented
03/24/2014	AvlVisualizor Implemented
03/25/2014	Scanner & Parser Implemented
03/26/2014	Language Tutorial & LRM Completed
03/28/2014	AvlInt and AvlArray in AvlType Implemented
04/16/2014	AvlChar in AvlType Implemented
04/17/2014	AvlBool in AvlType Implemented
04/20/2014	AST Implemented
04/20/2014	AvlString in AvlType Implemented
04/26/2014	Code Generator Implemented
05/03/2014	Test Cases Design Completed
05/09/2014	AvlIndex in AvlType Implemented
05/10/2014	Semantic Check Implemented
05/11/2014	Test Finished

Table 5: Timeline

4.5 Project Log

Project log is appended in Appendix.

5 Language evolution

In the original language proposal, we proposed a language that visualizes algorithms for teaching and learning purpose. Actually it was me who initially came up with this idea, and that was how I ended up with this role. Originally we had plenty of interesting and ambitious ideas, including visualize AhoCorasick string matching algorithm. But as the implementation process going, we found that it was not easy to implement all of them because different algorithms requires different data structures, and different data structures requires different methods for vitalization. For example, it is very different to visualize an array than to visualize a finite automaton. Therefore, we decided to start from the most basic structures, array.

Without distracting thoughts, we focus on implementing array. We firstly had a case study on the algorithms that use array as their primary data structures, like sorting algorithms, and tried to extract common features of them. We then concluded that there need to have swap as a build-in operation on arrays. For example, in compare-based sorting algorithms, it is essential to have swap because the basic idea behind these algorithms is to compare two elements and adjust their positions. Therefore, having swap as build-in operator makes it convenient for the programmer to visualize array-based algorithms.

Also, we thought that it is useful to give the programmer the right to decide which part of his program to be visualized. This makes sense because although it is a cool idea to visualize algorithms, it is possible that user may not want to visualize everything they write. For example, as mentioned above, it is possible to visualize sorting algorithms. But in other algorithms, sorting might be a sub-routine and should not be visualized. Therefore, we have defined `<begin_display>` and `<end_display>` to enclose the part that needs to be visualized. Also, for each variable, programmer can also choose to visualize it or not.

Along the process of implementation, we also made some decisions on what not to be included in the language. For example, we decided not to include pointers, because it is way to complicated to implement. As another example, while a struct is very useful in C, we decided to not include that in our language, because there is no way to correctly visualize every user-defined struct.

5.1 Compiler Tools

We decided to use C++ as the target language for our compiler because that is the most familiar language for all of the team member.

Lex and Yacc are used as scanner and parser for our compiler, because it is compatible with C/C++, and there are tons of tutorials, resource and documentations online for them.

5.1.1 Libraries

The main functionality of our language is visualization. Therefore, we used OpenGL, freeglut, glew glm and glfw as they are the most commonly used libraries in computer graphics.

5.2 Consistency

Although there are some aggressive ideas in the white paper, in the LRM, the only structure left is the array. Therefore, we did not had a chance to deviate from the LRM. Besides, we thought it is nice to have our grammar as close to that of C as possible, because that will create a more reasonable learning curve for programmers. Therefore, we largely referred to the C standard as we writing our grammar. This also helps us to stick to our original idea, grammar wise.

6 Translator Architecture

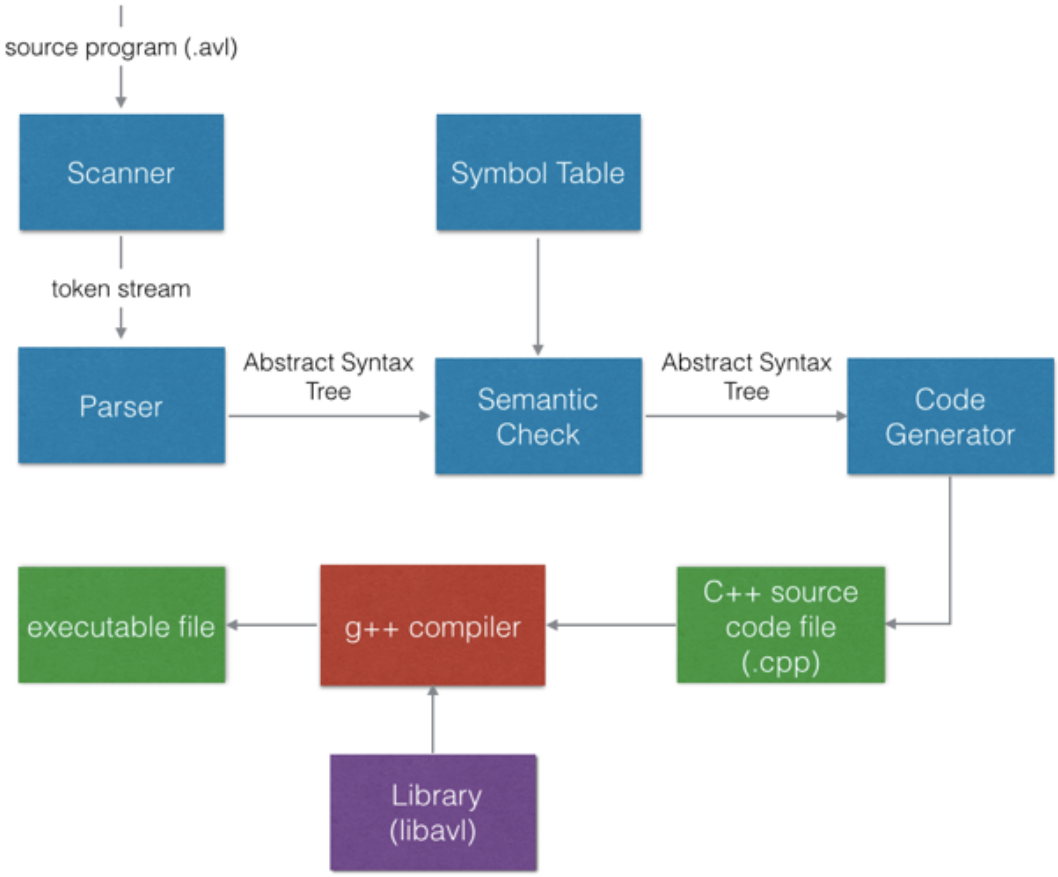

Figure 1: Translator architecture

6.1 Scanner

The string of source code, from .avl file, is passed into Scanner, which continuously groups characters into lexemes, and outputs tokens. Jiuyang Zhao firstly drafted the scanner, and the entire team contribute to the final version.

6.2 Parser

Our grammar is written in the parser. It uses tokens produced by the scanner to create an abstract syntax tree, which depicts the grammatical structure of the token stream. Augmented by the SDD, the parser generated an AST during evaluating token stream. This AST is consisted of nodes, which has type and value. Qianxi Zhang and Jiuyang Zhao deliberated the grammar. Qianxi Zhang and Yu Zheng implemented the SDD. Finally the entire team validated it consistent with the LRM.

6.3 Symbol Table

The entire program has a linked list of symbol tables, one per scope. Each identifier has a corresponding entry in the symbol table. Hash value of the identifier name is used as the key. The symbol table was written Qinfan Wu and refined by the entire team.

6.4 Semantic Check

The semantic analyzer uses the AST and the information in the symbol table to check the source program for semantic consistency with the language definition. In this phase, we do scope checking and validation on semantics. For a running program, there is a stack storing scope-directed symbol tables. To validate a function call, we search it in the symbol table at the bottom of the stack. To validate a variable, we search it from the top of the stack. If no symbol table contains it, an error is reported. Shining Sun implemented this part.

6.5 Code Generator

For the purpose of easily debugging, we separate semantic check from code generator. It takes the AST generated by the parser as input. While traversing the tree, the generator produces C++ code, by parsing each node. It outputs the C++ code to a .cpp file. The generator was initially written by Qianxi Zhang, and the entire team gave feedback and refined to the final version.

6.6 Library (libavl)

The libavl is a shared library, which is used for supporting visualization. It defines different data types and visualizers in order to use OpenGL to do visualization. The translated C++ code is linked with libavl, and then passed to C++ compiler, which generates an executable file. This library architecture was written by Qinfan Wu, Shinig Sun, and Yu Zheng.

All of the above modules are grouped together and run by automake, which was written by our System Integrator Qinfan Wu.

7 Development and Runtime Environment

7.1 Software development environment

The avl compiler was developed under Unix-based environments. Our goal is to make a compiler highly portable on all Unix-based environments, including Linux and Mac OS. In addition, we made full use of Git and GitHub for reliable version control.

Flex is used to create the scanner, which reads the source file and generates a stream of tokens. The parser is created by Bison. Combined with the code generator, we generate c++ target code. The output c++ file is formatted by a tool called astyle and finally compiled by g++.

7.2 Makefile

We use GNU Autotools (autoconf/automake/libtool...) to generate `Makefile`s automatically. Autotools help us create `Makefile`s that are fully GNU standards-compliant and extremely easy to maintain.

For developers, we only need to create the following files to specify how the `Makefile`s will be generated:

- `configure.ac` in the top level directory
- `Makefile.am`s in directories where `Makefile`s are needed.

In order to create Makefiles, all the develop needs to do is to run the following commands:

```
autoreconf --install
./configure
```

Another significant feature provided by Autotools is that running test cases can be as easy as typing a single command:

```
make check
```

As we can see from Fig. 2, the report tells us which of the test cases passed and which of the test cases failed. Each test script would write the output into a log file, so we are able to identify the error message from the corresponding log file.

All the `Makefile`s are listed below.

configure.ac

```
AC_INIT([avl], [0.1], [avl-plt@googlegroups.com], [],
[https://github.com/wqfish/avl])

AM_INIT_AUTOMAKE([-Wall -Werror foreign])

AC_PROG_CC
```

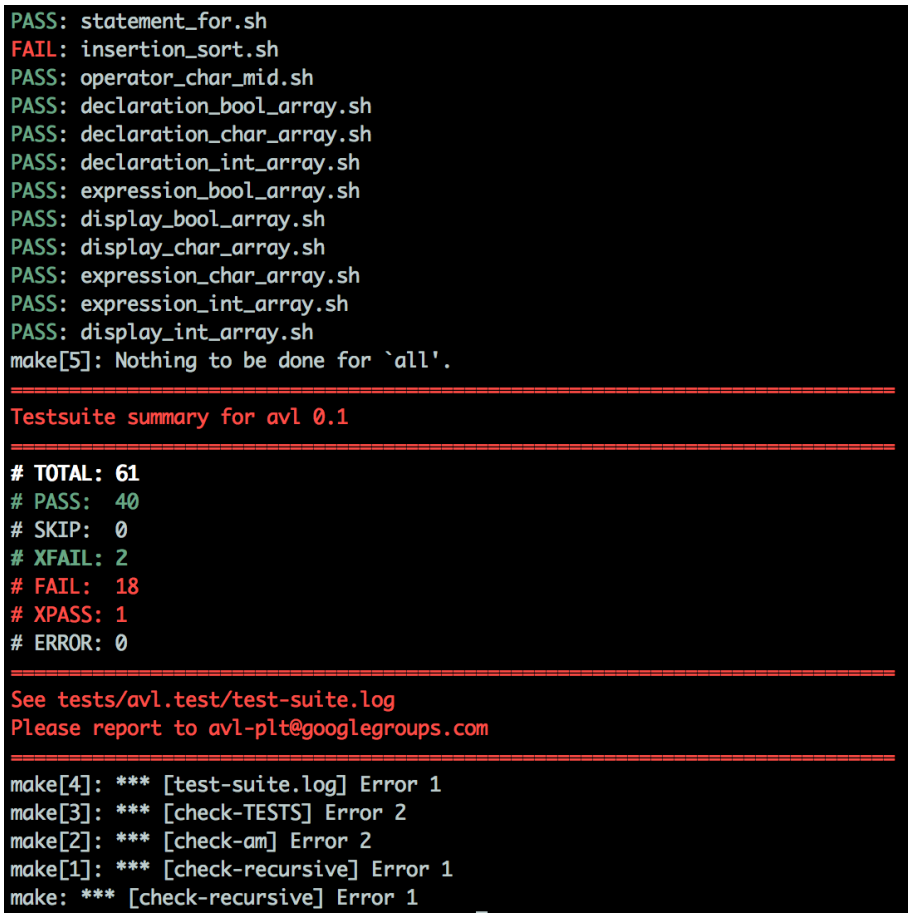

Figure 2: Automated testing

```
AC_PROG_CXX
AC_PROG_LEX
AC_PROG_YACC

AC_CONFIG_MACRO_DIR([m4])
AM_PROG_AR
AC_PROG_LIBTOOL

AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([
         Makefile
         src/Makefile
         lib/Makefile
         sample_code/Makefile
         tests/Makefile
         tests/libavl.test/Makefile
         tests/avl.test/Makefile
         tests/error.test/Makefile
])

AC_OUTPUT
```

Makefile.am

```
ACLOCAL_AMFLAGS = -I m4
SUBDIRS = src lib sample_code tests
```

src/Makefile.am

```
BUILT_SOURCES = parser.h
AM_YFLAGS = -d
bin_PROGRAMS = avl
avl_SOURCES = scanner.l parser.y avl.c syntax_tree.c code_generator.c sym_list.c sym_table.c
st_list.c
AM_CFLAGS = -Werror
```

lib/Makefile.am

```
lib_LTLIBRARIES = libavl.la
libavl_la_SOURCES = AvlVisualizer.cpp
include_HEADERS = AvlVisualizer.h AvlTypes.h AvlUtils.h
AM_CPPFLAGS = -Wall -Wextra -Werror -std=c++11 -I/opt/local/include
```

sample_code/Makefile.am

```
noinst_PROGRAMS = sorting merge_sorted_array
sorting_SOURCES = sorting.cpp
merge_sorted_array_SOURCES = merge_sorted_array.cpp

AM_CPPFLAGS = -Wall -Werror -std=c++11 -I$(top_builddir)/lib -I/opt/local/include
AM_LDFLAGS = -L$(top_builddir)/lib -L/opt/local/lib -lavl -lGL -lglut
```

tests/Makefile.am

```
SUBDIRS = libavl.test avl.test error.test
```

```
tests/libavl.test/Makefile.am
check_PROGRAMS = \
                 avlint_assign1 \
                 avlint_add1 \
                 avlint_add2 \
                 avlint_add3 \
                 avlint_sub1

avlint_assign1_SOURCES = avlint_assign1.cpp
avlint_add1_SOURCES = avlint_add1.cpp
avlint_add2_SOURCES = avlint_add2.cpp
avlint_add3_SOURCES = avlint_add3.cpp
avlint_sub1_SOURCES = avlint_sub1.cpp

AM_CPPFLAGS = -Wall -Werror -std=c++11 -I$(top_builddir)/lib -I/opt/local/include
AM_LDFLAGS = -L$(top_builddir)/lib -L/opt/local/lib -lavl -lglut -lGL

AM_TESTS_ENVIRONMENT = . $(srcdir)/init_test.sh;
AM_TESTS_FD_REDIRECT = 9>&2

TESTS = \
        avlint_assign1.sh \
        avlint_add1.sh \
        avlint_add2.sh \
        avlint_add3.sh \
        avlint_sub1.sh
```

tests/avl.test/Makefile.am

```
AM_TESTS_ENVIRONMENT = . $(srcdir)/init_test.sh;
```

```
AM_TESTS_FD_REDIRECT = 9>&2

TESTS = \
        cast.sh \
        constant.sh \
        declaration_bool.sh \
        declaration_bool_array.sh \
        other_test_cases omitted...
        maxSubarray.sh

XFAIL_TESTS = \
        cast.sh \
        statement_scope.sh \
        empty.sh
```

tests/error.test/Makefile.am

```
AM_TESTS_ENVIRONMENT = . $(srcdir)/init_test.sh;
AM_TESTS_FD_REDIRECT = 9>&2

TESTS = \
        func_malformat.sh \
        func_malparam.sh \
        func_malscope.sh \
        other_test_cases omitted...
        array_malvalue.sh

XFAIL_TESTS = \
        func_malformat.sh \
        func_malparam.sh \
        other_test_cases omitted...
        array_malvalue.sh
```

For users who want to use the compiler, in its simplest scenario, all the user has to do is to unpack the package, then run the following commands:

```
./configure
make
sudo make install
```

According to GNU standards, the avl compiler is installed to `/usr/local/bin`, header files to `/usr/local/include` and libraries to `/usr/local/lib`. Thus, after the installation the user would be able to invoke avl from the command line.

7.3 Runtime Environment

As discussed above, the avl compiler works on all Unix-based systems, including Linux and Mac OS. The latest version of the following tools are required:

- gcc
- astyle
- OpenGL libraries: freeglut glew glm glfw

For Mac, an X window system is also required (https://xquartz.macosforge.org/).
The avl compiler can be invoked as follows:

```
$ avl --help
Usage: avl [-h|-o|-t] file
Options:
  -h --help            Display this information
  -o --output=<file>   Compile and place the executable into <file>
  -t --translate       Translate the source files into c++
                          xxx.avl -> xxx.cpp
Use at most one option at a time.

For bug reporting instructions, please see:
<https://github.com/wqfish/avl>.
```

8 Test plan: Jiuyang Zhao

8.1 Tools

AVL compiler uses GNU Autotools to generate makefiles automatically. Autotools provides the function make check to run our tests cases automatically. By running this command, our makefile will run the test cases sequentially one by another. It give the result into four classes. `#PASS`, `#FAIL`, `#XPASS` and `#XFAIL`. Each of them denotes the meaning that: the correct test cases that passed(good case), the correct test cases but failed(bad case), the incorrect test cases but passed(bad case) and the incorrect test cases that failed(good case).

After running make check, we can get to statistics result of the successful rate of our test cases. After the we run the that command, a log file will be generated for each test case to record the output of stdout and stderr. So once that are unexpected result, we can check the log file and debug easily.

8.2 Relevant and Description

Our test cases can be divided into seven classes, each classes corresponds to a part in AVL grammar. They are:

- constant_*: This is a most simple one. Just test if there are any error when we taking each data types as constant.
- declaration_*: In declaration part we test different way of declaration for data type. For example: adding keyword display, hide or not, declare with initialization value or not, and different way in declare an array.
- expression_*: In expression part we test some operation in expression like swaping an array, or displaying/hiding a variable in half way.
- operator_*: In operator we test the basic operation for every data type, including postfix, unary and binary operator. Because different data type are implement in different class at our back end, and we overload every operator for them. So it is necessary to check these member functions.
- display_*: For each object like AvlInt and AvlArray in our back-end, we use the function render to draw each object and ready to print. We design part of test cases to check whether each kind of data type can be displayed correctly.
- function_*: Here we test for parameter and return value for each data type. Also we test the function of subarray and the displaying when functions are nested.
- statement_*: Here we mainly aim to test for different of control flows like if, for and while. Also we have tested for some basic scope.

8.3 Selected Test Cases

This is a simple test cases for declare an bool array:

```
int main()
{
    bool b6[5];
    display bool b7[5];
    hide bool b8[6];
    bool b9[3] = {true, true, false};

    return 0;
}
```

This is another test cases for for loop taken as looping variable.

```
int main(){
    /* empty, line 4 failed */
    for (int i = 0; i < 10; i++);

    for (int i = 0; i<10; i++) {

    }

    /* regular for */
    for (int i = 0; i < 10; i++) {
        print i;
    }

    /* changing i inside */
    for (int i = 0; i < 10; ) {
        print i;
        i = i + 2;
    }

    /* decleare i outside for */
    int i = 0;
    for (i; i < 10; i++)
        print i;
    for(i; i > 0;)
        print i--;
```

```
    /* testing continue, break */
    for (i; i < 10; i++) {
        if( i == 2)
            continue;
        else if (i == 5)
            break;
        print i;
    }

    /* testing nested for */
    for (i = 0; i < 3; i++) {
        for (int j = 0; j < 2; j++)
            print j;
    }

    /* testing return */
    for (i = 0; i < 10; i++){
        print i;
        if (i == 5)
            return 0;
    }

    return 0;
}
```

8.4 Notable Bugs Encountered

1. We must add `()` when we translate print into `std::cout` because in c++ the precedence of `<<` is higher than some operator we need in AVL like `||`.

2. Because we translate int to our own defined type AvlInt, we must treat the `int` in `int main()..` differently to avoid the main function in c++ returning an `AvlInt`.

3. Our test cases in loop statement find out that we cannot handle empty loop before.

9 Conclusions

9.1 Lessons Learned as a Team

1. One of the most important things is to invent a great language idea. The idea should be interesting enough and can be done in given a semester. Also, the team should be able to estimate the total amount of work before making the plan for such a semester-long project.

2. Meetings regularly to keep every team member up to date with bi-weekly tasks. Keep coding every week improves efficiency and productivity. Each team member should feel a greater sense of responsibility for his/her scope. In addition, each team member should finish his/her task in time.

3. Using version control systems appropriately greatly helps development of a large project.

9.2 Lessons Learned by Each Team Member

9.2.1 Jiuyang Zhao

From this project. I have done the task I never covered before. Testing is not as easy as most people imagine. First, To write a good test code we need to deeply know the target to the program we write. So I take part in designing grammar, and backend function coding to know both front-end and back-end things in a compiler, but I found this knowledge is still far from enough and often I still have trouble to locate the bugs. Second, the concept of unitest is difficult to implement. In this project we use autotool in GNU to generate automatic test-bench. However it is not atomic enough to cope with every corner of our program. Most of test cases I write was to test the every production in grammar, and the different function in our back-end class. However, to test the function like scope and memory leak is still very difficult.

Also designing a good language is also very tough because there are no existing standard and method to designing context free grammar. We have some technique to generate regular expression. But we dont have too much way to deal with CFG. The only technique we have is some basic things like remove left reduction, factoring and disambiguous a grammar. With the powerful modern tools like bison, this problem will be solved automatically, and how to design a grammar still remains a core problem. The lesson we learn from this is trying to use the CFG from existent language such as C and python. There CFG are already well defined so we dont need to do most of task our self to write a concise programming language.

9.2.2 Qianxi Zhang

First of all, the project helped me to have a deeper understanding of the knowledge we learned in the course. As we discussed how to implement each part of the project a little bit earlier than the course, we may think the question by ourselves. After that, we can compare our ideas with the algorithm taught in class which helped a lot for grasp the outstanding points of the algorithm. The most important things I learned from the project is about team working.

1. Creative Power: I really appreciate the creative power of the team, which I would never have by working by myself. The first several ideas about the project were trivial or had already been implemented by other teams in the past years. For example, language to solve the mathematical problems or graph theory problems, language to generate movies or language to teach students. Although each idea seemed simple, but when we combined them together, a novel language AVL came out. I think it is worth to spend a long time to do brainstorming and don't give up when you feel depressed. Discussing and exchanging ideas will get magic result and a great idea is half the battle for a team.

2. Listening and Learning from Others: When implementing our project, we may often have different ideas with each other. Instead of consisting on our own ideas, the better way is to listen to others, and try to understand their core intention of the ideas. Then we can discuss the advantages and disadvantages of the ideas and finally find what is truly best for our project. Moreover, each member have their own talent, and we can learn a lot from others. For example, the rigorous coding style, the knowledge to set up the complex project environment or even basic skills such as how to use VIM efficiently.

3. Coding: For coding part, we learned a lot about version management. We emphasized the importance of uploading code after tested locally or at least can be compiled. Otherwise, others may be unable to debug their new code. Another way to avoid this problem is to let different member manage different file at same time and the interface of each module should be well defined earlier. Moreover, discussing frequently can help us catch up with others steps. Since we manage different parts of the project, if we have different understanding of the newest feature of the language, it may lead to bugs between the modules.

9.2.3 Qinfan Wu

It is relatively hard to building a project from scratch. Also, we need to learn how to work in a team of five, how to communicate with each other and build a big project in parallel. A few things that worth noting are as follows.

1. Have a good design and understanding of the project before coding. Keep modular design and always make sure that you can add more features and make modifications without unnecessary effort.

2. Every time you implement a small feature, you need to do enough testing by creating some test cases, then commit to the git repository. Also, it is hard to create test cases that cover all the cases.

3. Split up the whole project into appropriate modules. First define the interface between all the modules and then implement each functionality.

4. Working in a team with several people is more productive than working alone. Also, having roles assigned helps divide the task into appropriate parts.

5. Automate the building and testing process as much as you can. Almost all of them can be done by Makefiles and bash scripts.

6. All the code must be clearly organized and well commented.

9.2.4 Shining Sun

This is one of the largest project that I have worked on. I had a enjoyable time, also a hard time. Also, this is the largest team I have ever been in. Although suffering from time to time, I did learn a lot from this project.

1. Plan well. One of the lessons that I learned from this project is that always plan ahead of time, especially large projects like this. Planning is good for all aspects. Time wise, a good plan can save us from rushing towards the deadline. Work load wise, a good plan can keep everybody in the team involved, and have a reasonable work load such that nobody can sit back and take everything for granted, and nobody needs to have to much burden on his shoulder.

2. Execution. A good play needs good execution. Even with a perfect plan, if nobody follows it, it still does not make any sense. There are tons of ways to keep the execution of the team. For example, there could be some reward for people finish their work on time, and punishment for those who do not.

3. Communication. I did not realize the importance of communication until I participated in this project. It is so important and not so easy to understand each others idea. We had a lot of arguments just because people did not understand each other. I think everybody agrees that meeting regularly is very important. But I also learned that meeting with an agenda is also very important. At the beginning of each meeting, an agenda should be firstly drafted. This is essential for a meeting to be concise and time-saving.

4. Coding. Coding is of course the main part of the project. With such a large project, I learned that coding in good structure and style is very important. We have written about 7000 lines of code, and if the structure of the code is bad, it is almost impossible to maintain the code. This is also very important for my future career, because in real-world industry, 7000 lines of code is actually not that much. Software nowadays can have millions lines of code. If there is not a good structure and documentation, there is noway to maintain such a massive project.

9.2.5 Yu Zheng

Building a project from scratch was quite an interesting experience. Also designing and implementing a translator was pretty helpful for understanding programming languages. Through this project, I learned how a source program was translated into an executable file. More important, I learned how a team worked together, and show a team should work efficiently and effectively.

1. Start early. Weekly meeting is essential, no matter how unnecessary you think it would be. Get prepared before meeting, and a well organized meeting is quite efficient.

2. Working with talented teammates widened my horizon. They are not only classmates, but also teachers that encouraged me a lot. Working together can be very creative and productive.

3. Project control tools, e.g. Automake, are life-saver. Generating projects automatically and programmatically guarantees the development process. Of course, a version control tool e.g. GibHub, is crucial, especially when the team size is larger than 3.

4. Nothing would be sweeter and cooler than solving an issue.

5. You will never regret to take this course. This project is full of fun and really educative.

9.3 Advice for Future Teams

1. Invent a good language idea.

2. Use an appropriate version control system. Git/GitHub is a good choice.

3. Use Autotools to generate and maintain Makefiles. Automake has good support for Lex/Yacc. Autotools are also helpful for running test cases automatically.

4. Given the time, dont create too complicated grammars, which makes semantic check much harder. Concentrate on the core functionality.

5. Follow good coding style. Set -Wall -Wextra -Werror for gcc and treat all warnings as errors.

6. Good communication between all team members is important.

9.4 Suggestions for the instructor on what topics to keep, drop, or add in future courses

1. We want to learn more detailed information about semantic check, especially type checking, scope checking, and symbol table.

2. More detailed information about SDT is desired.

3. We are curious about back end of the compiler. More back end topics are desired.

4. We want to know more technique on how to design a context free grammar we want.

10 Appendix

10.1 Source Code

file: avl/configure.ac

```
AC_INIT([avl], [0.1], [avl-plt@googlegroups.com], [],
    [https://github.com/wqfish/avl])

AM_INIT_AUTOMAKE([-Wall -Werror foreign])

AC_PROG_CC
AC_PROG_CXX
AC_PROG_LEX
AC_PROG_YACC

AC_CONFIG_MACRO_DIR([m4])
AM_PROG_AR
AC_PROG_LIBTOOL

AC_CONFIG_HEADERS([config.h])
AC_CONFIG_FILES([
         Makefile
         src/Makefile
         lib/Makefile
         sample_code/Makefile
         tests/Makefile
         tests/libavl.test/Makefile
         tests/avl.test/Makefile
         tests/error.test/Makefile
         ])

AC_OUTPUT
```

file: avl/lib/AvlTypes.h

```
#ifndef AVL_TYPES_H_
#define AVL_TYPES_H_

#include <GL/freeglut.h>
```

```
#include <initializer_list>
#include <vector>
#include <string>
#include <memory>
#include <thread>
#include <mutex>
#include <iostream>
#include "AvlUtils.h"
#include "AvlVisualizer.h"

const float DEFAULT_DELAY = 0.5;

const int FPS = 20;

typedef unsigned int AvlColor;
const AvlColor AVL_RED = 0xFF0000;
const AvlColor AVL_GREEN = 0x00FF00;
const AvlColor AVL_WHIGHT = 0xFFFFFF;
const AvlColor AVL_COLOR_HIGHLIGHT = AVL_GREEN;
const AvlColor AVL_COLOR_DEFAULT = AVL_RED;
const AvlColor AVL_AUXILIARY_COLOR = AVL_WHIGHT;

class AvlFont
{
    public:
        // constructor
        AvlFont(void *font = GLUT_BITMAP_9_BY_15)
        {
            _font = font;
            if (font == GLUT_BITMAP_9_BY_15) {
                _width = 9;
                _height = 15;
            }
            else {
                _width = 0;
                _height = 0;
            }
        }

        // assignment operator
        const AvlFont& operator=(void *font)
```

```
        {
            _font = font;
            if (font == GLUT_BITMAP_9_BY_15) {
                _width = 9;
                _height = 15;
            }
            else {
                _width = 0;
                _height = 0;
            }

            return *this;
        }

        // getter functions
        GLfloat width() const { return _width; }
        GLfloat height() const { return _height; }
        void *font() const { return _font; }

    private:
        void *_font;
        GLfloat _width;
        GLfloat _height;
}; // end of AvlFont

class AvlObject
{
    public:
        // constructor
        AvlObject(GLfloat x = 0, GLfloat y = 0, const AvlFont &font = GLUT_BITMAP_9_BY_15)
        {
            _x = x;
            _y = y;
            _width = font.width();
            _height = font.height();
            _font = font;
            _color = AVL_COLOR_DEFAULT;

            _vi = NULL;
        }
```

```
// copy constructor
AvlObject(const AvlObject &obj)
{
    _x = obj._x;
    _y = obj._y;
    _width = obj._width;
    _height = obj._height;
    _font = obj._font;
    _color = obj._color;

    _vi = obj._vi;
}

// assignment operator
const AvlObject& operator=(const AvlObject &obj)
{
    _x = obj._x;
    _y = obj._y;
    _width = obj._width;
    _height = obj._height;
    _font = obj._font;
    _color = obj._color;

    _vi = obj._vi;

    return *this;
}

// destructor (virtual)
virtual ~AvlObject()
{
    _vi->delObject(name());
}

// getter functions
GLfloat x() const { return _x; }
GLfloat y() const { return _y; }
GLfloat width() const { return _width; }
GLfloat height() const { return _height; }
AvlFont font() const { return _font; }
AvlColor color() const { return _color; }
```

```
        std::string name() const { return _name; }

        // setter functions
        void set_x(GLfloat x) { _x = x; }
        void set_y(GLfloat y) { _y = y; }
        void set_font(const AvlFont &font) { _font = font; }
        void set_color(AvlColor color) { _color = color; }
        void set_name(const std::string& name) { _name = name; }

        virtual void render() = 0;

        // locks
        void lock() { mtx.lock(); }
        bool try_lock() { return mtx.try_lock(); }
        void unlock() { mtx.unlock(); }

        // set visualizer
        void setVisualizer(AvlVisualizer *vi) { _vi = vi; }

    protected:
        // position setter
        void set_width(GLfloat width) { _width = width; }
        void set_height(GLfloat height) { _height = height; }

        bool isDisplaying() const { return _vi->getLevel() > 0; }

    private:
        GLfloat _x;
        GLfloat _y;
        GLfloat _width;
        GLfloat _height;
        AvlFont _font;
        AvlColor _color;

        std::mutex mtx;

        AvlVisualizer *_vi;

        std::string _name;
}; // end of AvlObject
```

```
inline void moveObject(std::shared_ptr<AvlObject> obj, GLfloat x, GLfloat y, float seconds)
{
    int steps = FPS * seconds;
    GLfloat delta_x = x / steps;
    GLfloat delta_y = y / steps;

    for (int i = 0; i < steps; i++) {
        obj->lock();
        obj->set_x(obj->x() + delta_x);
        obj->set_y(obj->y() + delta_y);
        obj->unlock();
        avlSleep(1.0 / FPS);
    }
}

class AvlInt : public AvlObject
{
    public:
        // constructor
        AvlInt(int v = 0, GLfloat x = 0, GLfloat y = 0, void *font = GLUT_BITMAP_9_BY_15)
            : AvlObject(x, y, font)
        {
            value = v;
            update();
        }

        // destructor
        virtual ~AvlInt() {}

        // assignment operator
        const AvlInt& operator=(int v) { value = v; update(); return *this; }

        // << operator
        friend std::ostream& operator<<(std::ostream &os, const AvlInt &v)
        {
            os << v.value;
            return os;
        }

        // unary plus and minus
        const AvlInt& operator+() const { return *this; }
```

```
const AvlInt operator-() const
{
    AvlInt ret = *this;
    ret.value = -value;
    ret.update();
    return ret;
}

// pre-increment and post-increment
const AvlInt& operator++() { value++; update(); return *this; }
const AvlInt operator++(int)
{
    AvlInt ret = *this;
    value++;
    update();
    return ret;
}

// pre-decrement and post-decrement
const AvlInt& operator--() { value--; update(); return *this; }
const AvlInt operator--(int)
{
    AvlInt ret = *this;
    value--;
    update();
    return ret;
}

// binary plus
const AvlInt operator+(int v) const
{
    AvlInt ret = *this;
    ret.value += v;
    ret.update();
    return ret;
}

// binary plus
const AvlInt operator+(const AvlInt &v) const { return *this + v.value; }
friend const AvlInt operator+(int v1, const AvlInt &v2) { return v2 + v1; }
```

```
// compound plus
const AvlInt& operator+=(int v) { value += v; update(); return *this; }
const AvlInt& operator+=(const AvlInt &v) { return *this += v.value; }


// binary minus
const AvlInt operator-(int v) const
{
    AvlInt ret = *this;
    ret.value -= v;
    ret.update();
    return ret;
}
const AvlInt operator-(const AvlInt &v) const { return *this - v.value; }
friend const AvlInt operator-(int v1, const AvlInt &v2) { return -v2 + v1; }

// compound minus
const AvlInt& operator-=(int v) { value -= v; update(); return *this; }
const AvlInt& operator-=(const AvlInt &v) { return *this -= v.value; }


// multiplication
const AvlInt operator*(int v) const
{
    AvlInt ret = *this;
    ret.value *= v;
    ret.update();
    return ret;
}
const AvlInt operator*(const AvlInt &v) const { return *this * v.value; }
friend const AvlInt operator*(int v1, const AvlInt &v2) { return v2 * v1; }

// compound multiplication
const AvlInt& operator*=(int v) { value *= v; update(); return *this; }
const AvlInt& operator*=(const AvlInt &v) { return *this *= v.value; }

// devision
const AvlInt operator/(int v) const
{
    AvlInt ret = *this;
    ret.value /= v;
```

```
    ret.update();
    return ret;
}
const AvlInt operator/(const AvlInt &v) const { return *this / v.value; }
friend const AvlInt operator/(int v1, const AvlInt &v2)
{
    AvlInt ret = v2;
    ret.value = v1 / v2.value;
    ret.update();
    return ret;
}

// compound division
const AvlInt& operator/=(int v) { value /= v; update(); return *this; }
const AvlInt& operator/=(const AvlInt &v) { return *this /= v.value; }

// binary mod
const AvlInt operator%(int v) const
{

    AvlInt ret = *this;
    ret.value %= v;
    ret.update();
    return ret;
}
const AvlInt operator%(const AvlInt &v) const { return *this % v.value; }
friend const AvlInt operator%(int v1, const AvlInt &v2)
{
    AvlInt ret = v2;
    ret.value = v1 % v2.value;
    ret.update();
    return ret;
}

// comparison operators
bool operator <(int v) const { return value < v; }
bool operator <(const AvlInt &v) const { return value < v.value; }
friend bool operator<(int v1, const AvlInt v2) { return v1 < v2.value; }

bool operator <=(int v) const { return value <= v; }
bool operator <=(const AvlInt &v) const { return value <= v.value; }
```

```
friend bool operator<=(int v1, const AvlInt v2) { return v1 <= v2.value; }

bool operator >(int v) const { return value > v; }
bool operator >(const AvlInt &v) const { return value > v.value; }
friend bool operator>(int v1, const AvlInt v2) { return v1 > v2.value; }

bool operator >=(int v) const { return value >= v; }
bool operator >=(const AvlInt &v) const { return value >= v.value; }
friend bool operator>=(int v1, const AvlInt v2) { return v1 >= v2.value; }

bool operator ==(int v) const { return value == v; }
bool operator ==(const AvlInt &v) const { return value == v.value; }
friend bool operator==(int v1, const AvlInt v2) { return v1 == v2.value; }

bool operator !=(int v) const { return value != v; }
bool operator !=(const AvlInt &v) const { return value != v.value; }
friend bool operator!=(int v1, const AvlInt v2) { return v1 != v2.value; }

// value getter
int val() const { return value; }

// render function
virtual void render()
{
    unsigned int red = color() / 0x10000;
    unsigned int green = color() % 0x10000 / 0x100;
    unsigned int blue = color() % 0x100;
    glColor4f(red / 255.0, green / 255.0, blue / 255.0, 0.0);

    glRasterPos2f(x(), y());
    glutBitmapString(font().font(), (const unsigned char *)(std::to_string(value).c_st
}

void highlight() {
    set_color(AVL_COLOR_HIGHLIGHT);
}
void lowlight() {
    set_color(AVL_COLOR_DEFAULT);
}

const AvlInt& assign(const AvlInt& v)
```

```
        {
            if (!isDisplaying()) {
                *this = v;
                return *this;
            }

            avlSleep(DEFAULT_DELAY);
            *this = v;
            avlSleep(DEFAULT_DELAY);

            return *this;
        }

    private:
        void update() { set_width(numOfDigit() * font().width()); }

        int numOfDigit() const
        {
            if (value == 0)
                return 1;

            int num = value >= 0 ? 0 : 1;
            int k = value;

            while (k != 0) {
                k /= 10;
                num++;
            }

            return num;
        }

        int value;
}; // end of AvlInt

class AvlChar: public AvlObject
{
    public:
        // constructor
        AvlChar (char v = 0, GLfloat x = 0, GLfloat y = 0, void *font = GLUT_BITMAP_9_BY_15)
            : AvlObject(x, y, font)
```

```
{
    value = v;
}

// destructor
virtual ~AvlChar() {}

// assignment operator
const AvlChar & operator=(char v) { value = v; return *this; }

// << operator
friend std::ostream& operator<<(std::ostream &os, const AvlChar &v)
{
    os << v.value;
    return os;
}

// unary plus and minus
const AvlChar & operator+() const { return *this; }
const AvlChar operator-() const
{
    AvlChar ret = *this;
    ret.value = -value;
    return ret;
}

// pre-increment and post-increment
const AvlChar & operator++() { value++; return *this; }
const AvlChar operator++(int)
{
    AvlChar ret = *this;
    value++;
    return ret;
}

// pre-decrement and post-decrement
const AvlChar & operator--() { value--; return *this; }
const AvlChar operator--(int)
{
    AvlChar ret = *this;
    value--;
```

```
    return ret;
}

// binary plus
const AvlChar operator+(int v) const
{
    AvlChar ret = *this;
    ret.value += v;
    return ret;
}

const AvlChar operator+(char v) const
{
    AvlChar ret = *this;
    ret.value += v;
    return ret;
}

// binary plus
const AvlChar operator+(const AvlInt &v) const { return *this + v.val(); }
const AvlChar operator+(const AvlChar &v) const { return *this +  v.val(); }
friend const AvlChar operator+(int v1, const AvlChar &v2) { return v2 + v1; }
friend const AvlChar operator+(char v1, const AvlChar &v2) { return v2 + v1; }
friend const AvlChar operator+(AvlInt v1, const AvlChar &v2) { return v2 + v1.val(); }
friend const AvlChar operator+(AvlChar v1, const AvlChar &v2) { return v2 + v1.val(); }

// compound plus
const AvlChar & operator+=(int v) { value += v; return *this; }
const AvlChar & operator+=(char v) { value += v; return *this; }
const AvlChar & operator+=(const AvlInt &v) { return *this += v.val(); }
const AvlChar & operator+=(const AvlChar &v) { return *this += v.val(); }


// binary minus
const AvlChar operator-(int v) const
{
    AvlChar ret = *this;
    ret.value -= v;
    return ret;
}
```

```
const AvlChar operator-(char v) const
{
    AvlChar ret = *this;
    ret.value -= v;
    return ret;
}

const AvlChar operator-(const AvlInt &v) const { return *this - v.val(); }
const AvlChar operator-(const AvlChar &v) const { return *this - v.val(); }
friend const AvlChar operator-(int v1, const AvlChar &v2) { return -v2 + v1; }
friend const AvlChar operator-(char v1, const AvlChar &v2) { return -v2 + v1; }
friend const AvlChar operator-(AvlInt v1, const AvlChar &v2) { return -v2 + v1.val(); }
friend const AvlChar operator-(AvlChar v1, const AvlChar &v2) { return -v2 + v1.val();

// compound minus
const AvlChar & operator-=(int v) { value -= v; return *this; }
const AvlChar & operator-=(char v) { value -= v; return *this; }
const AvlChar & operator-=(const AvlInt &v) { return *this -= v.val(); }
const AvlChar & operator-=(const AvlChar &v) { return *this -= v.val(); }


// multiplication
const AvlChar operator*(int v) const
{
    AvlChar ret = *this;
    ret.value *= v;
    return ret;
}

const AvlChar operator*(char v) const
{
    AvlChar ret = *this;
    ret.value *= v;
    return ret;
}

const AvlChar operator*(const AvlInt &v) const { return *this * v.val(); }
const AvlChar operator*(const AvlChar &v) const { return *this * v.val(); }
friend const AvlChar operator*(int v1, const AvlChar &v2) { return v2 * v1; }
friend const AvlChar operator*(char v1, const AvlChar &v2) { return v2 * v1; }
friend const AvlChar operator*(AvlInt v1, const AvlChar &v2) { return v2 * v1.val(); }
```

```
friend const AvlChar operator*(AvlChar v1, const AvlChar &v2) { return v2 * v1.val(); }

// compound multiplication
const AvlChar & operator*=(int v) { value *= v; return *this; }
const AvlChar & operator*=(char v) { value *= v; return *this; }
const AvlChar & operator*=(const AvlInt &v) { return *this *= v.val(); }
const AvlChar & operator*=(const AvlChar &v) { return *this *= v.val(); }

// division
const AvlChar operator/(int v) const
{
    AvlChar ret = *this;
    ret.value /= v;
    return ret;
}

const AvlChar operator/(char v) const
{
    AvlChar ret = *this;
    ret.value /= v;
    return ret;
}

const AvlChar operator/(const AvlInt &v) const { return *this / v.val(); }
const AvlChar operator/(const AvlChar &v) const { return *this / v.val(); }
friend const AvlChar operator/(int v1, const AvlChar &v2) { return v2 / v1; }
friend const AvlChar operator/(char v1, const AvlChar &v2) { return v2 / v1; }
friend const AvlChar operator/(AvlInt v1, const AvlChar &v2) { return v2 / v1.val(); }
friend const AvlChar operator/(AvlChar v1, const AvlChar &v2) { return v2 / v1.val(); }

// compound division
const AvlChar & operator/=(int v) { value /= v; return *this; }
const AvlChar & operator/=(char v) { value /= v; return *this; }
const AvlChar & operator/=(const AvlInt &v) { return *this /= v.val(); }
const AvlChar & operator/=(const AvlChar &v) { return *this /= v.val(); }

// comparison operators
bool operator <(int v) const { return value < v; }
bool operator <(char v) const { return value < v; }
bool operator <(const AvlInt &v) const { return value < v.val(); }
bool operator <(const AvlChar &v) const { return value < v.value; }
```

```
friend bool operator<(int v1, const AvlChar v2) { return v1 < v2.value; }
friend bool operator<(AvlInt v1, const AvlChar v2) { return v1.val() < v2.value; }

bool operator <=(int v) const { return value <= v; }
bool operator <=(char v) const { return value <= v; }
bool operator <=(const AvlInt &v) const { return value <= v.val(); }
bool operator <=(const AvlChar &v) const { return value <= v.val(); }
friend bool operator<=(int v1, const AvlChar v2) { return v1 <= v2.value; }
friend bool operator<=(AvlInt v1, const AvlChar v2) { return v1.val() <= v2.value; }

bool operator >(int v) const { return value > v; }
bool operator >(char v) const { return value > v; }
bool operator >(const AvlInt &v) const { return value > v.val(); }
bool operator >(const AvlChar &v) const { return value > v.value; }
friend bool operator>(int v1, const AvlChar v2) { return v1 > v2.value; }
friend bool operator>(AvlInt v1, const AvlChar v2) { return v1.val() > v2.value; }

bool operator >=(int v) const { return value >= v; }
bool operator >=(char v) const { return value >= v; }
bool operator >=(const AvlInt &v) const { return value >= v.val(); }
bool operator >=(const AvlChar &v) const { return value >= v.value; }
friend bool operator>=(int v1, const AvlChar v2) { return v1 >= v2.value; }
friend bool operator>=(AvlInt v1, const AvlChar v2) { return v1.val() >= v2.value; }

bool operator ==(int v) const { return value == v; }
bool operator ==(char v) const { return value == v; }
bool operator ==(const AvlChar &v) const { return value == v.value; }
bool operator ==(const AvlInt &v) const { return value == v.val(); }
friend bool operator==(char v1, const AvlChar v2) { return v1 == v2.value; }
friend bool operator==(AvlInt v1, const AvlChar v2) { return v1.val() == v2.value; }

bool operator !=(int v) const { return value != v; }
bool operator !=(char v) const { return value != v; }
bool operator !=(const AvlInt &v) const { return value != v.val(); }
bool operator !=(const AvlChar &v) const { return value != v.value; }
friend bool operator!=(int v1, const AvlChar v2) { return v1 != v2.value; }
friend bool operator!=(AvlInt v1, const AvlChar v2) { return v1.val() != v2.value; }


// value getter
char val() const { return value; }
```

```
        // render function
        virtual void render()
        {
            unsigned int red = color() / 0x10000;
            unsigned int green = color() % 0x10000 / 0x100;
            unsigned int blue = color() % 0x100;
            glColor4f(red / 255.0, green / 255.0, blue / 255.0, 0.0);

            glRasterPos2f(x(), y());
            glutBitmapString(font().font(), (const unsigned char *)std::string(1, value).c_str
        }

        void highlight() {
            set_color(AVL_COLOR_HIGHLIGHT);
        }
        void lowlight() {
            set_color(AVL_COLOR_DEFAULT);
        }

    private:
        char value;
}; // end of AvlChar

class array_entry {
public:
    AvlObject* obj;
    GLfloat x;
    GLfloat y;
};

class AvlIndex : public AvlObject {
public:
    // constructor
    AvlIndex (int v = 0, GLfloat x = 0, GLfloat y = 0, void *font = GLUT_BITMAP_9_BY_15) : Avl
        this->value = v;
    }

    // constructor
    AvlIndex (const AvlInt &v, GLfloat x = 0, GLfloat y = 0, void *font = GLUT_BITMAP_9_BY_15)
        this->value = v.val();
```

```
}

// << operator
friend std::ostream& operator<<(std::ostream &os, const AvlIndex &v)
{
    os << v.value;
    return os;
}

// assignment operator
const AvlIndex& operator= (const AvlIndex& that) {
    this->value = that.value;
    return *this;
}

// assignment operator for integer type
const AvlIndex& operator= (int a) {
    this->value = a;
    return *this;
}

// assignment operator for AvlInt type
const AvlIndex& operator= (const AvlInt &a) {
    this->value = a.val();
    return *this;
}

// unary plus and minus
const AvlIndex& operator+() const { return *this; }
const AvlIndex operator-() const
{
    AvlIndex ret = *this;
    ret.value = -value;
    ret.update();
    return ret;
}

// pre-increment and post-increment
const AvlIndex& operator++() { value++; update(); return *this; }
const AvlIndex operator++(int)
{
```

```
    AvlIndex ret = *this;
    value++;
    update();
    return ret;
}

// pre-decrement and post-decrement
const AvlIndex& operator--() { value--; update(); return *this; }
const AvlIndex operator--(int)
{
    AvlIndex ret = *this;
    value--;
    update();
    return ret;
}

// binary plus
const AvlIndex operator+(size_t v) const
{
    AvlIndex ret = *this;
    ret.value += v;
    ret.update();
    return ret;
}

const AvlIndex operator+(int v) const
{
    AvlIndex ret = *this;
    ret.value += v;
    ret.update();
    return ret;
}
const AvlIndex operator+(char v) const
{
    AvlIndex ret = *this;
    ret.value += (int) v;
    ret.update();
    return ret;
}

// binary plus
```

```
const AvlIndex operator+(const AvlInt &v) const { return *this + v.val(); }
const AvlIndex operator+(const AvlChar &v) const { return *this + v.val(); }
const AvlIndex operator+(const AvlIndex &v) const { return *this + v.value; }
friend const AvlIndex operator+(int v1, const AvlIndex &v2) { return v2 + v1; }
friend const AvlIndex operator+(char v1, const AvlIndex &v2) { return v2 + v1; }
friend const AvlIndex operator+(AvlInt v1, const AvlIndex &v2) { return v2 + v1.val(); }
friend const AvlIndex operator+(AvlChar v1, const AvlIndex &v2) { return v2 + v1.val(); }

// compound plus
const AvlIndex& operator+=(int v) { value += v; update(); return *this; }
const AvlIndex& operator+=(size_t v) { value += v; update(); return *this; }
const AvlIndex& operator+=(char v) { value += v; update(); return *this; }
const AvlIndex& operator+=(const AvlInt &v) { return *this += v.val(); }
const AvlIndex& operator+=(const AvlChar &v) { return *this += v.val(); }
const AvlIndex& operator+=(const AvlIndex &v) { return *this += v.value; }

// binary minus
const AvlIndex operator-(int v) const
{
    AvlIndex ret = *this;
    ret.value -= v;
    ret.update();
    return ret;
}

const AvlIndex operator-(size_t v) const
{
    AvlIndex ret = *this;
    ret.value -= v;
    ret.update();
    return ret;
}
const AvlIndex operator-(char v) const
{
    AvlIndex ret = *this;
    ret.value -= v;
    ret.update();
    return ret;
}

const AvlIndex operator-(const AvlInt &v) const { return *this - v.val(); }
```

```
const AvlIndex operator-(const AvlChar &v) const { return *this - v.val(); }
const AvlIndex operator-(const AvlIndex &v) const { return *this - v.value; }
friend const AvlIndex operator-(int v1, const AvlIndex &v2) { return -v2 + v1; }
friend const AvlIndex operator-(char v1, const AvlIndex &v2) { return -v2 + v1; }
friend const AvlIndex operator-(AvlInt v1, const AvlIndex &v2) { return -v2 + v1.val(); }
friend const AvlIndex operator-(AvlChar v1, const AvlIndex &v2) { return -v2 + v1.val(); }

// compound minus
const AvlIndex& operator-=(int v) { value -= v; update(); return *this; }
const AvlIndex& operator-=(char v) { value -= v; update(); return *this; }
const AvlIndex& operator-=(const AvlInt &v) { return *this -= v.val(); }
const AvlIndex& operator-=(const AvlIndex &v) { return *this -= v.value; }
const AvlIndex& operator-=(const AvlChar &v) { return *this -= v.val(); }

// multiplication
const AvlIndex operator*(int v) const
{
    AvlIndex ret = *this;
    ret.value *= v;
    ret.update();
    return ret;
}

const AvlIndex operator*(char v) const
{
    AvlIndex ret = *this;
    ret.value *= v;
    ret.update();
    return ret;
}

const AvlIndex operator*(const AvlInt &v) const { return *this * v.val(); }
const AvlIndex operator*(const AvlChar &v) const { return *this * v.val(); }
const AvlIndex operator*(const AvlIndex &v) const { return *this * v.value; }
friend const AvlIndex operator*(int v1, const AvlIndex &v2) { return v2 * v1; }
friend const AvlIndex operator*(char v1, const AvlIndex &v2) { return v2 * v1; }
friend const AvlIndex operator*(AvlInt v1, const AvlIndex &v2) { return v2 * v1.val(); }
friend const AvlIndex operator*(AvlChar v1, const AvlIndex &v2) { return v2 * v1.val(); }


// compound multiplication
```

```
const AvlIndex& operator*=(int v) { value *= v; update(); return *this; }
const AvlIndex& operator*=(char v) { value *= v; update(); return *this; }
const AvlIndex& operator*=(const AvlInt &v) { return *this *= v.val(); }
const AvlIndex& operator*=(const AvlChar &v) { return *this *= v.val(); }
const AvlIndex& operator*=(const AvlIndex &v) { return *this *= v.value; }

// devision
const AvlIndex operator/(int v) const
{
    AvlIndex ret = *this;
    ret.value /= v;
    ret.update();
    return ret;
}

const AvlIndex operator/(char v) const
{
    AvlIndex ret = *this;
    ret.value /= v;
    ret.update();
    return ret;
}

const AvlIndex operator/(const AvlInt &v) const { return *this / v.val(); }
const AvlIndex operator/(const AvlChar &v) const { return *this / v.val(); }
const AvlIndex operator/(const AvlIndex &v) const { return *this / v.value; }
friend const AvlIndex operator/(int v1, const AvlIndex &v2) { return v2 / v1; }
friend const AvlIndex operator/(char v1, const AvlIndex &v2) { return v2 / v1; }
friend const AvlIndex operator/(AvlInt v1, const AvlIndex &v2) { return v2 / v1.val(); }
friend const AvlIndex operator/(AvlChar v1, const AvlIndex &v2) { return v2 / v1.val(); }

// compound division
const AvlIndex & operator/=(int v) { value /= v; update(); return *this; }
const AvlIndex & operator/=(char v) { value /= v; update(); return *this; }
const AvlIndex & operator/=(const AvlInt &v) { return *this /= v.val(); }
const AvlIndex & operator/=(const AvlChar &v) { return *this /= v.val(); }
const AvlIndex & operator/=(const AvlIndex &v) { return *this /= v.value; }

// comparison operators
bool operator <(int v) const { return value < v; }
bool operator <(size_t v) const { return (size_t)value < v; }
```

```
bool operator <(char v) const { return value < v; }
bool operator <(const AvlInt &v) const { return value < v.val(); }
bool operator <(const AvlChar &v) const { return value < v.val(); }
bool operator <(const AvlIndex &v) const { return value < v.val(); }
friend bool operator<(int v1, const AvlIndex v2) { return v1 < v2.value; }
friend bool operator<(char v1, const AvlIndex v2) { return v1 < v2.value; }
friend bool operator<(AvlInt v1, const AvlIndex v2) { return v1.val() < v2.value; }
friend bool operator<(AvlChar v1, const AvlIndex v2) { return v1.val() < v2.value; }

bool operator <=(int v) const { return value <= v; }
bool operator <=(char v) const { return value <= v; }
bool operator <=(size_t v) const { return (size_t)value <= v; }
bool operator <=(const AvlInt &v) const { return value <= v.val(); }
bool operator <=(const AvlChar &v) const { return value <= v.val(); }
bool operator <=(const AvlIndex &v) const { return value <= v.value; }
friend bool operator<=(int v1, const AvlIndex v2) { return v1 <= v2.value; }
friend bool operator<=(char v1, const AvlIndex v2) { return v1 <= v2.value; }
friend bool operator<=(AvlInt v1, const AvlIndex v2) { return v1.val() <= v2.value; }
friend bool operator<=(AvlChar v1, const AvlIndex v2) { return v1.val() <= v2.value; }

bool operator >(int v) const { return value > v; }
bool operator >(char v) const { return value > v; }
bool operator >(size_t v) const { return (size_t)value > v; }
bool operator >(const AvlInt &v) const { return value > v.val(); }
bool operator >(const AvlChar &v) const { return value > v.val(); }
bool operator >(const AvlIndex &v) const { return value > v.value; }
friend bool operator>(int v1, const AvlIndex v2) { return v1 > v2.value; }
friend bool operator>(char v1, const AvlIndex v2) { return v1 > v2.value; }
friend bool operator>(AvlInt v1, const AvlIndex v2) { return v1.val() > v2.value; }
friend bool operator>(AvlChar v1, const AvlIndex v2) { return v1.val() > v2.value; }

bool operator >=(int v) const { return value >= v; }
bool operator >=(char v) const { return value >= v; }
bool operator >=(size_t v) const { return (size_t)value >= v; }
bool operator >=(const AvlInt &v) const { return value >= v.val(); }
bool operator >=(const AvlChar &v) const { return value >= v.val(); }
bool operator >=(const AvlIndex &v) const { return value >= v.value; }
friend bool operator>=(int v1, const AvlIndex v2) { return v1 >= v2.value; }
friend bool operator>=(char v1, const AvlIndex v2) { return v1 >= v2.value; }
friend bool operator>=(AvlInt v1, const AvlIndex v2) { return v1.val() >= v2.value; }
friend bool operator>=(AvlChar v1, const AvlIndex v2) { return v1.val() >= v2.value; }
```

```
bool operator ==(int v) const { return value == v; }
bool operator ==(char v) const { return value == v; }
bool operator ==(size_t v) const { return (size_t)value == v; }
bool operator ==(const AvlInt &v) const { return value == v.val(); }
bool operator ==(const AvlChar &v) const { return value == v.val(); }
bool operator ==(const AvlIndex &v) const { return value == v.value; }
friend bool operator==(int v1, const AvlIndex v2) { return v1 == v2.value; }
friend bool operator==(char v1, const AvlIndex v2) { return v1 == v2.value; }
friend bool operator==(AvlInt v1, const AvlIndex v2) { return v1.val() == v2.value; }
friend bool operator==(AvlChar v1, const AvlIndex v2) { return v1.val() == v2.value; }

bool operator !=(int v) const { return value != v; }
bool operator !=(char v) const { return value != v; }
bool operator !=(size_t v) const { return (size_t)value != v; }
bool operator !=(const AvlInt &v) const { return value != v.val(); }
bool operator !=(const AvlChar &v) const { return value != v.val(); }
bool operator !=(const AvlIndex &v) const { return value != v.value; }
friend bool operator!=(int v1, const AvlIndex v2) { return v1 != v2.value; }
friend bool operator!=(char v1, const AvlIndex v2) { return v1 != v2.value; }
friend bool operator!=(AvlInt v1, const AvlIndex v2) { return v1.val() != v2.value; }
friend bool operator!=(AvlChar v1, const AvlIndex v2) { return v1.val() != v2.value; }

// value getter
int val() const { return value; }

// render function
virtual void render()
{
    unsigned int red = color() / 0x10000;
    unsigned int green = color() % 0x10000 / 0x100;
    unsigned int blue = color() % 0x100;
    glColor4f(red / 255.0, green / 255.0, blue / 255.0, 0.0);

    for (std::vector<array_entry>::iterator itr = array_set.begin(); itr != array_set.end()
        glRasterPos2f(itr->x, itr->y);
        glutBitmapString(AvlFont().font(), (const unsigned char *)(name().c_str()));

    }
}
```

```
    void highlight() {
        set_color(AVL_COLOR_HIGHLIGHT);
    }
    void lowlight() {
        set_color(AVL_COLOR_DEFAULT);
    }

    void add_array(AvlObject* obj, GLfloat a, GLfloat b) {
        for (size_t i = 0; i < array_set.size(); i ++) {
            if (array_set[i].x == a && array_set[i].y == b)
                return;
            if (obj == array_set[i].obj)
                return;
        }
        array_entry ae;
        ae.obj = obj;
        ae.x = a;
        ae.y = b;
        array_set.push_back(ae);
    }

private:
    int value;
    std::vector<array_entry> array_set;
    void update() { set_width(name().size() * font().width()); }


}; // end of AvlIndex



class AvlString: public AvlObject
{
    public:
        // constructor
        AvlString (const char *v, GLfloat x = 0, GLfloat y = 0, void *font = GLUT_BITMAP_9_BY_
            : AvlObject(x, y, font)
        {
            value = v;
            update();
        }
```

```
AvlString (char v, GLfloat x = 0, GLfloat y = 0, void *font = GLUT_BITMAP_9_BY_15)
    : AvlObject(x, y, font)
{
    value = v;
    update();
}


// destructor
virtual ~AvlString() {}

// assignment operator
const AvlString & operator=(const char *v) { value = v; update(); return *this; }
const AvlString & operator=(char v) { value = v; update(); return *this; }

// << operator
friend std::ostream& operator<<(std::ostream &os, const AvlString &v)
{
    os << v.value;
    return os;
}



// binary plus
const AvlString operator+(const char *v) const
{
    AvlString ret = *this;
    ret.value += v;
    ret.update();
    return ret;
}

const AvlString operator+(char v) const
{
    AvlString ret = *this;
    ret.value += v;
    ret.update();
    return ret;
}
```

```
// binary plus
const AvlString operator+(const AvlChar &v) const { return *this +  v.val(); }
const AvlString operator+(const AvlString &v) const
{
    AvlString ret = *this;
    ret.value += v.val();
    ret.update();
    return ret;
}
friend const AvlString operator+( char v1, const AvlString &v2 )
{
    AvlString ret = v1;
    ret.value += v2.val();
    ret.update();
    return ret;
}
friend const AvlString operator+(const char *v1, const AvlString &v2)
{
    AvlString ret = v1;
    ret.value += v2.val();
    ret.update();
    return ret;
}
friend const AvlString operator+(AvlChar v1, const AvlString &v2) { return v1.val() + v

// compound plus
const AvlString & operator+=(char v) { value += v; update(); return *this; }
const AvlString & operator+=(const AvlChar &v) { return *this += v.val(); }
const AvlString & operator+=(const char *v) { value += v; update(); return *this; }
const AvlString & operator+=(const AvlString &v) { value += v.val(); update(); return *



// value getter
std::string val() const { return value; }

// render function
virtual void render()
{
    unsigned int red = color() / 0x10000;
```

```
            unsigned int green = color() % 0x10000 / 0x100;
            unsigned int blue = color() % 0x100;
            glColor4f(red / 255.0, green / 255.0, blue / 255.0, 0.0);

            glRasterPos2f(x(), y());
            glutBitmapString(font().font(), reinterpret_cast<const unsigned char *> (value.c_st
        }

        void highlight() {
            set_color(AVL_COLOR_HIGHLIGHT);
        }
        void lowlight() {
            set_color(AVL_COLOR_DEFAULT);
        }

    private:
        void update() { set_width(value.size()*font().width()); }
        std::string value;
}; // end of AvlString


// ***NOTE!***: every element in this array is a pointer
template <typename T>
class AvlArray : public AvlObject
{
    public:
        // default constructor
        AvlArray(size_t size = 0, GLfloat x = 0, GLfloat y = 0,
                const AvlFont &font = GLUT_BITMAP_9_BY_15) : AvlObject(x, y, font), arr(size),
        {
            for (size_t i = 0; i < arr.size(); i ++) {
                arr[i] = std::shared_ptr<T>(new T);
            }

            updateMutex = std::shared_ptr<std::mutex>(new std::mutex);
            toplevelArray = this;

            update();
        }

        // constructor with initializer list
```

```
AvlArray(const std::initializer_list<T> &l) : arr(l.size()), index_x(l.size()), index_y
{
    typename std::initializer_list<T>::const_iterator src = l.begin();
    typename std::vector< std::shared_ptr<T> >::iterator dst = arr.begin();

    while (src != l.end()) {
        *dst = std::shared_ptr<T>(new T(*src));
        src++;
        dst++;
    }

    updateMutex = std::shared_ptr<std::mutex>(new std::mutex);
    toplevelArray = this;

    update();
}

// destructor
virtual ~AvlArray() {}

friend std::ostream& operator<<(std::ostream &os, const AvlArray<T> &ar)
{
    typename std::vector< std::shared_ptr<T> >::const_iterator src = ar.arr.begin();

    while (src != ar.arr.end()) {
        os << **src;
        if (src != ar.arr.end() - 1)
            os << " ";
        src++;
    }

    return os;
}

// subscript operator
T& operator[](size_t index) { return *arr[index]; }
const T& operator[](size_t index) const { return *arr[index]; }

// subscript operator with AvlIndex type
T& operator[] (AvlIndex &index) {
    index.add_array(this, index_x[index.val()], index_y[index.val()] - 20);
```

```
    return *arr[index.val()];
}

const T& operator[] (const AvlIndex &index) const {
    return *arr[index.val()];
}

// size
size_t size() const { return arr.size(); }

// sub-array
AvlArray<T> subarray(size_t start, size_t end) const
{
    AvlArray<T> ret(end - start);

    typename std::vector< std::shared_ptr<T> >::const_iterator src = arr.begin() + sta
    typename std::vector< std::shared_ptr<T> >::iterator dst = ret.arr.begin();

    while (src != arr.begin() + end) {
        *dst = *src;
        src++;
        dst++;
    }

    ret.updateMutex = updateMutex;
    ret.toplevelArray = toplevelArray;

    return ret;
}

// sub-array with AvlIndex
AvlArray<T> subarray(const AvlIndex &a, const AvlIndex &b) const
{
    size_t start = a.val();
    size_t end = b.val();
    AvlArray<T> ret(end - start);

    typename std::vector< std::shared_ptr<T> >::const_iterator src = arr.begin() + sta
    typename std::vector< std::shared_ptr<T> >::iterator dst = ret.arr.begin();

    while (src != arr.begin() + end) {
```

```
        *dst = *src;
        src++;
        dst++;
    }

    ret.updateMutex = updateMutex;
    ret.toplevelArray = toplevelArray;

    return ret;
}

// render
virtual void render()
{
    if (updateMutex->try_lock()) {
        update();
        updateMutex->unlock();
    }

    auxiliary_display();


    for (auto& v : arr) {
        v->render();
    }
}

void highlight()
{
    for (auto& v: arr)
        v->highlight();
}

void lowlight()
{
    for (auto& v: toplevelArray->arr)
        v->lowlight();
}

// swap two elements in an array
void swap(size_t idx1, size_t idx2)
```

```
{
    if (!isDisplaying()) {
        T tmp = *arr[idx1];
        *arr[idx1] = *arr[idx2];
        *arr[idx2] = tmp;
        return;
    }

    std::shared_ptr<T> obj[2];
    std::thread moveThread[2];

    obj[0] = arr[idx1];
    obj[1] = arr[idx2];

    updateMutex->lock();

    moveThread[0] = std::thread(moveObject, obj[0], 0, obj[0]->height() * 1.5, DEFAULT_
    moveThread[1] = std::thread(moveObject, obj[1], 0, obj[1]->height() * 1.5, DEFAULT_
    moveThread[0].join();
    moveThread[1].join();

    moveThread[0] = std::thread(moveObject, obj[0], obj[1]->x() - obj[0]->x(), 0, DEFAU
    moveThread[1] = std::thread(moveObject, obj[1], obj[0]->x() - obj[1]->x(), 0, DEFAU
    moveThread[0].join();
    moveThread[1].join();

    moveThread[0] = std::thread(moveObject, obj[0], 0, -obj[0]->height() * 1.5, DEFAULT
    moveThread[1] = std::thread(moveObject, obj[1], 0, -obj[1]->height() * 1.5, DEFAULT
    moveThread[0].join();
    moveThread[1].join();

    T tmp = *arr[idx1];
    *arr[idx1] = *arr[idx2];
    *arr[idx2] = tmp;

    updateMutex->unlock();

    avlSleep(0.1);
}

// swap two elements in an array using AvlIndex
```

```
void swap(const AvlIndex &a, const AvlIndex &b)
{
    size_t idx1 = a.val();
    size_t idx2 = b.val();

    if (!isDisplaying()) {
        T tmp = *arr[idx1];
        *arr[idx1] = *arr[idx2];
        *arr[idx2] = tmp;
        return;
    }

    std::shared_ptr<T> obj[2];
    std::thread moveThread[2];

    obj[0] = arr[idx1];
    obj[1] = arr[idx2];

    updateMutex->lock();

    moveThread[0] = std::thread(moveObject, obj[0], 0, obj[0]->height() * 1.5, DEFAULT_
    moveThread[1] = std::thread(moveObject, obj[1], 0, obj[1]->height() * 1.5, DEFAULT_
    moveThread[0].join();
    moveThread[1].join();

    moveThread[0] = std::thread(moveObject, obj[0], obj[1]->x() - obj[0]->x(), 0, DEFAU
    moveThread[1] = std::thread(moveObject, obj[1], obj[0]->x() - obj[1]->x(), 0, DEFAU
    moveThread[0].join();
    moveThread[1].join();

    moveThread[0] = std::thread(moveObject, obj[0], 0, -obj[0]->height() * 1.5, DEFAULT
    moveThread[1] = std::thread(moveObject, obj[1], 0, -obj[1]->height() * 1.5, DEFAULT
    moveThread[0].join();
    moveThread[1].join();

    T tmp = *arr[idx1];
    *arr[idx1] = *arr[idx2];
    *arr[idx2] = tmp;

    updateMutex->unlock();
```

```
    avlSleep(0.1);
}

//push
void push(const T a){

    std::shared_ptr<T> current = std::shared_ptr<T> (new T);
    *current = a;
    arr.push_back(current);
    index_x.push_back(0);
    index_y.push_back(0);

    update();
}

//pop
T pop(){
    if(empty())
        return 0;
    else{
        T current = **(arr.end()-1);
        arr.pop_back();
        index_x.pop_back();
        index_y.pop_back();

        update();
        return current;
    }
}

//empty
bool empty(){
    if ( arr.size() == 0)
        return true;
    else
        return false;
}

//dequeue
T dequeue(){
    if(empty())
```

```
            return 0;
        else{
            T current = *arr[0];
            arr.erase(arr.begin());
            index_x.pop_back();
            index_y.pop_back();

            update();
            return current;

        }
    }

private:
    void auxiliary_display() {
        // display "Content"
        unsigned int red = AVL_AUXILIARY_COLOR / 0x10000;
        unsigned int green = AVL_AUXILIARY_COLOR % 0x10000 / 0x100;
        unsigned int blue = AVL_AUXILIARY_COLOR % 0x100;
        glColor4f(red / 255.0, green / 255.0, blue / 255.0, 0.0);

        glRasterPos2f(x() - 9*(name().length() + 2), y());
        glutBitmapString(AvlFont().font(), (const unsigned char *)(std::string(this->name()

        // display "Index" and index
        glRasterPos2f(x() - 63, y() - 20);
        glutBitmapString(AvlFont().font(), (const unsigned char *)(std::string("Index: ").c

        for (size_t i = 0; i < arr.size(); i ++) {
            glRasterPos2f(index_x[i], index_y[i]);
            glutBitmapString(AvlFont().font(), (const unsigned char *)(std::to_string(i).c_
        }
    }

    void update()
    {
        float digit_gat = 1.5;

        GLfloat w = -font().width();

        for (typename std::vector< std::shared_ptr<T> >::iterator it = arr.begin();
```

```
                    it != arr.end(); it++) {
                if (it == arr.begin()) {
                    (*it)->set_x( x() );
                    (*it)->set_y( y() );
                    (*it)->set_font( font() );
                    w += (*it)->width() + font().width();

                    index_x[it - arr.begin()] = x();
                    index_y[it - arr.begin()] = y() - 20;
                }

                else {
                    (*it)->set_x( (*(it-1))->x() + (*(it-1))->width() + digit_gat * font().widt
                    (*it)->set_y( (*(it-1))->y() );
                    (*it)->set_font( (*(it-1))->font() );
                    w += (*it)->width() + font().width();

                    index_x[it - arr.begin()] = (*(it-1))->x() + (*(it-1))->width() + digit_gat
                    index_y[it - arr.begin()] = (*(it-1))->y() - 20;
                }

            }

            set_width(w);
        }

        std::vector< std::shared_ptr<T> > arr;
        std::shared_ptr<std::mutex> updateMutex;

        // array of the previous level, or parent level
        AvlArray<T> *toplevelArray;

        friend class AvlIndex;

        std::vector<GLfloat> index_x;
        std::vector<GLfloat> index_y;
}; //end of AvlArray

class AvlBool: public AvlObject
{
    public:
```

```
// constructor
AvlBool(bool v = false, GLfloat x = 0, GLfloat y = 0, void *font = GLUT_BITMAP_9_BY_15)
    : AvlObject(x, y, font)
{
    value = v;
}

// destructor
virtual ~AvlBool() {}

// assignment operator
const AvlBool& operator=(bool v) { value = v; return *this; }

// << operator
friend std::ostream& operator<<(std::ostream &os, const AvlBool &v)
{
    os << v.value;
    return os;
}

// unary operator
const AvlBool operator!() const
{
    AvlBool ret = *this;
    ret.value = !value;
    return ret;
}

// comparison operators
bool operator==(bool v) const { return value == v; }
bool operator==(const AvlBool &v) const { return value == v.value; }
friend bool operator==(bool v1, const AvlBool v2) { return v1 == v2.value; }

bool operator!=(bool v) const { return value != v; }
bool operator!=(const AvlBool &v) const { return value != v.value; }
friend bool operator!=(bool v1, const AvlBool v2) { return v1 != v2.value; }

bool operator&&(bool v) const { return value && v; }
bool operator&&(const AvlBool &v) const { return value && v.value; }
friend bool operator&&(bool v1, const AvlBool v2) { return v1 && v2.value; }
```

```
        bool operator||(bool v) const { return value || v; }
        bool operator||(const AvlBool &v) const { return value || v.value; }
        friend bool operator||(bool v1, const AvlBool v2) { return v1 || v2.value; }

        // value getter
        bool val() const { return value; }

        // render function
        virtual void render()
        {
            unsigned int red = color() / 0x10000;
            unsigned int green = color() % 0x10000 / 0x100;
            unsigned int blue = color() % 0x100;
            glColor4f(red / 255.0, green / 255.0, blue / 255.0, 0.0);

            glRasterPos2f(x(), y());
            std::string display_value = value ? "T" : "F";
            glutBitmapString(font().font(), (const unsigned char *)display_value.c_str());
        }

        void highlight() {
            set_color(AVL_COLOR_HIGHLIGHT);
        }
        void lowlight() {
            set_color(AVL_COLOR_DEFAULT);
        }

    private:
        bool value;
}; // end of AvlBool


#endif // AVL_TYPES_H_
```

file: avl/lib/AvlUtils.h

```
#ifndef AVL_UTILS_H_
#define AVL_UTILS_H_
```

```
#include <thread>
#include <chrono>

inline void avlSleep(float seconds)
{
    int ms = seconds * 1000;
    std::chrono::duration<int, std::milli> milliseconds(ms);

    std::this_thread::sleep_for(milliseconds);
}

#endif // AVL_UTILS_H_
```

file: avl/lib/AvlVisualizer.cpp

```
#include "AvlVisualizer.h"
#include "AvlTypes.h"

int AvlVisualizer::currentWidth = 0;
int AvlVisualizer::currentHeight = 0;
std::unordered_map<std::string, AvlObject *> AvlVisualizer::objects;
std::atomic<int> AvlVisualizer::level(0);
std::stack<int> AvlVisualizer::levelBackup;

AvlVisualizer::AvlVisualizer(int argc, char *argv[])
{
    glutInit(&argc, argv);

    glutInitContextVersion(OpenGL_Major_Version, OpenGL_Minor_Version);
    glutInitContextFlags(GLUT_FORWARD_COMPATIBLE);
    glutInitContextProfile(GLUT_CORE_PROFILE);

    glutSetOption(GLUT_ACTION_ON_WINDOW_CLOSE, GLUT_ACTION_GLUTMAINLOOP_RETURNS);

    glutInitWindowSize(Default_Width, Default_Height);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);

    windowHandle = glutCreateWindow(Title);
```

```
    glutReshapeFunc(AvlVisualizer::avlResize);
    glutDisplayFunc(AvlVisualizer::avlDisplay);

    glClearColor(0.0, 0.0, 0.0, 0.0);
}

AvlVisualizer::~AvlVisualizer()
{
}

void AvlVisualizer::show()
{
    glutMainLoop();
}

void AvlVisualizer::start()
{
    level++;
}

void AvlVisualizer::stop()
{
    level--;
}

void AvlVisualizer::reset()
{
    levelBackup.push(level.load());
    level.store(0);
}

void AvlVisualizer::restore()
{
    level.store(levelBackup.top());
    levelBackup.pop();
}

int AvlVisualizer::getLevel() const
{
    return level.load();
}
```

```
void AvlVisualizer::addObject(AvlObject *obj, const std::string &name)
{
    if (objects.find(name) == objects.end()) {
        objects[name] = obj;
        obj->setVisualizer(this);
    }

    placeObject();
}

void AvlVisualizer::delObject(const std::string &name)
{
    if (objects.find(name) != objects.end())
        objects.erase(name);

    placeObject();
}

void AvlVisualizer::placeObject()
{
    std::vector<AvlObject *> arrays;
    std::vector<AvlObject *> others;

    for (auto& o : objects) {
        AvlObject *obj = o.second;
        if (typeid(*obj) == typeid(AvlArray<AvlInt>) || typeid(*obj) == typeid(AvlArray<AvlCha
                || typeid(*obj) == typeid(AvlArray<AvlBool>))
            arrays.push_back(obj);
        else {
            if (typeid(*obj) != typeid(AvlIndex))
                others.push_back(obj);
        }
    }

    if (!arrays.empty()) {
        int h = Default_Height / arrays.size() / 2;
        for (size_t i = 0; i < arrays.size(); i++) {
            arrays[i]->lock();
            int x = -arrays[i]->width() / 2;
            int y = h * (int(arrays.size()) / 2 - i);
```

```
            arrays[i]->set_x(x);
            arrays[i]->set_y(y);
            arrays[i]->unlock();
        }
    }

    if (!others.empty()) {
        int w = Default_Width / others.size() / 2;
        for (size_t i = 0; i < others.size(); i++) {
            others[i]->lock();
            int x = w * (int(others.size() / 2) - i);
            int y = Default_Height / 2 - AvlFont().height() * 2;
            others[i]->set_x(x);
            others[i]->set_y(y);
            others[i]->unlock();
        }
    }
}

void AvlVisualizer::avlResize(int width, int height)
{
    GLdouble size;
    GLdouble aspect;

    glViewport(0, 0, width, height);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    size = (GLdouble)(width >= height ? width : height) / 2.0;
    if (width <= height) {
        aspect = (GLdouble)height / (GLdouble)width;
        glOrtho(-size, size, -size*aspect, size*aspect, -100000.0, 100000.0);
    }
    else {
        aspect = (GLdouble)width / (GLdouble)height;
        glOrtho(-size*aspect, size*aspect, -size, size, -100000.0, 100000.0);
    }

    glScaled(aspect, aspect, 1.0);

    glMatrixMode(GL_MODELVIEW);
```

```
    glLoadIdentity();
}

void AvlVisualizer::avlDisplay()
{
    if (level.load() > 0) {
        glClearColor(0.0, 0.0, 0.0, 0.0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        for (auto& obj : objects)
            obj.second->render();
    }

    glutSwapBuffers();
    glutPostRedisplay();
}
```

file: avl/lib/AvlVisualizer.h

```
#ifndef AVL_VISUALIZER_H_
#define AVL_VISUALIZER_H_

#include <GL/freeglut.h>
#include <GL/glut.h>
#include <unordered_map>
#include <stack>
#include <string>
#include <atomic>

const int OpenGL_Major_Version = 2;
const int OpenGL_Minor_Version = 1;
const char *const Title = "Algorithm Visualization";
const int Default_Width = 640;
const int Default_Height = 480;

class AvlObject;

class AvlVisualizer
{
```

```
    public:
        AvlVisualizer(int argc, char *argv[]);
        ~AvlVisualizer();

        void show();

        void addObject(AvlObject *obj, const std::string &name);
        void delObject(const std::string &name);

        void start();
        void stop();
        void reset();
        void restore();
        int getLevel() const;

    private:
        static void avlResize(int width, int height);
        static void avlDisplay();

        void placeObject();

        int windowHandle;
        static int currentWidth;
        static int currentHeight;

        static std::unordered_map<std::string, AvlObject *> objects;

        static std::atomic<int> level;
        static std::stack<int> levelBackup;
};

#endif // AVL_VISUALIZER_H_
```

file: avl/lib/Makefile.am

```
lib_LTLIBRARIES = libavl.la
libavl_la_SOURCES = AvlVisualizer.cpp
include_HEADERS = AvlVisualizer.h AvlTypes.h AvlUtils.h
AM_CPPFLAGS = -Wall -Wextra -Werror -std=c++11 -I/opt/local/include
```

file: avl/m4/.gitignore

```
# Ignore everything in this directory
*
# Except this file
!.gitignore
```

file: avl/Makefile.am

```
ACLOCAL_AMFLAGS = -I m4
SUBDIRS = src lib sample_code tests
```

file: avl/README.md

```
AVL
===
Please see [Issues](https://github.com/wqfish/avl/issues) for bugs and creating new tasks.

How to build
============

1. Prerequisites
    - Mac
        * Setup [MacPorts](http://guide.macports.org)
        * Install [XQuartz](https://xquartz.macosforge.org/)
        * Install dependencies
        ```bash
        sudo port install astyle automake libtool freeglut glew glm glfw
        ```
    - Linux
        * Only need to install libraries
        ```bash
        sudo apt-get install **
        ```
```

2. Create configure script.
    ```bash
    autoreconf --install
    ```

3. Create Makefiles.
    ```bash
    ./configure
    ```

4. Make.
    ```bash
    make
    ```

5. Install
    ```bash
    sudo make install
    ```
This would install avl to /usr/local/bin, header files to /usr/local/include
and libraries to /usr/local/lib.

How to run test cases
=====================

```bash
make check
```
You can run ```make check -j [N]``` if you have multiple CPUs.

How to add a test case
======================

Suppose you want to test ```if``` control flow.

1. Create ```if1.avl``` in ```tests/avl.test/```, assuming you have multiple test cases for ``

2. Write a bash script ```if1.sh```, also put it into ```tests/avl.test/```:
    ```bash
    #!/bin/bash
    ```

````
    avl -o if1 if1.avl
    ```
````

Don't forget ```chmod +x if1.sh```. If the test case is expected to fail, the bash script should be as follows:

````
    ```bash
    #!/bin/bash

    avl -t if1.avl
    ```
````

since the error should be reported before invoking `g++`.

3. Modify ```tests/avl.test/Makefile.am```, add ```if1.sh``` to ```TESTS``` variable.
If the test case is expected to fail, also add the script to ```XFAIL_TESTS``` variable.
Whether the test case passes or fails depends on the exit code of the script (0 for success).

4. Rerun ```make check```.

file: avl/sample_code/.gitignore

```
sorting
merge_sorted_array
```

file: avl/sample_code/Makefile.am

```
noinst_PROGRAMS = sorting merge_sorted_array
sorting_SOURCES = sorting.cpp
merge_sorted_array_SOURCES = merge_sorted_array.cpp

AM_CPPFLAGS = -Wall -Werror -std=c++11 -I$(top_builddir)/lib -I/opt/local/include
AM_LDFLAGS = -L$(top_builddir)/lib -L/opt/local/lib -lavl -lGL -lglut
```

file: avl/sample_code/merge_sorted_array.avl

```
/* merge two sorted array @a and @b
```
```

```
 * and store the result in @c */

void merge(int a[], int b[], int c[])
{
    <begin_display>

    display index i = 0;
    display index j = 0;
    display index k = 0;

    while (true) {
        if (i >= len(a) || j >= len(b))
            break;

        if (a[i] < b[j]) {
            swap(a, i, i);
            c[k] = a[i++];
        }
        else {
            swap(b, j, j);
            c[k] = b[j++];
        }

        k++;
    }

    for (i ; i < len(a); i++) {
        swap(a, i, i);
        c[k] = a[i];
        k++;
    }
    for (j; j < len(b); j++) {
        swap(b, j, j);
        c[k] = b[j];
        k++;
    }

    <end_display>
}

int main()
```

```
{
    <begin_display>

    display int a[] = {1, 5, 7, 9, 20};
    display int b[] = {2, 4, 8, 16};
    display int c[9];

    merge(a, b, c);

    <end_display>

    print c;

    return 0;
}
```

file: avl/sample_code/merge_sorted_array.cpp

```
#include <AvlVisualizer.h>
#include <AvlTypes.h>
#include <condition_variable>
#include <cstdlib>

AvlVisualizer *__avl__vi = NULL;
bool __avl__ready() { return __avl__vi != NULL; }
std::mutex __avl__mtx;
std::condition_variable_any __avl__cv;

void __avl__display(int argc, char **argv)
{
    __avl__mtx.lock();
    __avl__vi = new AvlVisualizer(argc, argv);
    __avl__cv.notify_one();
    __avl__mtx.unlock();

    __avl__vi->show();
}

void merge(AvlArray<AvlInt> a, AvlArray<AvlInt> b, AvlArray<AvlInt> c)
```

```
{
    __avl__vi->reset();
    a.lowlight();
    a.highlight();
    b.lowlight();
    b.highlight();
    c.lowlight();
    c.highlight();

    __avl__vi->start();
    avlSleep(0.5);

    AvlIndex i = 0;
    i.set_name("i");
    __avl__vi->addObject(&i, "i");
    AvlIndex j = 0;
    j.set_name("j");
    __avl__vi->addObject(&j, "j");
    AvlIndex k = 0;
    k.set_name("k");
    __avl__vi->addObject(&k, "k");

    while (true) {
        if (i >= a.size() || j >= b.size())
            break;

        c[k].assign(a[i] < b[j] ? a[i++] : b[j++]);
        k++;
    }

    for (; i < a.size(); i++) {
        c[k].assign(a[i]);
        k++;
    }
    for (; j < b.size(); j++) {
        c[k].assign(b[j]);
        k++;
    }

    avlSleep(0.1);
    __avl__vi->stop();
```

```
    __avl__vi->delObject("i");
    __avl__vi->delObject("j");
    __avl__vi->delObject("k");

    a.lowlight();
    b.lowlight();
    c.lowlight();
    __avl__vi->restore();
}

int main(int argc, char *argv[])
{
    std::thread __avl__loop(__avl__display, argc, argv);
    __avl__mtx.lock();
    __avl__cv.wait(__avl__mtx, __avl__ready);
    __avl__mtx.unlock();
    avlSleep(0.5);

    __avl__vi->start();
    avlSleep(0.5);

    AvlArray<AvlInt> a = {1, 5, 7, 9, 20};
    a.set_name("a");
    __avl__vi->addObject(&a, "a");
    AvlArray<AvlInt> b = {2, 4, 8, 16};
    b.set_name("b");
    __avl__vi->addObject(&b, "b");
    AvlArray<AvlInt> c(9);
    c.set_name("c");
    __avl__vi->addObject(&c, "c");

    merge(a, b, c);

    avlSleep(0.1);
    __avl__vi->stop();

    std::cout << c << std::endl;

    __avl__vi->delObject("a");
    __avl__vi->delObject("b");
```

```
    __avl__vi->delObject("c");

    __avl__loop.join();
    delete __avl__vi;

    return 0;
}
```

file: avl/sample_code/sorting.avl

```
void quicksort(int a[])
{
    if (len(a) <= 1)
        return;

    index i = 0;
    index j = 0;
    index k = len(a) - 1;

    <begin_display>
    while (j < k) {
        if (a[j] >= a[k]) {
            j = j + 1;
        }
        else {
            swap(a, i, j);
            i = i + 1;
            j++;
        }
    }

    swap(a, i, k);

    quicksort(a[0 : i]);
    quicksort(a[(i + 1) : (k + 1)]);
    <end_display>
}

void randomPermute(int a[])
```

```
{
    for (index i = 0; i < len(a); i++) {
        index j = rand() % len(a);
        swap(a, i, j);
    }
}

int main() {
    display int a[] = {5, 51, 2, 42, 7, 3, 6, 8, 10, 3, 11, 5, 9};

    print "before sorting: ", a;

    <begin_display>
    for (index i = 1; i < len(a); i++) {
        index j = i - 1;
        while (j >= 0 && a[j] > a[j+1]) {
            swap(a, j + 1, j);
            j = j - 1;
        }
    }

    print "after bubblesort: ", a;

    randomPermute(a);

    print "after permutation: ", a;

    quicksort(a);

    print "after quicksort: ", a;

    <end_display>

    return 0;
}
```

file: avl/sample_code/sorting.cpp

```
/* Headers:
```

```
 * insert the following lines at the beginning of
 * each target C++ file */
#include <AvlVisualizer.h>
#include <AvlTypes.h>
#include <condition_variable>
#include <cstdlib>

/* Note: these variable names are also reserved words,
 * so we add a prefix __avl__. */
AvlVisualizer *__avl__vi = NULL;
bool __avl__ready() { return __avl__vi != NULL; }
std::mutex __avl__mtx;
std::condition_variable_any __avl__cv;

void __avl__display(int argc, char **argv)
{
    __avl__mtx.lock();
    __avl__vi = new AvlVisualizer(argc, argv);
    __avl__cv.notify_one();
    __avl__mtx.unlock();

    __avl__vi->show();
}
/* End of Headers */


/* User defined function:
 * NOTE: the parameters should be passed by value */
void quicksort(AvlArray<AvlInt> a)
{
    /* Insert the following three lines at the
     * beginning of each function except main() */
    __avl__vi->reset();
    a.lowlight();
    a.highlight();

    /* len(a) -> a.size() */
    if (a.size() <= 1) {
        /* NOTE: 'return' is equivalent to end of function
         * so the following two lines are needed. Additional
         * '{' and '}' are also required.
```

```
     * !!!FIXME!!!: Is a return between <begin_display> and
     * <end_display> allowed? Seems no problem. */
    a.lowlight();
    __avl__vi->restore();
    return;
}

/* Translate type index to AvlIndex,
 * index should be non-negative */
AvlIndex i = 0;
i.set_name("i");
AvlIndex j = 0;
j.set_name("j");
AvlIndex k = a.size() - 1;
k.set_name("k");


/* substitude <begin_display> with the following two lines */
__avl__vi->start();
avlSleep(0.5);

while (j < k) {
    if (a[j] >= a[k]) {
        j = j + 1;
    }
    else {
        /* swap(a, i, j) -> a.swap(i, j) */
        a.swap(i, j);
        i = i + 1;
        j++;
    }
}

/* same as above */
a.swap(i, k);

/* translation rule for subarrys,
 * TODO: operator precedence? ':' and '+-' */
quicksort(a.subarray(0, i));
quicksort(a.subarray((i+1), (k+1)));
```

```
    /* substitude <end_display> with the following two lines */
    avlSleep(0.1);
    __avl__vi->stop();

    /* Insert the following two lines at the end of each function */
    a.lowlight();
    __avl__vi->restore();
}

/* This is an example of user defined function
 * which does not contain <begin_display> and
 * <end_display>, it should not be displayed */
void randomPermute(AvlArray<AvlInt> a)
{
    /* Insert the following three lines at the
     * beginning of each function except main */
    __avl__vi->reset();
    a.lowlight();
    a.highlight();

    /* !!FIXME!!
     * To Shining Sun:
     * we cannot translate "index i = 0" to
     * "AvlIndex i = 0; i.set_name("i")" here. */
    for (AvlIndex i = 0; i < a.size(); i++) {
        /* Should rand be a keyword or a function?
         * If function, consider inline C (like inline
         * assembly in C) */
        AvlIndex j = rand() % a.size();
        j.set_name("j");
        a.swap(i, j);
    }

    /* Insert the following two lines at the end of each function */
    a.lowlight();
    __avl__vi->restore();
}

/* main:
 * NOTE: argc and argv are required */
int main(int argc, char *argv[])
```

```
{
    /* only for main:
     * insert the following five lines at the beginning of main */
    std::thread __avl__loop(__avl__display, argc, argv);
    __avl__mtx.lock();
    __avl__cv.wait(__avl__mtx, __avl__ready);
    __avl__mtx.unlock();
    avlSleep(0.5);



    /* substitude array declaration with the following two lines */
    AvlArray<AvlInt> a = {5, 51, 2, 42, 7, 3, 6, 8, 10, 3, 11};

    a.set_name("a");
    /* If the variable should be displayed, the following line is needed */
    __avl__vi->addObject(&a, "a");

    /* translation rule for print */
    std::cout << "before sorting: " << a << std::endl;

    /* 1. Insertion Sort */

    /* substitude <begin_display> with the following two lines */
    __avl__vi->start();
    avlSleep(0.5);

    /* !!FIXME!!
     * same issue as in randomPermute */
    for (AvlIndex i = 1; i < a.size(); i++) {
        i.set_name("i");
        __avl__vi->addObject(&i, "i");
        AvlIndex j = i - 1;
        j.set_name("j");
        __avl__vi->addObject(&j, "j");
        while (j >= 0 && a[j] > a[j+1]) {
            a.swap(j + 1, j);
            j = j - 1;
        }
    }
```

```
    std::cout << "after insertion sort: " << a << std::endl;

    /* 2. random permutation */

    /* This function won't be displayed, although it is
     * put between <begin_display> and <end_display> */
    randomPermute(a);

    std::cout << "after permutation: " << a << std::endl;

    /* 3. quicksort */

    /* substitude <begin_display> with the following two lines */
    __avl__vi->start();
    avlSleep(0.5);


    quicksort(a);

    std::cout << "after quicksort: " << a << std::endl;

    /* substitude <end_display> with the following two lines */
    avlSleep(0.1);
    __avl__vi->stop();

    /* only for main:
     * insert the following two lines at the end of main
     * BEFORE return 0 */
    __avl__loop.join();
    delete __avl__vi;

    return 0;
}
```

file: avl/src/.gitignore

```
avl
parser.h
parser.c
```

```
parser.output
scanner.c



file: avl/src/avl.c

#include <sys/stat.h>
#include <sys/wait.h>
#include <unistd.h>
#include <getopt.h>
#include <string.h>
#include <stdlib.h>
#include <stdarg.h>
#include <stdio.h>
#include "config.h"
#include "syntax_tree.h"

int yyparse();

const char *const USAGE =
    "Usage: " PACKAGE " [-h|-o|-t] file\n"
    "Options:\n"
    "  -h --help                Display this information\n"
    "  -o --output=<file>       Compile and place the executable into <file>\n"
    "  -t --translate           Translate the source files into c++\n"
    "                           xxx.avl -> xxx.cpp\n"
    "Use at most one option at a time.\n"
    "\n"
    "For bug reporting instructions, please see:\n"
    "<" PACKAGE_URL ">.\n";

#define MAX_FILENAME 256
char input_file[MAX_FILENAME];
char output_file[MAX_FILENAME];
char translate_file[MAX_FILENAME];
char temp_file[MAX_FILENAME];
char format_file[MAX_FILENAME];
int translate_flag = 0;
int output_flag = 0;
```

```
extern FILE *yyin;
extern FILE *yyout;

const char *const DEFAULT_OUTPUT = "a.out";
const char *const DEFAULT_EXT = ".avl";
const char *const TRANSLATE_EXT = ".cpp";
const char *const DEFAULT_TEMP = "/tmp/avl_temp.XXXXXX";
char *const CXX_OPTIONS[] = {
    "g++",
    "-x",
    "c++",
    "-std=c++11",
    "-Wall",
    "-o",
    output_file,
    temp_file,
    "-I/opt/local/include",
    "-I/usr/include",
    "-I/usr/local/include",
    "-L/opt/local/lib",
    "-L/usr/lib",
    "-L/usr/local/lib",
    "-lavl",
    "-lGL",
    "-lglut",
    NULL
};
char *const ASTYLE_OPTIONS[] = {
    "astyle",
    "--style=linux",
    "--suffix=none",
    format_file,
    NULL
};

const struct option long_options[] = {
    {"help",      no_argument,       NULL, 'h'},
    {"output",    required_argument, NULL, 'o'},
    {"translate", no_argument,       NULL, 't'},
    {0, 0, 0, 0}
};
```

```
void usage()
{
    fprintf(stderr, "%s", USAGE);
}

void die(const char *format, ...)
{
    va_list ap;

    va_start(ap, format);
    fprintf(stderr, PACKAGE ": ");
    vfprintf(stderr, format, ap);
    fprintf(stderr, "\n");
    va_end(ap);

    exit(EXIT_FAILURE);
}

void die_err(const char *msg)
{
    perror(msg);
    exit(EXIT_FAILURE);
}

void parse_command_line(int argc, char *argv[])
{
    memset(input_file, 0, MAX_FILENAME);
    memset(output_file, 0, MAX_FILENAME);
    memset(translate_file, 0, MAX_FILENAME);
    memset(temp_file, 0, MAX_FILENAME);
    memset(format_file, 0, MAX_FILENAME);
    strcpy(output_file, DEFAULT_OUTPUT);
    strcpy(temp_file, DEFAULT_TEMP);

    while (1) {
        int option_index = 0;
        char c = getopt_long(argc, argv, "ho:t", long_options, &option_index);
        if (c == -1)
            break;
```

```
    switch (c) {
        case 'h':
            usage();
            exit(EXIT_SUCCESS);
        case 'o':
            if (!optarg) {
                usage();
                exit(EXIT_FAILURE);
            }
            if (strlen(optarg) <= 0)
                die("no output filename.");
            else if (strlen(optarg) >= MAX_FILENAME)
                die("output filename too long: %s.", optarg);
            strcpy(output_file, optarg);
            output_flag = 1;
            break;
        case 't':
            translate_flag = 1;
            break;
        default:
            usage();
            exit(EXIT_FAILURE);
    }
}

if (output_flag && translate_flag)
    die("Please specify at most one option at a time.");

if (optind != argc - 1) {
    usage();
    exit(EXIT_FAILURE);
}
if (strlen(argv[optind]) <= strlen(DEFAULT_EXT)
        || strlen(argv[optind]) >= MAX_FILENAME)
    die("invalid input filename: %s.", argv[optind]);
strcpy(input_file, argv[optind]);
if (strcmp(input_file + strlen(input_file) - strlen(DEFAULT_EXT),
            DEFAULT_EXT) != 0)
    die("not a valid avl source file: %s.", input_file);

struct stat buf;
```

```
    if (stat(input_file, &buf) != 0)
        die_err("invalid input file");

    if (translate_flag) {
        strcpy(translate_file, input_file);
        strcpy(translate_file + strlen(translate_file) - strlen(DEFAULT_EXT),
                TRANSLATE_EXT);
    }
}

void execute_program(char *const *options)
{
    pid_t pid = fork();

    if (pid < 0)
        die_err("fork() failed");
    else if (pid == 0) {
        char *const *s = options;
        while (*s)
            printf("%s ", *s++);
        printf("\n");

        if (execvp(options[0], options) < 0)
            die_err("execvp() failed");
    }

    int status;
    if (waitpid(pid, &status, 0) != pid)
        die_err("waitpid() failed");
    if (WEXITSTATUS(status))
        die("failed to execute %s.", options[0]);
}

int main(int argc, char *argv[])
{
    parse_command_line(argc, argv);

    yyin = fopen(input_file, "r");
    if (!yyin)
        die_err("can not open input file");
```

```
    if (translate_flag) {
        yyout = fopen(translate_file, "w");
        if (!yyout)
            die_err("can not open translate file");
    }
    else {
        int fd = mkstemp(temp_file);
        if (fd < 0)
            die_err("mkstemp() failed");
        yyout = fdopen(fd, "w");
        if (!yyout)
            die_err("can not open temporary file");
    }

    if (yyparse() != 0)
        die("Parsing failed.");

    fclose(yyin);
    fclose(yyout);

    /* code formatting */
    if (translate_flag)
        strcpy(format_file, translate_file);
    else
        strcpy(format_file, temp_file);

    execute_program(ASTYLE_OPTIONS);

    /* invoking g++ */
    if (!translate_flag)
        execute_program(CXX_OPTIONS);

    return 0;
}
```

file: avl/src/code_generator.c

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include "syntax_tree.h"
#include "st_list.h"
#include "sym_table.h"

void freeTree(nodeType* node);
int generateOpNode(oprNode* opr);
int generateSubtree(nodeType* node);
void print_append(char* s, int space);
void print_append_int(int value, int space);
void print_indent();
void print_header();
void print_line(const char *line);
void avl_code_generator(nodeType* root);

int newLine = 1;
int indent = 0;
extern FILE *yyout;
char** paraList;
int paraCount;
int mainFun;
int needSleep;

// function for generate code from syntax tree rooted at nodeType* root
void avl_code_generator(nodeType* root) {
    printf("begin translate.\n");
    print_header();
    paraList = NULL;
    int ret = generateSubtree(root);
    freeTree(root);
    if (ret == 0) {
        printf("Succeed\n");
    } else {
        printf("Fail\n");
    }
}

// count nunber of paramenters of a function
int numPara(oprNode* node) {
    if (node->opType == concatenate)
        return numPara(&node->op[0]->opr) + 1;
```

```
    return 1;
}

// generate code for the tree rooted at node (called recursively)
int generateSubtree(nodeType* node) {
    switch (node->type) {
        // leaf node
        case INTCON_NODE:
            print_append_int(node->intCon.value, 1);
            break;
        case CHARCON_NODE:
            print_append(node->charCon.value, 1);
            break;
        case BOOLCON_NODE:
            if (node->boolCon.value)
                print_append("true", 1);
            else
                print_append("false", 1);
            break;
        case STRLIT_NODE:
            print_append(node->strLit.value, 1);
            break;
        case VARTYPE_NODE:
            switch (node->varType.value) {
                case VOID_TYPE:
                    print_append("void", 1);
                    break;
                case CHAR_TYPE:
                    print_append("AvlChar", 1);
                    break;
                case INT_TYPE:
                    print_append("AvlInt", 1);
                    break;
                case STRING_TYPE:
                    print_append("std::string", 1);
                    break;
                case INDEX_TYPE:
                    print_append("AvlIndex", 1);
                    break;
                case BOOL_TYPE:
                    print_append("AvlBool", 1);
```

```
                    break;
            }
            break;
        case ID_NODE:
            print_append(node->id.value, 1);
            break;
        case MATHOP_NODE:
            print_append(node->mathOp.value, 1);
            break;
        // inner node
        case OPERATOR_NODE:
            if(generateOpNode(&node->opr)) return 1;
            break;
    }
    return 0;
}

// generate code for a tree rooted at an operation node (inner node)
int generateOpNode(oprNode* opr) {
    int i;
    char* id;
    nodeType* temp;
    switch (opr->opType) { // operation node type
        case parentheses_exp:
            print_append("(", 1);
            if (generateSubtree(opr->op[0])) return 1;
            print_append(")", 1);
            break;
        case array:
            if (generateSubtree(opr->op[0])) return 1;
            if (opr->numOperands == 2) {
                print_append("[", 0);
                if (generateSubtree(opr->op[1])) return 1;
                print_append("]", 1);
            } else {
                print_append(".subarray(", 0);
                if (generateSubtree(opr->op[1])) return 1;
                print_append(",", 0);
                if (generateSubtree(opr->op[2])) return 1;
                print_append(")", 1);
            }
```

```
    break;
case func_call:
    if (generateSubtree(opr->op[0])) return 1;
    print_append("(", 0);
    if (opr->numOperands == 2 && generateSubtree(opr->op[1])) return 1;
    print_append(")", 1);
    break;
case concatenate:
    if (generateSubtree(opr->op[0])) return 1;
    print_append(",", 0);
    if (generateSubtree(opr->op[1])) return 1;
    break;
case math_op:
    for (i=0; i<opr->numOperands; i++) {
        if (generateSubtree(opr->op[i])) return 1;
    }
    break;
case len:
    if (generateSubtree(opr->op[0])) return 1;
    print_append(".size()", 0);
    break;
case assignment:
    if (generateSubtree(opr->op[0])) return 1;
    print_append("=", 1);
    if (generateSubtree(opr->op[1])) return 1;
    break;
case disp_exp:
    if (opr->op[0]->type != ID_NODE) return 1;
    print_indent();
    fprintf(yyout, "__avl__vi->addObject(&%s, \"%s\")", opr->op[0]->id.value, opr->op[0
    break;
case hide_exp:
    if (opr->op[0]->type != ID_NODE) return 1;
    print_indent();
    fprintf(yyout, "__avl__vi->delObject(\"%s\")", opr->op[0]->id.value);
    break;
case swap:
    if (generateSubtree(opr->op[0])) return 1;
    print_append(".swap(", 0);
    if (generateSubtree(opr->op[1])) return 1;
    print_append(",", 0);
```

```
    if (generateSubtree(opr->op[2])) return 1;
    print_append(")", 1);
    break;
case print:
    print_append("std::cout << (", 1);
    if (generateSubtree(opr->op[0])) return 1;
    print_append(") << std::endl", 1);
    break;
case print_list:
    if (generateSubtree(opr->op[0])) return 1;
    if (opr->numOperands == 2) {
        print_append(") << (", 1);
        if (generateSubtree(opr->op[1])) return 1;
    }
    break;
case var_decl:
case var_decl_disp:
case var_decl_hide:
    // whether is an array
    temp = opr->op[1];
    while (temp->type == OPERATOR_NODE) {
        if (temp->opr.opType == arr_decl) break;
        temp = temp->opr.op[0];
    }
    if (temp->opr.opType == arr_decl) { // is array
        print_append("AvlArray<", 1);
        if (generateSubtree(opr->op[0])) return 1;
        print_append(">", 0);
        if (generateSubtree(temp->opr.op[0])) return 1;
        if (temp->opr.numOperands == 2) {
            if (!(opr->op[1]->type == OPERATOR_NODE && opr->op[1]->opr.opType == assig
                print_append("(", 0);
                if (generateSubtree(temp->opr.op[1])) return 1;
                print_append(")", 0);
            }
        }
    } else { // not array
        if (generateSubtree(opr->op[0])) return 1;
        if (generateSubtree(temp)) return 1;
    }
    // assignment
```

```
    if (opr->op[1]->type == OPERATOR_NODE && opr->op[1]->opr.opType == assignment) {
        print_append("=", 1);
        if (generateSubtree(opr->op[1]->opr.op[1])) return 1;
    }
    print_append(";", 0);
    newLine = 1;
    // id to temp
    while (temp->type == OPERATOR_NODE) {
        temp = temp->opr.op[0];
    }
    id = temp->id.value;
    print_indent();
    if (opr->op[0]->varType.value == STRING_TYPE)
        break;
    fprintf(yyout, "%s.set_name(\"%s\");\n", id, id);
    if (opr->opType == var_decl_disp) {
        print_indent();
        fprintf(yyout, "__avl__vi->addObject(&%s, \"%s\");\n",id, id);
    }
    newLine = 1;
    break;
case arr_decl:
    if (generateSubtree(opr->op[0])) return 1;
    break;
case init_list:
    print_append("{", 1);
    if (generateSubtree(opr->op[0])) return 1;
    print_append("}", 1);
    break;
case empty_state:
    print_append(";", 0);
    newLine = 1;
    break;
case exp_state:
    if (needSleep) print_append("avlSleep(0.15);", 1);
    newLine = 1;
    if (generateSubtree(opr->op[0])) return 1;
    print_append(";", 0);
    newLine = 1;
    if (needSleep) print_append("avlSleep(0.15);", 1);
    newLine = 1;
```

```
    break;
case declar_state:
    if (generateSubtree(opr->op[0])) return 1;
    break;
case comp_state:
    print_append("{", 1);
    newLine = 1;
    indent ++;
    if (opr->numOperands == 1 && generateSubtree(opr->op[0])) return 1;
    indent --;
    print_append("}", 0);
    newLine = 1;
    break;
case state_list:
    if (generateSubtree(opr->op[0])) return 1;
    if (generateSubtree(opr->op[1])) return 1;
    break;
case display_state:
    needSleep ++;
    print_append("__avl__vi->start();", 1);
    newLine = 1;
    print_append("avlSleep(0.5);", 0);
    newLine = 1;
    if (generateSubtree(opr->op[0])) return 1;
    needSleep --;
    print_append("avlSleep(0.1);", 1);
    newLine = 1;
    print_append("__avl__vi->stop();", 0);
    newLine = 1;
    break;
case select_state:
    print_append("if (", 1);
    if (generateSubtree(opr->op[0])) return 1;
    print_append(")", 1);
    if (opr->op[1]->type == OPERATOR_NODE && opr->op[1]->opr.opType != comp_state) {
        print_append("{", 1);
        newLine = 1;
        indent ++;
        if (generateSubtree(opr->op[1])) return 1;
        indent --;
        print_append("}", 0);
```

```
            newLine = 1;
        } else {
            if (generateSubtree(opr->op[1])) return 1;
        }
        if (opr->numOperands == 3) {
            print_append("else", 1);
            if (opr->op[2]->type == OPERATOR_NODE && opr->op[2]->opr.opType != comp_state)
                print_append("{", 1);
                newLine = 1;
                indent ++;
                if (generateSubtree(opr->op[2])) return 1;
                indent --;
                print_append("}", 0);
                newLine = 1;
            } else {
                if (generateSubtree(opr->op[2])) return 1;
            }
        }
        break;
    case while_state:
        print_append("while (", 1);
        if (generateSubtree(opr->op[0])) return 1;
        print_append(")", 1);
        if (opr->op[1]->type == OPERATOR_NODE && opr->op[1]->opr.opType != comp_state) {
            print_append("{", 1);
            newLine = 1;
            indent ++;
            if (generateSubtree(opr->op[1])) return 1;
            indent --;
            print_append("}", 0);
            newLine = 1;
        } else {
            if (generateSubtree(opr->op[1])) return 1;
        }
        break;
    case do_while_state:
        print_append("do", 1);
        if (opr->op[0]->type == OPERATOR_NODE && opr->op[0]->opr.opType != comp_state) {
            print_append("{", 1);
            newLine = 1;
            indent ++;
```

```
        if (generateSubtree(opr->op[0])) return 1;
        indent --;
        print_append("}", 0);
        newLine = 1;
    } else {
        if (generateSubtree(opr->op[0])) return 1;
    }
    print_append("while (", 1);
    if (generateSubtree(opr->op[1])) return 1;
    print_append(");", 1);
    newLine = 1;
    break;
case for_state:
    //declare in for
    if (opr->op[0]->opr.opType == var_decl ||opr->op[0]->opr.opType == var_decl_disp |
        print_append("{", 0);
        newLine = 1;
        indent ++;
        if (generateSubtree(opr->op[0])) return 1;
        print_append("for (", 1);
        print_append(";", 0);
        if (generateSubtree(opr->op[1])) return 1;
        print_append(";", 0);
        if (opr->numOperands == 4 && generateSubtree(opr->op[2])) return 1;
        print_append(")", 1);
        if (opr->op[opr->numOperands-1]->type == OPERATOR_NODE && opr->op[opr->numOpera
            print_append("{", 1);
            newLine = 1;
            indent ++;
            if (generateSubtree(opr->op[opr->numOperands-1])) return 1;
            indent --;
            print_append("}", 0);
            newLine = 1;
        } else {
            if (generateSubtree(opr->op[opr->numOperands-1])) return 1;
        }
        indent --;
        print_append("}", 0);
        newLine = 1;
    } else {
        print_append("for (", 1);
```

```
        if (generateSubtree(opr->op[0])) return 1;
        print_append(";", 0);
        if (generateSubtree(opr->op[1])) return 1;
        print_append(";", 0);
        if (opr->numOperands == 4 && generateSubtree(opr->op[2])) return 1;
        print_append(")", 1);
        if (opr->op[opr->numOperands-1]->type == OPERATOR_NODE && opr->op[opr->numOpera
            print_append("{", 1);
            newLine = 1;
            indent ++;
            if (generateSubtree(opr->op[opr->numOperands-1])) return 1;
            indent --;
            print_append("}", 0);
            newLine = 1;
        } else {
            if (generateSubtree(opr->op[opr->numOperands-1])) return 1;
        }
    }
    break;
case jump_continue_state:
    print_append("continue;", 1);
    newLine = 1;
    break;
case jump_break_state:
    print_append("break;", 1);
    newLine = 1;
    break;
case jump_ret_state:
    if (mainFun) {
        print_append("__avl__loop.join();", 1);
        newLine = 1;
        print_append("delete __avl__vi;", 1);
        newLine = 1;
    } else {
        for (i=0; i<paraCount; i++) {
            print_append(paraList[i], 0);
            print_append(".lowlight();", 0);
            newLine = 1;
        }
        print_append("__avl__vi->restore();", 1);
        newLine = 1;
```

```
    }
    print_append("return", 1);
    if (opr->numOperands == 1 && generateSubtree(opr->op[0])) return 1;
    print_append(";", 0);
    newLine = 1;
    break;
case trans_unit:
    if (generateSubtree(opr->op[0])) return 1;
    if (generateSubtree(opr->op[1])) return 1;
    break;
case func_def:
    needSleep = 0;
    if (strcmp(opr->op[1]->id.value,"main") == 0)
        mainFun = 1;
    else
        mainFun = 0;
    if (mainFun) {
        if (opr->op[0]->varType.value == INT_TYPE) print_append("int", 0);
        else print_append("void", 0);
    } else {
        if (generateSubtree(opr->op[0])) return 1;
    }
    if (generateSubtree(opr->op[1])) return 1;
    print_append("(", 0);
    if (paraList) free(paraList);
    paraCount = 0;
    if (mainFun) {
        print_append("int argc, char *argv[]",0);
    } else if (opr->numOperands == 4) {
        paraList = (char**)malloc(sizeof(char*)*numPara(&opr->op[2]->opr));
        if (generateSubtree(opr->op[2])) return 1;
    }
    print_append(")", 1);
    newLine = 1;
    print_append("{",0);
    newLine = 1;
    indent ++;
    if (mainFun) {
        print_line("\tstd::thread __avl__loop(__avl__display, argc, argv);");
        print_line("\t__avl__mtx.lock();");
        print_line("\t__avl__cv.wait(__avl__mtx, __avl__ready);");
```

```
            print_line("\t__avl__mtx.unlock();");
            print_line("\tavlSleep(0.5);");
        } else {
            print_line("\t__avl__vi->reset();");
            for (i=0; i<paraCount; i++) {
                print_append(paraList[i], 0);
                print_append(".lowlight();", 0);
                newLine = 1;
                print_append(paraList[i], 0);
                print_append(".highlight();", 0);
                newLine = 1;
            }
        }
        if (generateSubtree(opr->op[opr->numOperands-1])) return 1;
        if (opr->op[0]->varType.value == VOID_TYPE) {
            if (mainFun) {
                print_line("\t__avl__loop.join();");
                print_line("\tdelete __avl__vi;");
            } else {
                for (i=0; i<paraCount; i++) {
                    print_append(paraList[i], 0);
                    print_append(".lowlight();", 0);
                    newLine = 1;
                }
                print_line("\t__avl__vi->restore();");
            }
        }
        indent --;
        newLine = 1;
        print_append("}",0);
        break;
    case para_declar:
        // whether is an array
        temp = opr->op[1];
        while (temp->type == OPERATOR_NODE) {
            if (temp->opr.opType == arr_decl) break;
            temp = temp->opr.op[0];
        }
        if (temp->opr.opType == arr_decl) { // is array
            print_append("AvlArray<", 1);
            if (generateSubtree(opr->op[0])) return 1;
```

```
                print_append(">", 0);
                if (generateSubtree(temp->opr.op[0])) return 1;
                paraList[paraCount] = temp->opr.op[0]->id.value;
                paraCount ++;
            } else { // not array
                if (generateSubtree(opr->op[0])) return 1;
                if (generateSubtree(temp)) return 1;
            }
            break;
    }
    return 0;
}

void print_append(char* s, int space) {
    if (newLine) {
        fprintf(yyout, "\n");
        print_indent();
        newLine = 0;
        fprintf(yyout, "%s", s);
    } else {
        if (space)
            fprintf(yyout, " ");
        fprintf(yyout, "%s", s);
    }
}

void print_append_int(int value, int space) {
    if (newLine) {
        fprintf(yyout, "\n");
        print_indent();
        newLine = 0;
        fprintf(yyout, "%d", value);
    } else {
        if (space)
            fprintf(yyout, " ");
        fprintf(yyout, "%d", value);
    }
}

void print_indent() {
    int i;
```

```
    for (i=0; i<indent; i++) {
        fprintf(yyout, "\t");
    }
}

void print_header() {
    print_line("#include <AvlVisualizer.h>");
    print_line("#include <AvlTypes.h>");
    print_line("#include <condition_variable>");
    print_line("#include <cstdlib>");
    print_line("");

    print_line("AvlVisualizer *__avl__vi = NULL;");
    print_line("bool __avl__ready() { return __avl__vi != NULL; }");
    print_line("std::mutex __avl__mtx;");
    print_line("std::condition_variable_any __avl__cv;");
    print_line("");

    print_line("void __avl__display(int argc, char **argv)");
    print_line("{");
    print_line("\t__avl__mtx.lock();");
    print_line("\t__avl__vi = new AvlVisualizer(argc, argv);");
    print_line("\t__avl__cv.notify_one();");
    print_line("\t__avl__mtx.unlock();");
    print_line("");
    print_line("\t__avl__vi->show();");
    print_line("}");
    print_line("");
}

void print_line(const char *line) {
    fprintf(yyout, "%s\n", line);
}

void freeTree(nodeType* node) {
    int i;
    switch (node->type) {
        case CHARCON_NODE:
            free(node->charCon.value);
            break;
        case STRLIT_NODE:
```

```
            free(node->strLit.value);
            break;
        case ID_NODE:
            free(node->id.value);
            break;
        case MATHOP_NODE:
            free(node->mathOp.value);
            break;
        case OPERATOR_NODE:
            for (i=0; i<node->opr.numOperands; i++) {
                freeTree(node->opr.op[i]);
            }
            free(node->opr.op);
            break;
        default:
            break;
    }
    free(node);
}
```

file: avl/src/Makefile.am

```
BUILT_SOURCES = parser.h
AM_YFLAGS = -d
bin_PROGRAMS = avl
avl_SOURCES = scanner.l parser.y avl.c syntax_tree.c code_generator.c sym_list.c sym_table.c st
AM_CFLAGS = -Werror
```

file: avl/src/parser.y

```
%{
#include "syntax_tree.h"

int yylex();
void yyerror(const char *format, ...);
%}
```

```
%debug

%union {
    int intConVal;
    char* strLitVal;
    char* idVal;
    char* charLitVal;
    struct nodeTypeTag* nt;
}

%token <idVal> IDENTIFIER
%token LEN
%token <intConVal> CONSTANT
%token <strLitVal> STRING_LITERAL
%token <charLitVal> CHAR_LITERAL

%token INC_OP DEC_OP LE_OP GE_OP EQ_OP NE_OP
%token AND_OP OR_OP

%token PRINT SWAP

%token CHAR INT VOID BOOL INDEX STRING
%token DISPLAY HIDE

%token IF ELSE WHILE DO FOR CONTINUE BREAK RETURN END_DISPLAY BEGIN_DISPLAY TRUE FALSE

%start program

%%

primary_expression
    : IDENTIFIER                        { $<nt>$ = idNodeCreator($<idVal>1); }
    | CONSTANT                          { $<nt>$ = intConNodeCreator($<intConVal>1); }
    | TRUE                              { $<nt>$ = boolConNodeCreator(1); }
    | FALSE                             { $<nt>$ = boolConNodeCreator(0); }
    | CHAR_LITERAL                      { $<nt>$ = charConNodeCreator($<charLitVal>1); }
    | STRING_LITERAL                    { $<nt>$ = strLitNodeCreator($<strLitVal>1); }
    | '(' conditional_expression ')'    { $<nt>$ = operatorNodeCreator(parentheses_exp, 1, $<nt
    ;

postfix_expression
```

```
    : primary_expression                                                    { $<nt>$ = $<nt>1;
    | IDENTIFIER '[' conditional_expression ']'                             { $<nt>$ = operator
    | IDENTIFIER '[' conditional_expression ':' conditional_expression ']'  { $<nt>$ = operator
    | IDENTIFIER '(' ')'                                                    { $<nt>$ = operator
    | IDENTIFIER '(' argument_expression_list ')'                           { $<nt>$ = operator
    | postfix_expression INC_OP                                             { $<nt>$ = operator
    | postfix_expression DEC_OP                                             { $<nt>$ = operator
    ;

argument_expression_list
    : conditional_expression                                { $<nt>$ = $<nt>1; }
    | argument_expression_list ',' conditional_expression   { $<nt>$ = operatorNodeCreator(cond
    ;

unary_expression
    : postfix_expression               { $<nt>$ = $<nt>1;}
    | INC_OP unary_expression          { $<nt>$ = operatorNodeCreator(math_op, 2, mathOpNodeCr
    | DEC_OP unary_expression          { $<nt>$ = operatorNodeCreator(math_op, 2, mathOpNodeCr
    | unary_operator unary_expression    { $<nt>$ = operatorNodeCreator(math_op, 2, $<nt>1, $<n
    | LEN '(' unary_expression ')'     { $<nt>$ = operatorNodeCreator(len, 1, $<nt>3); }
    ;

unary_operator
    : '+'       { $<nt>$ = mathOpNodeCreator("+"); }
    | '-'       { $<nt>$ = mathOpNodeCreator("-"); }
    | '!'       { $<nt>$ = mathOpNodeCreator("!"); }
    ;

multiplicative_expression
    : unary_expression                                { $<nt>$ = $<nt>1; }
    | multiplicative_expression '*' unary_expression { $<nt>$ = operatorNodeCreator(math_op, 3
    | multiplicative_expression '/' unary_expression { $<nt>$ = operatorNodeCreator(math_op, 3
    | multiplicative_expression '%' unary_expression { $<nt>$ = operatorNodeCreator(math_op, 3
    ;

additive_expression
    : multiplicative_expression                         { $<nt>$ = $<nt>1; }
    | additive_expression '+' multiplicative_expression { $<nt>$ = operatorNodeCreator(math_op
    | additive_expression '-' multiplicative_expression { $<nt>$ = operatorNodeCreator(math_op
    ;
```

```
relational_expression
    : additive_expression                                { $<nt>$ = $<nt>1; }
    | relational_expression '<' additive_expression      { $<nt>$ = operatorNodeCreator(math_op
    | relational_expression '>' additive_expression      { $<nt>$ = operatorNodeCreator(math_op
    | relational_expression LE_OP additive_expression    { $<nt>$ = operatorNodeCreator(math_op
    | relational_expression GE_OP additive_expression    { $<nt>$ = operatorNodeCreator(math_op
    ;

equality_expression
    : relational_expression                              { $<nt>$ = $<nt>1; }
    | equality_expression EQ_OP relational_expression    { $<nt>$ = operatorNodeCreator(math_op
    | equality_expression NE_OP relational_expression    { $<nt>$ = operatorNodeCreator(math_op
    ;

logical_and_expression
    : equality_expression                                { $<nt>$ = $<nt>1; }
    | logical_and_expression AND_OP equality_expression { $<nt>$ = operatorNodeCreator(math_op
    ;

logical_or_expression
    : logical_and_expression                                 { $<nt>$ = $<nt>1; }
    | logical_or_expression OR_OP logical_and_expression     { $<nt>$ = operatorNodeCreator(matl
    ;

conditional_expression
    : logical_or_expression                                                          { $<nt>$ =
    | logical_or_expression '?' conditional_expression ':' conditional_expression    { $<nt>$ =
    ;

assignment_expression
    : postfix_expression '=' conditional_expression      { $<nt>$ = operatorNodeCreator(assignme
    ;

type_specifier
    : VOID                      { $<nt>$ = varTypeNodeCreator(VOID_TYPE); }
    | CHAR                      { $<nt>$ = varTypeNodeCreator(CHAR_TYPE); }
    | INT                       { $<nt>$ = varTypeNodeCreator(INT_TYPE); }
    | STRING                    { $<nt>$ = varTypeNodeCreator(STRING_TYPE); }
    | INDEX                     { $<nt>$ = varTypeNodeCreator(INDEX_TYPE); }
    | BOOL                      { $<nt>$ = varTypeNodeCreator(BOOL_TYPE); }
    ;
```

```
expression
    : conditional_expression                                                    { $<nt>$ =
    | assignment_expression                                                     { $<nt>$ =
    | DISPLAY IDENTIFIER                                                        { $<nt>$ =
    | HIDE IDENTIFIER                                                           { $<nt>$ =
    | SWAP '(' IDENTIFIER ',' conditional_expression ',' conditional_expression ')' { $<nt>$ =
    | PRINT print_list                                                          { $<nt>$ =
    ;

print_list
    : conditional_expression                        { $<nt>$ = $<nt>1; }
    | print_list ',' conditional_expression         { $<nt>$ = operatorNodeCreator(print_list,
    ;

declaration
    : type_specifier init_declarator                { $<nt>$ = operatorNodeCreator(var_decl, 2
    | DISPLAY type_specifier init_declarator        { $<nt>$ = operatorNodeCreator(var_decl_dis
    | HIDE type_specifier init_declarator           { $<nt>$ = operatorNodeCreator(var_decl_hid
    ;

init_declarator
    : declarator                                    { $<nt>$ = $<nt>1; }
    | declarator '=' initializer                    { $<nt>$ = operatorNodeCreator(assignment,
    ;

declarator
    : IDENTIFIER                                    { $<nt>$ = idNodeCreator($<idVal>1); }
    | IDENTIFIER '[' conditional_expression ']'     { $<nt>$ = operatorNodeCreator(arr_decl, 2
    | IDENTIFIER '[' ']'                            { $<nt>$ = operatorNodeCreator(arr_decl, 1
    ;

initializer
    : conditional_expression                        { $<nt>$ = $<nt>1; }
    | '{' initializer_list '}'                      { $<nt>$ = operatorNodeCreator(init_list, i
    ;

initializer_list
    : conditional_expression                        { $<nt>$ = $<nt>1; }
    | initializer_list ',' conditional_expression   { $<nt>$ = operatorNodeCreator(concatenate
    ;
```

```
statement
    : compound_statement                                { $<nt>$ = $<nt>1; }
    | expression_statement                              { $<nt>$ = $<nt>1; }
    | declaration_statement                             { $<nt>$ = $<nt>1; }
    | display_statement                                 { $<nt>$ = $<nt>1; }
    | selection_statement                               { $<nt>$ = $<nt>1; }
    | iteration_statement                               { $<nt>$ = $<nt>1; }
    | jump_statement                                    { $<nt>$ = $<nt>1; }
    | ';'                                               { $<nt>$ = operatorNodeCreator(empty_state
    ;

expression_statement
    : expression ';'                                    { $<nt>$ = operatorNodeCreator(exp_state, 1
    ;

declaration_statement
    : declaration ';'                                   { $<nt>$ = operatorNodeCreator(declar_state
    ;

compound_statement
    : '{' '}'                                           { $<nt>$ = operatorNodeCreator(comp_state,
    | '{' statement_list '}'                            { $<nt>$ = operatorNodeCreator(comp_state,
    ;

statement_list
    : statement                                         { $<nt>$ = $<nt>1; }
    | statement_list statement                          { $<nt>$ = operatorNodeCreator(state_list,
    ;

display_statement
    : BEGIN_DISPLAY statement_list END_DISPLAY          { $<nt>$ = operatorNodeCreator(display_stat
    ;

selection_statement
    : IF '(' conditional_expression ')' statement                   { $<nt>$ = operatorNodeCrea
    | IF '(' conditional_expression ')' statement ELSE statement    { $<nt>$ = operatorNodeCrea
    ;

iteration_statement
    : WHILE '(' conditional_expression ')' statement                                { $<nt>$ =
```

```
    | DO statement WHILE '(' conditional_expression ')' ';'                          { $<nt>$ =
    | FOR '(' expression ';' conditional_expression ';' ')' statement               { $<nt>$ =
    | FOR '(' expression ';' conditional_expression ';' expression ')' statement    { $<nt>$ =
    | FOR '(' declaration ';' conditional_expression ';' ')' statement              { $<nt>$ =
    | FOR '(' declaration ';' conditional_expression ';' expression ')' statement   { $<nt>$ =
    ;

jump_statement
    : CONTINUE ';'                                  { $<nt>$ = operatorNodeCreator(jump_continu
    | BREAK ';'                                     { $<nt>$ = operatorNodeCreator(jump_break_s
    | RETURN ';'                                    { $<nt>$ = operatorNodeCreator(jump_ret_sta
    | RETURN conditional_expression ';'             { $<nt>$ = operatorNodeCreator(jump_ret_sta
    ;

translation_unit
    : function_definition                   { $<nt>$ = $<nt>1; }
    | translation_unit function_definition  { $<nt>$ = operatorNodeCreator(trans_unit, 2, $<nt>
    ;

function_definition
    : type_specifier  IDENTIFIER '(' parameter_list ')' compound_statement  { $<nt>$ = operator
    | type_specifier  IDENTIFIER '('  ')' compound_statement                { $<nt>$ = operator
    ;

parameter_list
    : parameter_declaration                     { $<nt>$ = $<nt>1; }
    | parameter_list ',' parameter_declaration  { $<nt>$ = operatorNodeCreator(concatenate, 2,
    ;

parameter_declaration
    : type_specifier declarator                 { $<nt>$ = operatorNodeCreator(para_declar, 2,
    ;

program
    : translation_unit                          { int ret = typeChecking($<nt>1); if (!ret) av
    ;

%%
```

file: avl/src/scanner.l

```
%{
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include "parser.h"

void yyerror(const char *format, ...);
%}

%option debug warn verbose

D                   [0-9]
L                   [A-Za-z_]
C                   [A-Za-z0-9,.?!:; ]

%x COMMENT

%%

"/*"                { BEGIN(COMMENT); }
<COMMENT>"*/"       { BEGIN(INITIAL); }
<COMMENT>.          { /* disregard all characters in comments. */ }

true                { return TRUE; }
false               { return FALSE; }
break               { return BREAK; }
char                { return CHAR; }
string              { return STRING; }
continue            { return CONTINUE; }
do                  { return DO; }
else                { return ELSE; }
for                 { return FOR; }
if                  { return IF; }
int                 { return INT; }
index               { return INDEX; }
return              { return RETURN; }
```

```
len                 { return LEN; }
void                { return VOID; }
while               { return WHILE; }
swap                { return SWAP; }
print               { return PRINT; }
bool                { return BOOL; }
display             { return DISPLAY; }
hide                { return HIDE; }
\<begin_display\>   { return BEGIN_DISPLAY; }
\<end_display\>     { return END_DISPLAY; }

{L}({L}|{D})*       { /* identifier */
                        yylval.idVal = strdup(yytext);
                        return IDENTIFIER;
                    }

{D}+                { /* integer */
                        yylval.intConVal = atoi(yytext);
                        return CONSTANT;
                    }

'{C}'               { /* single character, escape characters are not allowed */
                        yylval.strLitVal = strdup(yytext);
                        return CHAR_LITERAL;
                    }

\"{C}*\"            { /* string literal, escape characters are not allowed */
                        yylval.strLitVal = strdup(yytext);
                        return STRING_LITERAL;
                    }

"++"                { return INC_OP; }
"--"                { return DEC_OP; }
"&&"                { return AND_OP; }
"||"                { return OR_OP; }
"<="                { return LE_OP; }
">="                { return GE_OP; }
"=="                { return EQ_OP; }
"!="                { return NE_OP; }
";"                 { return ';'; }
"{"                 { return '{'; }
```

```
"}"                 { return '}'; }
","                 { return ','; }
":"                 { return ':'; }
"="                 { return '='; }
"("                 { return '('; }
")"                 { return ')'; }
"["                 { return '['; }
"]"                 { return ']'; }
"!"                 { return '!'; }
"-"                 { return '-'; }
"+"                 { return '+'; }
"*"                 { return '*'; }
"/"                 { return '/'; }
"%"                 { return '%'; }
"<"                 { return '<'; }
">"                 { return '>'; }
"?"                 { return '?'; }

[ \t\n]             { /* disregard white spaces */ }
.                   { yyerror("bad character found: %s", yytext); }

%%

int yywrap()
{
    return 1;
}

void yyerror(const char *format, ...)
{
    va_list ap;

    va_start(ap, format);
    vfprintf(stderr, format, ap);
    fprintf(stderr, "\n");
    va_end(ap);
}
```

file: avl/src/st_list.c

```
#include <stdlib.h>
#include <string.h>
#include "st_list.h"

void st_list_init(struct st_list *sl)
{
    sl->first = NULL;
    sl->last = NULL;
}

void st_list_destroy(struct st_list *sl)
{
    while (sl->first)
    {
        struct st_node *node = sl->first->next;
        free(sl->first);
        sl->first = node;
    }

    sl->last = NULL;
}

void st_list_add(struct st_list *sl, struct sym_table *st)
{
    struct st_node *node = (struct st_node *)malloc(sizeof(struct st_node));
    memset(node, 0, sizeof(struct st_node));
    node->st = st;

    if (!sl->first)
    {
        sl->first = node;
        sl->last = node;
    }
    else
    {
        node->next = sl->first;
        sl->first->prev = node;
        sl->first = node;
    }
}
```

```
void st_list_add_tail(struct st_list *sl, struct sym_table *st)
{
    struct st_node *node = (struct st_node *)malloc(sizeof(struct st_node));
    memset(node, 0, sizeof(struct st_node));
    node->st = st;

    if (!sl->first)
    {
        sl->first = node;
        sl->last = node;
    }
    else
    {
        node->prev = sl->last;
        sl->last->next = node;
        sl->last = node;
    }
}

void st_list_del(struct st_list *sl)
{
    if (!sl->first)
        return;

    if (sl->first == sl->last)
    {
        free(sl->first);
        sl->first = NULL;
        sl->last = NULL;
        return;
    }

    struct st_node *node = sl->first->next;
    free(sl->first);
    node->prev = NULL;
    sl->first = node;
}

void st_list_del_tail(struct st_list *sl)
{
```

```
    if (!sl->last)
        return;

    if (sl->first == sl->last)
    {
        free(sl->last);
        sl->first = NULL;
        sl->last = NULL;
        return;
    }

    struct st_node *node = sl->last->prev;
    free(sl->last);
    node->next = NULL;
    sl->last = node;
}

struct sym_table *st_list_head(const struct st_list *sl)
{
    return sl->first->st;
}

struct sym_table *st_list_tail(const struct st_list *sl)
{
    return sl->last->st;
}

struct identifier *st_list_find(const struct st_list *st, const char *id)
{
    struct st_node *node = st->first;
    struct identifier *ret = NULL;

    while (node)
    {
        ret = sym_table_find(node->st, id);
        if (ret)
            return ret;

        node = node->next;
    }
```

```
    return ret;
}
```

file: avl/src/st_list.h

```
#ifndef ST_LIST_H_
#define ST_LIST_H_

#include "sym_table.h"

struct st_node
{
    struct sym_table *st;
    struct st_node *prev;
    struct st_node *next;
};

struct st_list
{
    struct st_node *first;
    struct st_node *last;
};

/* initialize a list for symbol tables */
void st_list_init(struct st_list *sl);

/* destroy a list for symbol tables */
void st_list_destroy(struct st_list *sl);

/* add a symbol table to the head of a list */
void st_list_add(struct st_list *sl, struct sym_table *st);

/* add a symbol table to the end of a list */
void st_list_add_tail(struct st_list *sl, struct sym_table *st);

/* delete a symbol table from the head of a list */
void st_list_del(struct st_list *sl);

/* delete a symbol table from the end of a list */
```

```
void st_list_del_tail(struct st_list *sl);

/* return the head of a list */
struct sym_table *st_list_head(const struct st_list *sl);

/* return the end of a list */
struct sym_table *st_list_tail(const struct st_list *sl);

/* search for an identifier in a list,
 * return a pointer if the id is found,
 * return NULL otherwise*/
struct identifier *
st_list_find(const struct st_list *st, const char *id);

#endif /* ST_LIST_H_ */
```

file: avl/src/sym_list.c

```
#include <string.h>
#include <stdlib.h>
#include "sym_list.h"

void sym_list_init(struct sym_list *sl)
{
    sl->first = NULL;
    sl->last = NULL;
}

void sym_list_destroy(struct sym_list *sl)
{
    while (sl->first)
    {
        struct identifier_node *node = sl->first->next;
        free(sl->first);
        sl->first = node;
    }

    sl->last = NULL;
}
```

```
void sym_list_add(struct sym_list *sl, const struct identifier *id)
{
    if (sym_list_find(sl, id->name) != NULL)
        return;

    struct identifier_node *node = (struct identifier_node *)malloc(sizeof(struct identifier_no
    memset(node, 0, sizeof(struct identifier_node));
    memcpy(&node->id, id, sizeof(struct identifier));

    if (!sl->first)
    {
        sl->first = node;
        sl->last = node;
    }
    else
    {
        node->next = sl->first;
        sl->first->prev = node;
        sl->first = node;
    }
}

void sym_list_add_tail(struct sym_list *sl, const struct identifier *id)
{
    if (sym_list_find(sl, id->name) != NULL)
        return;

    struct identifier_node *node = (struct identifier_node *)malloc(sizeof(struct identifier_no
    memset(node, 0, sizeof(struct identifier_node));
    memcpy(&node->id, id, sizeof(struct identifier));

    if (!sl->first)
    {
        sl->first = node;
        sl->last = node;
    }
    else
    {
        node->prev = sl->last;
        sl->last->next = node;
```

```
        sl->last = node;
    }
}

void sym_list_del(struct sym_list *sl, const char *id)
{
    if (sym_list_find(sl, id) == NULL)
        return;

    if (sl->first == sl->last)
    {
        free(sl->first);
        sl->first = NULL;
        sl->last = NULL;
        return;
    }

    struct identifier_node *node = sl->first;

    while (node)
    {
        if (strcmp(node->id.name, id) == 0)
        {
            if (node == sl->first)
            {
                node->next->prev = NULL;
                sl->first = node->next;
            }
            else if (node == sl->last)
            {
                node->prev->next = NULL;
                sl->last = node->prev;
            }
            else
            {
                node->prev->next = node->next;
                node->next->prev = node->prev;
            }
            free(node);
            return;
        }
```

```
        else
            node = node->next;
    }
}

struct identifier *sym_list_find(const struct sym_list *sl, const char *id)
{
    struct identifier_node *node = sl->first;

    while (node)
    {
        if (strcmp(node->id.name, id) == 0)
            return &node->id;

        node = node->next;
    }

    return NULL;
}
```

file: avl/src/sym_list.h

```
#ifndef SYM_LIST_H_
#define SYM_LIST_H_

#include "syntax_tree.h"

/* the max length of any identifier is SYM_LEN - 1 */
//#define SYM_LEN 32

struct identifier
{
    char* name;
    /* more attributes to be added */
    varTypeEnum type;
    int isArray;
    int isFunc;
    int numArgs;
    varTypeEnum* args;
```

```
    int* argsIsArray;
};

struct identifier_node
{
    struct identifier id;
    struct identifier_node *prev;
    struct identifier_node *next;
};

struct sym_list
{
    struct identifier_node *first;
    struct identifier_node *last;
};

/* initialize a symbol list */
void sym_list_init(struct sym_list *sl);

/* destroy a symbol list */
void sym_list_destroy(struct sym_list *sl);

/* add an identifier to the head of a list */
void sym_list_add(struct sym_list *sl, const struct identifier *id);

/* add an identifier to the tail of a list */
void sym_list_add_tail(struct sym_list *sl, const struct identifier *id);

/* delete an identifier */
void sym_list_del(struct sym_list *sl, const char *id);

/* search for an identifier in a list,
 * return a pointer if the id is found,
 * return NULL otherwise */
struct identifier *
sym_list_find(const struct sym_list *sl, const char *id);

#endif /* SYM_LIST_H_ */
```

file: avl/src/sym_table.c

```
#include <string.h>
#include "sym_table.h"

size_t hash(const char *id)
{
    size_t num = 0;
    size_t len = strlen(id);
    size_t i;

    for (i = 0; i < len; i++)
    {
        num *= 64;
        num %= SYM_TABLE_SIZE;

        if (id[i] == '_')
            num += 0;
        else if ('a' <= id[i] && id[i] <= 'z')
            num += (unsigned int)(id[i] - 'a' + 1);
        else if ('A' <= id[i] && id[i] <= 'Z')
            num += (unsigned int)(id[i] - 'A' + 27);
        else /* '0' <= id[i] && id[i] <= '9' */
            num += (unsigned int)(id[i] - '0' + 53);
    }

    return num % SYM_TABLE_SIZE;
}

void sym_table_init(struct sym_table *st)
{
    size_t i;

    for (i = 0; i < SYM_TABLE_SIZE; i++)
        sym_list_init(&st->htable[i]);
}

void sym_table_destroy(struct sym_table *st)
{
    size_t i;
```

```
    for (i = 0; i < SYM_TABLE_SIZE; i++)
        sym_list_destroy(&st->htable[i]);
}

void sym_table_add(struct sym_table *st, const struct identifier *id)
{
    size_t idx = hash(id->name);

    sym_list_add(&st->htable[idx], id);
}

void sym_table_del(struct sym_table *st, const char *id)
{
    size_t idx = hash(id);

    sym_list_del(&st->htable[idx], id);
}

struct identifier *sym_table_find(const struct sym_table *st, const char *id)
{
    size_t idx = hash(id);

    return sym_list_find(&st->htable[idx], id);
}
```

file: avl/src/sym_table.h

```
#ifndef SYM_TABLE_H_
#define SYM_TABLE_H_

#include "sym_list.h"

#define SYM_TABLE_SIZE 701

struct sym_table
{
    struct sym_list htable[SYM_TABLE_SIZE];
};
```

```
/* initialize a symbol table */
void sym_table_init(struct sym_table *st);

/* destroy a symbol table */
void sym_table_destroy(struct sym_table *st);

/* add an identifier to a symbol table */
void sym_table_add(struct sym_table *st, const struct identifier *id);

/* delete an identifier */
void sym_table_del(struct sym_table *st, const char *id);

/* search for an identifier in a symbol table,
 * return a pointer if the id is found,
 * return NULL otherwise */
struct identifier *sym_table_find(const struct sym_table *st, const char *id);

#endif /* SYM_TABLE_H_ */
```

file: avl/src/syntax_tree.c

```
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include "syntax_tree.h"

////////////////////////////////////////////////////////////

nodeType* intConNodeCreator (int value) {
    nodeType *p;

    // allocating memory
    if ((p = (nodeType*)malloc(sizeof(nodeType))) == NULL)
        printf("out of memory\n");

    p->type = INTCON_NODE;
    p->intCon.value = value;
```

```
    return p;
}

//////////////////////////////////////////////////////////////

nodeType* charConNodeCreator (char* value) {
    nodeType *p;

    // allocating memory
    if ((p = (nodeType*)malloc(sizeof(nodeType))) == NULL)
        printf("out of memory\n");

    p->type = CHARCON_NODE;
    p->charCon.value = value;

    return p;
}

nodeType* boolConNodeCreator (int value) {
    nodeType *p;

    // allocating memory
    if ((p = (nodeType*)malloc(sizeof(nodeType))) == NULL)
        printf("out of memory\n");

    p->type = BOOLCON_NODE;
    p->boolCon.value = value;

    return p;
}
//////////////////////////////////////////////////////////////

nodeType* strLitNodeCreator (char* value) {
    nodeType* p;

    // allocating memory
    if ((p = (nodeType*)malloc(sizeof(nodeType))) == NULL)
        printf("out of memory\n");

    p->type = STRLIT_NODE;
    p->strLit.value = value;
```

```
    return p;
}

////////////////////////////////////////////////////////////

nodeType* varTypeNodeCreator (varTypeEnum type) {
    nodeType* p;

    // allocating memory
    if ((p = (nodeType*)malloc(sizeof(nodeType))) == NULL)
        printf("out of memory\n");

    p->type = VARTYPE_NODE;
    p->varType.value = type;

    return p;
}

////////////////////////////////////////////////////////////

nodeType* idNodeCreator (char* value) {
    nodeType* p;

    // allocating memory
    if ((p = (nodeType*)malloc(sizeof(nodeType))) == NULL)
        printf("out of memory\n");

    p->type = ID_NODE;
    p->id.value = value;

    return p;
}

////////////////////////////////////////////////////////////

nodeType* mathOpNodeCreator(char* value) {
    nodeType* p;

    // allocating memroy
    if ((p = (nodeType*)malloc(sizeof(nodeType))) == NULL)
```

```
        printf("out of memory\n");

    p->type = MATHOP_NODE;
    p->mathOp.value = strdup(value);

    return p;
}

////////////////////////////////////////////////////////////

nodeType* operatorNodeCreator (operatorTypeEnum oprtr, int numOperands, ...) {
    nodeType* p;
    va_list ap;

    // allocating memory
    if ((p = (nodeType*)malloc(sizeof(nodeType))) == NULL)
        printf("out of memory\n");

    if ((p->opr.op = (nodeType**)malloc(numOperands * sizeof(nodeType*))) == NULL)
        printf("out of memory\n");

    p->type = OPERATOR_NODE;
    p->opr.opType = oprtr;
    p->opr.numOperands = numOperands;

    va_start(ap, numOperands);
    int i;
    for (i = 0; i < numOperands; i ++) {
        p->opr.op[i] = va_arg(ap, nodeType*);
    }
    va_end(ap);

    return p;
}
```

file: avl/src/syntax_tree.h

```
#ifndef SYNTAX_TREE_H_
#define SYNTAX_TREE_H_
```

```
typedef enum {
    INTCON_NODE,
    BOOLCON_NODE,
    CHARCON_NODE,
    STRLIT_NODE,
    VARTYPE_NODE,
    ID_NODE,
    MATHOP_NODE,
    OPERATOR_NODE
} nodeTypeEnum;

typedef enum {
    parentheses_exp,
    array,
    func_call,
    concatenate,
    math_op,
    len,
    assignment,
    disp_exp,
    hide_exp,
    swap,
    print,
    print_list,
    var_decl,
    var_decl_disp,
    var_decl_hide,
    arr_decl,
    init_list,
    empty_state,
    exp_state,
    declar_state,
    comp_state,
    state_list,
    display_state,
    select_state,
    while_state,
    do_while_state,
    for_state,
    jump_continue_state,
```

```
    jump_break_state,
    jump_ret_state,
    trans_unit,
    func_def,
    para_declar
} operatorTypeEnum;

typedef enum {
    VOID_TYPE,
    CHAR_TYPE,
    INT_TYPE,
    STRING_TYPE,
    INDEX_TYPE,
    BOOL_TYPE
} varTypeEnum;

// integer constants
typedef struct {
    int value;
} intConNode;

//char constants
typedef struct {
    char* value;
} charConNode;

// bool constants
typedef struct {
    int value;
} boolConNode;

// string literal
typedef struct {
    char* value;
} strLitNode;

// type specifier
typedef struct {
    varTypeEnum value;
} varTypeNode;
```

```
// identifiers
typedef struct {
    char* value;
} idNode;

// math operator
typedef struct {
    char* value;
} mathOpNode;

// operators
typedef struct {
    operatorTypeEnum opType;
    int numOperands;
    struct nodeTypeTag **op;
} oprNode;


////////////////////////////////////////////////////////

typedef struct nodeTypeTag {
    nodeTypeEnum type;

    union {
        intConNode intCon;
        charConNode charCon;
        boolConNode boolCon;
        strLitNode strLit;
        varTypeNode varType;
        idNode id;
        mathOpNode mathOp;
        oprNode opr;

    };

} nodeType;

////////////////////////////////////////////////////////

nodeType* intConNodeCreator (int value);
```

```
nodeType* charConNodeCreator (char* value);

nodeType* boolConNodeCreator (int value);

nodeType* strLitNodeCreator (char* value);

nodeType* varTypeNodeCreator (varTypeEnum type);

nodeType* idNodeCreator (char* value);

nodeType* mathOpNodeCreator(char* value);

nodeType* operatorNodeCreator (operatorTypeEnum, int, ...);

void avl_code_generator(nodeType* root);

int typeChecking(nodeType* root);

void freeTree(nodeType* node);

#endif
```

file: avl/src/type_check.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "syntax_tree.h"
#include "st_list.h"
#include "sym_table.h"

void freeTree(nodeType* node);
int typeCheckingOpNode(oprNode* opr);
int typeCheckingSubTree(nodeType* node);

void getParaList(nodeType*);
void clearParaList();

char* idGen(nodeType* node);
```

```
varTypeEnum typeGen(nodeType* node);


extern FILE *yyout;
struct identifier** paraList;
size_t paraCount;
int mainFun;

struct st_list symbolTableStack;

int typeChecking(nodeType* root) {

    st_list_init(&symbolTableStack);

    paraList = NULL;
    paraCount = 0;
    int ret = typeCheckingSubTree(root);
    clearParaList();

    if (ret == 0) {
        printf("Type checking succeeds.\n");
    } else {
        printf("Type checking fails.\n");
        freeTree(root);
    }
    return ret;
}



int typeCheckingSubTree(nodeType* node) {
    if (node->type == OPERATOR_NODE) {
        if(typeCheckingOpNode(&node->opr)) {
            return 1;
        }
    }
    return 0;
}

int typeCheckingOpNode(oprNode* opr) {
    int i;
```

```
nodeType* temp = 0;
char* idName = 0;
struct identifier* id = 0;
struct sym_table* newSymbolTable = 0;
switch (opr->opType) {

    case parentheses_exp:
        temp = opr->op[0];
        /*
         * Now temp is a conditional-expression node.
         *      It can be various operator node;
         *      It can be ID node;
         *      It can be const node;
         */
        if (temp->type == OPERATOR_NODE) {
            if (typeCheckingOpNode (&(temp->opr)))
                return 1;
        }

        else if (temp->type == ID_NODE) {
            idName = idGen(opr->op[0]);
            id = sym_table_find(symbolTableStack.first->st, idName);
            if (!id) {
                printf ("Identifier not found: %s\n", idName);
                return 1;
            }
        }

        break;

    case array:
        // else if (opr->op[0]->type == ID_NODE) {
        idName = idGen(opr->op[0]);
        id = sym_table_find(symbolTableStack.first->st, idName);
        if (!id) {
            printf ("Identifier not found: %s\n", idName);
            return 1;
        }

        // one index
        if (opr->numOperands == 2) {
```

```
    temp = opr->op[1];
    /*
     * Now temp is a conditional-expression node.
     *      It can be various operator node;
     *      It can be ID node;
     *      It can be const node;
     */
    if (temp->type == OPERATOR_NODE) {
        if (typeCheckingOpNode (&(temp->opr)))
            return 1;
    }

    else if (temp->type == ID_NODE) {
        idName = idGen(opr->op[0]);
        id = sym_table_find(symbolTableStack.first->st, idName);
        if (!id) {
            printf ("Identifier not found: %s\n", idName);
            return 1;
        }
    }
}

// two index, representing a range
else {
    temp = opr->op[1];
    /*
     * Now temp is a conditional-expression node.
     *      It can be various operator node;
     *      It can be ID node;
     *      It can be const node;
     */
    if (temp->type == OPERATOR_NODE) {
        if (typeCheckingOpNode (&(temp->opr)))
            return 1;
    }

    else if (temp->type == ID_NODE) {
        idName = idGen(opr->op[0]);
        id = sym_table_find(symbolTableStack.first->st, idName);
        if (!id) {
            printf ("Identifier not found: %s\n", idName);
```

```
                return 1;
            }
        }

        temp = opr->op[2];
        /*
         * Now temp is a conditional-expression node.
         *      It can be various operator node;
         *      It can be ID node;
         *      It can be const node;
         */
        if (temp->type == OPERATOR_NODE) {
            if (typeCheckingOpNode (&(temp->opr)))
                return 1;
        }

        else if (temp->type == ID_NODE) {
            idName = idGen(opr->op[0]);
            id = sym_table_find(symbolTableStack.first->st, idName);
            if (!id) {
                printf ("Identifier not found: %s\n", idName);
                return 1;
            }
        }

    }

    break;

case func_call:
    idName = idGen(opr->op[0]);

    // no symbol table loaded
    if (!symbolTableStack.first) {
        printf("Function not defined: %s\n.", idName);
        return 1;
    }

    // id not found
    id = sym_table_find(symbolTableStack.last->st, idName);
    if (!id) {
```

```
            printf("Function not defined: %s\n.", idName);
            return 1;
        }

        if (!id->isFunc) {
            printf("%s is not a function.", idName);
            return 1;
        }

        // there are arguments
        if (opr->numOperands == 2) {
            temp = opr->op[1]; // now temp is an argument expression list

            int numArguments = 1;
            while (temp->type == OPERATOR_NODE && temp->opr.opType == concatenate) {
                numArguments += 1;
                temp = temp->opr.op[0];
            }

            if (id->numArgs != numArguments) {
                printf("Function arguments do not match: %s\n", idName);
                return 1;
            }
        }

        break;

    case math_op:
        // iterate the node's operands (children)
        for (i=0; i < opr->numOperands; i++) {
            temp = opr->op[i];
            if (temp->type == MATHOP_NODE)
                continue;

            if (temp->type == OPERATOR_NODE && temp->opr.opType == math_op) {
                if (typeCheckingOpNode(&(temp->opr)))
                    return 1;
            }

            else if (temp->type == OPERATOR_NODE) {
                printf("Math operation on illegal operands.\n");
```

```
            return 1;
        }

        else if (temp->type == ID_NODE) {
            idName = idGen(opr->op[i]);
            id = sym_table_find(symbolTableStack.first->st, idName);
            if (!id) {
                printf("Identifier not found: %s\n", idName);
                return 1;
            }

            if (id->type != INT_TYPE && id->type != INDEX_TYPE && id->type != BOOL_TYPE
                printf("Math operation can only be applied to int, index or bool type:
                return 1;
            }
        }

        else if (opr->op[i]->type != INTCON_NODE || opr->op[i]->type != BOOLCON_NODE )
            printf("Math operation can only be applied to int, index or bool type.\n")
            return 1;
        }
    }
    break;

case len:
    if (opr->op[0]->type != ID_NODE) {
        printf ("len operator can only be applied to an array.\n");
        return 1;
    }

    else {
        idName = idGen(opr->op[0]);
        id = sym_table_find(symbolTableStack.first->st, idName);

        if (!id) {
            printf("Identifier not found: %s\n", idName);
            return 1;
        }

        if (! (id->isArray) ) {
            printf ("len operator can only be applied to an array.\n");
```

```
            return 1;
        }
    }

    break;

case assignment:
    temp = opr->op[0];
    /*
     * Now temp could be postfix-expression or declarator.
     * 1. If it is postfix-expression:
     *      It can be an identifier node;
     *      It can be an operator node with array operation
     *      It CANNOT be anything else, including const node, other kinds of operator n
     *
     * 2. If it is declarator:
     *      It can be an identifier node;
     *      It can be an operator node with arr_decl operation, but if so, the first op
     */

    // It is an operator node, with array operation
    if (temp->type == OPERATOR_NODE && temp->opr.opType == array) {
        if (typeCheckingOpNode(&(temp->opr)))
            return 1;
    }

    // It is an operator node, with arr_decl operation
    if (temp->type == OPERATOR_NODE && temp->opr.opType == arr_decl) {
        temp = temp->opr.op[0];
        idName = idGen(temp);
        id = sym_table_find(symbolTableStack.first->st, idName);
        if(!id) {
            printf ("Identifier not found: %s\n", idName);
            return 1;
        }
    }

    // It is an identifier node
    else if (temp->type == ID_NODE) {
        idName = idGen(temp);
        id = sym_table_find(symbolTableStack.first->st, idName);
```

```
        if(!id) {
            printf ("Identifier not found: %s\n", idName);
            return 1;
        }
    }

    else {
        printf("Illegal assignment.\n");
        return 1;
    }

    if (typeCheckingSubTree(opr->op[1])) return 1;
    break;

case disp_exp:
    temp = opr->op[0];
    idName = idGen(temp);

    id = sym_table_find(symbolTableStack.first->st, idName);
    if(!id) {
        printf ("Identifier not found: %s\n", idName);
        return 1;
    }

    if (id->type == STRING_TYPE) {
        printf ("Cannot display string type: %s\n", idName);
        return 1;
    }

    break;

case hide_exp:
    temp = opr->op[0];
    idName = idGen(temp);

    id = sym_table_find(symbolTableStack.first->st, idName);
    if(!id) {
        printf ("Identifier not found: %s\n", idName);
        return 1;
    }
    break;
```

```
case swap:
    temp = opr->op[0];
    if (temp->type != ID_NODE) {
        printf("Swap can only be applied to an array.\n");
        return 1;
    }

    idName = idGen(temp);
    id = sym_table_find(symbolTableStack.first->st, idName);
    if (!id) {
        printf ("Identifier not found: %s\n", idName);
        return 1;
    }

    if (!id->isArray) {
        printf("Swap can only be applied to an array.\n");
        return 1;
    }

    // index one
    temp = opr->op[1];
    if (temp->type == OPERATOR_NODE) {
        if (temp->opr.opType != math_op) {
            printf ("Illegal index.\n");
            return 1;
        }
        if (typeCheckingOpNode(&temp->opr)) return 1;
    }
    else if (temp->type == ID_NODE) {
        idName = idGen(temp);
        id = sym_table_find(symbolTableStack.first->st, idName);
        if (!id) {
            printf("Identifier not found: %s\n", idName);
            return 1;
        }
        if (id->type != INT_TYPE && id->type != INDEX_TYPE) {
            printf("Index can only be int type or index type: %s\n", idName);
            return 1;
        }
    }
```

```
else if (temp->type != INTCON_NODE) {
    printf ("Illegal index.\n");
    return 1;
}

// index two
else if (temp->type != INTCON_NODE ) {
    printf("Index can only be int type or index type: %s\n", idName);
    return 1;
}

temp = opr->op[2];
if (temp->type == OPERATOR_NODE) {
    if (temp->opr.opType != math_op) {
        printf ("Illegal index.\n");
        return 1;
    }
    if (typeCheckingOpNode(&temp->opr)) return 1;
}

else if (temp->type == ID_NODE) {
    idName = idGen(temp);
    id = sym_table_find(symbolTableStack.first->st, idName);
    if (!id) {
        printf("Identifier not found: %s\n", idName);
        return 1;
    }
    if (id->type != INT_TYPE && id->type != INDEX_TYPE) {
        printf("Index can only be int or index: %s\n", idName);
        return 1;
    }
}

else if (temp->type != INTCON_NODE) {
    printf ("Illegal index.\n");
    return 1;
}

break;
```

```
case print:
    if (typeCheckingSubTree(opr->op[0])) return 1;
    break;

case print_list:
    temp = opr->op[0];
    if (temp->type == OPERATOR_NODE && temp->opr.opType == print_list) {
        if (typeCheckingOpNode(&temp->opr))
            return 1;
    }

    else if (temp->type == OPERATOR_NODE) {
        if (typeCheckingOpNode(&temp->opr))
                return 1;
    }

    else if (temp->type == ID_NODE) {
        idName = idGen(temp);
        id = sym_table_find(symbolTableStack.first->st, idName);
        if(!id) {
            printf ("Identifier not found: %s\n", idName);
            return 1;
        }
    }

    temp = opr->op[2];
    idName = idGen(temp);
    id = sym_table_find(symbolTableStack.first->st, idName);
    if(!id) {
        printf ("Identifier not found: %s\n", idName);
        return 1;
    }

    break;

case var_decl:
case var_decl_disp:
case var_decl_hide:

    // whether is an array
    temp = opr->op[1];
```

```
while (temp->type == OPERATOR_NODE) {
    if (temp->opr.opType == arr_decl) break;
    temp = temp->opr.op[0];
}

if (temp->opr.opType == arr_decl) { // is array
    if (temp->opr.op[0]->type != ID_NODE) {
        printf("Illegal array declaration!\n");
        return 1;
    }
    idName = idGen(temp->opr.op[0]);
    id = sym_table_find(symbolTableStack.first->st, idName);
    if (id) {
        printf("Redefinition of %s\n.", idName);
        return 1;
    }

    id->name = idName;
    id->type = typeGen(opr->op[0]);
    id->isArray = 1;
    id->isFunc = 0;
    id->numArgs = 0;
    id->args = 0;
    id->argsIsArray = 0;

    sym_table_add(symbolTableStack.first->st, id);
}

else { // not array
    idName = idGen(temp);
    id = sym_table_find(symbolTableStack.first->st, idName);
    if (id) {
        printf("Redefinition of %s\n", idName);
        return 1;
    }
    id->name = idName;
    id->type = typeGen(opr->op[0]);
    id->isArray = 0;
    id->isFunc = 0;
    id->numArgs = 0;
    id->args = 0;
```

```
            id->argsIsArray = 0;

            sym_table_add(symbolTableStack.first->st, id);

        }
        // assignment
        if (opr->op[1]->type == OPERATOR_NODE && opr->op[1]->opr.opType == assignment) {
            temp = opr->op[1]->opr.op[1];
            if (temp->type == ID_NODE) {
                idName = idGen(temp);
                id = sym_table_find(symbolTableStack.first->st, idName);
                if (!id) {
                    printf("Identifier not found: %s\n", idName);
                    return 1;
                }
            }
            else if (temp->type == OPERATOR_NODE) {
                if (typeCheckingOpNode(&temp->opr))
                        return 1;
            }
        }
        break;

    case init_list:
        temp = opr->op[0];
        if (temp->type == ID_NODE) {
            idName = idGen(temp);
            id = sym_table_find(symbolTableStack.first->st, idName);
            if (!id) {
                printf("Identifier not found: %s\n", idName);
                return 1;
            }
        }

        else if (temp->type == OPERATOR_NODE && temp->opr.opType == concatenate) {
            idName = idGen(temp->opr.op[1]);
            id = sym_table_find(symbolTableStack.first->st, idName);
            if (!id) {
                printf("Identifier not found: %s\n", idName);
                return 1;
            }
```

```
            if (typeCheckingOpNode(&(temp->opr.op[0]->opr)))
                return 1;
        }

        if (typeCheckingSubTree(opr->op[0])) return 1;
        break;

    case exp_state:
        temp = opr->op[0];
        if (temp->type == ID_NODE) {
            idName = idGen(temp->opr.op[1]);
            id = sym_table_find(symbolTableStack.first->st, idName);
            if (!id) {
                printf("Identifier not found: %s\n", idName);
                return 1;
            }
        }

        if (temp->type == OPERATOR_NODE) {
            if (typeCheckingOpNode(&temp->opr))
                return 1;
        }

        break;

    case declar_state:
        if (typeCheckingSubTree(opr->op[0])) return 1;
        break;

    case comp_state:
        if (opr->numOperands == 1 && typeCheckingSubTree(opr->op[0])) return 1;
        break;

    case state_list:
        if (typeCheckingSubTree(opr->op[0])) return 1;
        if (typeCheckingSubTree(opr->op[1])) return 1;
        break;

    case select_state:
        temp = opr->op[0];
```

```
if (temp->type == OPERATOR_NODE) {
    if (temp->opr.opType != math_op) {
        printf ("Illegal index.\n");
        return 1;
    }
    if (typeCheckingOpNode(&temp->opr)) return 1;

}

else if (temp->type == ID_NODE) {
    idName = idGen(temp);
    id = sym_table_find(symbolTableStack.first->st, idName);
    if (!id) {
        printf ("Identifier not found: %s\n", idName);
        return 1;
    }
    if (id->type != BOOL_TYPE && id->type != INT_TYPE) {
        printf("Illegal if condition\n");
        return 1;
    }
}

else if (temp->type != INTCON_NODE && temp->type != BOOLCON_NODE) {
    printf("Illegal if condition\n");
    return 1;
}

// generate new symbol table for the new scope
newSymbolTable = (struct sym_table*) malloc(sizeof(struct sym_table));
sym_table_init(newSymbolTable);
st_list_add(&symbolTableStack, newSymbolTable);

if (typeCheckingSubTree(opr->op[1])) return 1;

// remove the top symbol table
st_list_del(&symbolTableStack);

if (opr->numOperands == 3) {
    // generate new symbol table for the new scope
    newSymbolTable = (struct sym_table*) malloc(sizeof(struct sym_table));
    sym_table_init(newSymbolTable);
```

```
        st_list_add(&symbolTableStack, newSymbolTable);

        if (typeCheckingSubTree(opr->op[2])) return 1;

        // remove the top symbol table
        st_list_del(&symbolTableStack);
    }
    break;

case while_state:
    temp = opr->op[0];
    if (temp->type == OPERATOR_NODE) {
        if (temp->opr.opType != math_op) {
            printf ("Illegal index.\n");
            return 1;
        }
        if (typeCheckingOpNode(&temp->opr)) return 1;

    }

    else if (temp->type == ID_NODE) {
        idName = idGen(temp);
        id = sym_table_find(symbolTableStack.first->st, idName);
        if (!id) {
            printf ("Identifier not found: %s\n", idName);
            return 1;
        }
        if (id->type != BOOL_TYPE && id->type != INT_TYPE) {
            printf("Illegal while condition\n");
            return 1;
        }
    }

    else if (temp->type != INTCON_NODE && temp->type != BOOLCON_NODE) {
        printf("Illegal while condition\n");
        return 1;
    }

    // generate new symbol table for the new scope
    newSymbolTable = (struct sym_table*) malloc(sizeof(struct sym_table));
    sym_table_init(newSymbolTable);
```

```
    st_list_add(&symbolTableStack, newSymbolTable);

    if (typeCheckingSubTree(opr->op[1])) return 1;

    // remove the top symbol table
    st_list_del(&symbolTableStack);

    break;

case do_while_state:
    // generate new symbol table for the new scope
    newSymbolTable = (struct sym_table*) malloc(sizeof(struct sym_table));
    sym_table_init(newSymbolTable);
    st_list_add(&symbolTableStack, newSymbolTable);

    if (typeCheckingSubTree(opr->op[0])) return 1;

    // remove the top symbol table
    st_list_del(&symbolTableStack);

    temp = opr->op[1];
    if (temp->type == OPERATOR_NODE) {
        if (temp->opr.opType != math_op) {
            printf ("Illegal index.\n");
            return 1;
        }
        if (typeCheckingOpNode(&temp->opr)) return 1;

    }

    else if (temp->type == ID_NODE) {
        idName = idGen(temp);
        id = sym_table_find(symbolTableStack.first->st, idName);
        if (!id) {
            printf ("Identifier not found: %s\n", idName);
            return 1;
        }
        if (id->type != BOOL_TYPE && id->type != INT_TYPE) {
            printf("Illegal while condition\n");
            return 1;
        }
```

```
    }

    else if (temp->type != INTCON_NODE && temp->type != BOOLCON_NODE) {
        printf("Illegal while condition\n");
        return 1;
    }

    break;

case for_state:
    if (opr->op[0]->opr.opType == var_decl ||opr->op[0]->opr.opType == var_decl_disp |
        // generate new symbol table for the new scope
        newSymbolTable = (struct sym_table*) malloc(sizeof(struct sym_table));
        sym_table_init(newSymbolTable);
        st_list_add(&symbolTableStack, newSymbolTable);

        if (typeCheckingSubTree(opr->op[0])) return 1;

        temp = opr->op[1];
        if (temp->type == OPERATOR_NODE) {
            if (temp->opr.opType != math_op) {
                printf ("Illegal index.\n");
                return 1;
            }
            if (typeCheckingOpNode(&temp->opr)) return 1;

        }

        else if (temp->type == ID_NODE) {
            idName = idGen(temp);
            id = sym_table_find(symbolTableStack.first->st, idName);
            if (!id) {
                printf ("Identifier not found: %s\n", idName);
                return 1;
            }
            if (id->type != BOOL_TYPE && id->type != INT_TYPE) {
                printf("Illegal for condition\n");
                return 1;
            }
        }
```

```
    else if (temp->type != INTCON_NODE && temp->type != BOOLCON_NODE) {
        printf("Illegal for condition\n");
        return 1;
    }

    if (opr->numOperands == 4 && typeCheckingSubTree(opr->op[2])) return 1;
    if (typeCheckingSubTree(opr->op[opr->numOperands-1])) return 1;

    // remove the top symbol table
    st_list_del(&symbolTableStack);
}

else {
    if (typeCheckingSubTree(opr->op[0])) return 1;

    temp = opr->op[1];
    if (temp->type == OPERATOR_NODE) {
        if (temp->opr.opType != math_op) {
            printf ("Illegal index.\n");
            return 1;
        }
        if (typeCheckingOpNode(&temp->opr)) return 1;

    }

    else if (temp->type == ID_NODE) {
        idName = idGen(temp);
        id = sym_table_find(symbolTableStack.first->st, idName);
        if (!id) {
            printf ("Identifier not found: %s\n", idName);
            return 1;
        }
        if (id->type != BOOL_TYPE && id->type != INT_TYPE) {
            printf("Illegal for condition\n");
            return 1;
        }
    }

    else if (temp->type != INTCON_NODE && temp->type != BOOLCON_NODE) {
        printf("Illegal for condition\n");
        return 1;
```

```
        }

        if (opr->numOperands == 4 && typeCheckingSubTree(opr->op[2])) return 1;

        // generate new symbol table for the new scope
        newSymbolTable = (struct sym_table*) malloc(sizeof(struct sym_table));
        sym_table_init(newSymbolTable);
        st_list_add(&symbolTableStack, newSymbolTable);

        if (typeCheckingSubTree(opr->op[opr->numOperands-1])) return 1;

        // remove the top symbol table
        st_list_del(&symbolTableStack);
    }
    break;

case jump_continue_state:
    break;

case jump_break_state:
    st_list_del(&symbolTableStack);
    break;

case jump_ret_state:
    if (opr->numOperands == 1 && typeCheckingSubTree(opr->op[0])) return 1;
    st_list_del(&symbolTableStack);
    break;

case trans_unit:
    if (typeCheckingSubTree(opr->op[0])) return 1;
    if (typeCheckingSubTree(opr->op[1])) return 1;
    break;

case func_def:
    // if there is not one symbol table in the stack, create one as the base
    if (!symbolTableStack.first) {
        // generate new symbol table for the new scope
        newSymbolTable = (struct sym_table*) malloc(sizeof(struct sym_table));
        sym_table_init(newSymbolTable);
        st_list_add(&symbolTableStack, newSymbolTable);
    }
```

```
varTypeEnum returnType = typeGen(opr->op[0]); // function return type
idName = idGen(opr->op[1]); // function name

id = sym_table_find(symbolTableStack.last->st, idName);
if (id) {
    printf("Function redefinition: %s\n", idName);
    return 1;
}

if (strcmp(opr->op[1]->id.value,"main") == 0)
    mainFun = 1;
else
    mainFun = 0;

// clear the parameter list
clearParaList();
paraCount = 0;

// whether main function or not, a new scope is created.
// generate new symbol table for the new scope
newSymbolTable = (struct sym_table*) malloc(sizeof(struct sym_table));
sym_table_init(newSymbolTable);
st_list_add(&symbolTableStack, newSymbolTable);

if (mainFun) {
    if(returnType != VOID_TYPE && returnType != INT_TYPE) {
        printf("Main function must return int or void.\n");
        return 1;
    }

    if (opr->numOperands == 4) {
        printf("Main function must not take any arguments.\n");
        return 1;
    }

    id->name = idName;
    id->type = returnType;
    id->isArray = 0;
    id->isFunc = 1;
    id->numArgs = 0;
```

```
                id->args = 0;
                id->argsIsArray = 0;

                sym_table_add(symbolTableStack.first->st, id);

            }
            else if (opr->numOperands == 4) {
                getParaList(opr->op[2]);

                id->name = idName;
                id->type = returnType;
                id->isArray = 0;
                id->isFunc = 0;
                id->numArgs = paraCount;
                id->args = (varTypeEnum*) malloc(sizeof(varTypeEnum));
                id->argsIsArray = (int*) malloc(sizeof(varTypeEnum));
                for (int paraIndex = 0; paraIndex < paraCount; paraIndex ++) {
                    id->args[i] = paraList[i]->type;
                    id->argsIsArray[i] = paraList[i]->isArray;
                }

                sym_table_add(symbolTableStack.first->st, id);
            }

            if (typeCheckingSubTree(opr->op[opr->numOperands-1])) return 1;

            // remove the top symbol table
            st_list_del(&symbolTableStack);
            break;

        default:
            break;
    }
    return 0;
}

void clearParaList() {
    int i = 0;
    if (!paraList) return;
    for (; i < paraCount; i ++) {
        struct identifier* tmp = paraList[i];
```

```
        if (!tmp) break;
        if (tmp->args) free(tmp->args);
        if(tmp->argsIsArray) free(tmp->argsIsArray);
        free(tmp);
    }
    free (paraList);
}

varTypeEnum typeGen(nodeType* node) {
    if (node->type != VARTYPE_NODE)
        return -1;

    switch (node->varType.value) {
        case VOID_TYPE:
            return VOID_TYPE;
            break;
        case CHAR_TYPE:
            return CHAR_TYPE;
            break;
        case INT_TYPE:
            return INT_TYPE;
            break;
        case STRING_TYPE:
            return STRING_TYPE;
            break;
        case INDEX_TYPE:
            return INDEX_TYPE;
            break;
        case BOOL_TYPE:
            return BOOL_TYPE;
            break;
    }
}

char* idGen(nodeType* node) {
    if (node->type == ID_NODE)
        return node->id.value;
    else
        printf ("Shining");
    return 0;
}
```

```
int getNumPara(oprNode* node) {
    if (node->opType == concatenate)
        return getNumPara(&node->op[0]->opr) + 1;
    return 1;
}

void getParaList(nodeType* node) {
    if (node->type != OPERATOR_NODE)
        return ;
    paraCount = getNumPara(&node->opr);

    nodeType* tmp = node;
    nodeType ** para_decl = (nodeType **) malloc (paraCount * sizeof (nodeType *));
    if (paraCount == 1) {
        para_decl[0] = node;
    }

    else {
        int i =0;
        while (tmp->opr.opType == concatenate) {
            para_decl[i] = tmp->opr.op[1];
            i ++;
            tmp = tmp->opr.op[0];
        }
    }

    clearParaList();
    paraList = (struct identifier**) malloc(paraCount * sizeof(struct identifier*));
    for (int i = 0; i < paraCount; i ++) {
        struct identifier *tmp = (struct identifier*) malloc(sizeof(struct identifier));

        nodeType* nodeTmp = para_decl[i];
        nodeType* temp1 = nodeTmp->opr.op[0];
        nodeType* temp2 = nodeTmp->opr.op[1];

        tmp->type = typeGen(temp1);
        if (temp2->type == ID_NODE) { // an identifier
            tmp->name = idGen(temp2);
            tmp->isArray = 0;
            tmp->isFunc = 0;
```

```
            tmp->numArgs = 0;
            tmp->args = 0;
            tmp->argsIsArray = 0;
        }

        else { // an array
            tmp->name = idGen(temp2->opr.op[0]);
            tmp->isArray = 1;
            tmp->isFunc = 0;
            tmp->numArgs = 0;
            tmp->args = 0;
            tmp->argsIsArray = 0;
        }

        free (nodeTmp);

        paraList[i] = tmp;
    }
    free (para_decl);
}
```

```
file: avl/tests/avl.test/.gitignore

constant
declaration_bool
declaration_bool_array
declaration_bool_array.avl~
declaration_char
declaration_index
declaration_int
declaration_string
display_bool
display_bool_array
display_char_array
display_int
display_int_array
display_string
expression_bool
```

```
expression_char
expression_char_array
expression_index
expression_int
expression_int_array
expression_string
insertion_sort
Makefile
Makefile.am
Makefile.in
operator_char_post
operator_char_unary
operator_int_post
operator_int_unary
operator_string_mid
statement_empty
statement_if
```

file: avl/tests/avl.test/cast.avl

```
int main(){

    int a = 48;
    char c = 'a';
    char b = (char) a;
    int d = (int) c;
    print b;
    print d;


}
```

file: avl/tests/avl.test/cast.sh

```
#!/bin/bash
avl -o cast cast.avl
```

file: avl/tests/avl.test/constant.avl

```
int main() {

    1;
    "hello world";
    "";
    'c';

    (1);
    ("hello world");
    ("");
    ('c');

    print 1, "hello world", 's', "";
    return 0;
}
```

file: avl/tests/avl.test/constant.sh

```
#!/bin/bash
avl -o constant constant.avl
```

file: avl/tests/avl.test/declaration_bool.avl

```
int main(){

    bool b;
        bool b1 = true;
        display bool b2 = true;
        hide bool b3 = true;
        display bool b4;
        display bool b5;
    return 0;
}
```

file: avl/tests/avl.test/declaration_bool.sh

```
#!/bin/bash
avl -o declaration_bool declaration_bool.avl
```

file: avl/tests/avl.test/declaration_bool_array.avl

```
int main(){

    bool b6[5];
        display bool b7[5];
        hide bool b8[6];
        bool b9[3] = {true, true, false};

    return 0;
}
```

file: avl/tests/avl.test/declaration_bool_array.sh

```
#!/bin/bash
avl -o declaration_bool_array declaration_bool_array.avl
```

file: avl/tests/avl.test/declaration_char.avl

```
int main(){

    char c;
        char c1 = 'c';
        display char c2 = 'c';
        hide char c3 = 'c';
        display char c4;
        hide char c5;

    return 0;
}
```

file: avl/tests/avl.test/declaration_char.sh

```
#!/bin/bash
avl -o declaration_char declaration_char.avl
```

file: avl/tests/avl.test/declaration_char_array.avl

```
int main(){

    char c6[5];
        /*char c7[] = "hello world";*/
        display char c8[5];
        hide char c9[6];
        /*char c10[6] = "hello";*/
        char c12[5] = {'i','h','e','l','l'};

    return 0;
}
```

file: avl/tests/avl.test/declaration_char_array.sh

```
#!/bin/bash
avl -o declaration_char_array declaration_char_array.avl
```

file: avl/tests/avl.test/declaration_index.avl

```
int main(){

    index in1;
    index in2 = 1;
    display index in3 = 1;
    hide index in4 = 1;
    display index in5;
    hide index in6;
```

```
    return 0;
}
```

file: avl/tests/avl.test/declaration_index.sh

```
#!/bin/bash
avl -o declaration_index declaration_index.avl
```

file: avl/tests/avl.test/declaration_int.avl

```
int main(){


    int i;
        int i1 = 1;
        display int i2 = 1;
        hide int i3 = 1;
        display int i4;
        hide i5;

    return 0;
}
```

file: avl/tests/avl.test/declaration_int.sh

```
#!/bin/bash
avl -o declaration_int declaration_int.avl
```

file: avl/tests/avl.test/declaration_int_array.avl

```
int main(){

    int i6[5];
        display int i7[5];
        hide int i8[6];
```

```
        int i9[3] = {1,2,3};

    return 0;

}
```

file: avl/tests/avl.test/declaration_int_array.sh

```
#!/bin/bash
avl -o declaration_int_array declaration_int_array.avl
```

file: avl/tests/avl.test/declaration_string.avl

```
int main(){

    string s;
        string s1 = "hello world!";
        /*display string s2 = "hello world!";
        hide string s3 = "hello world!";
        display string s4;
        hide string s5;*/

    return 0;
}
```

file: avl/tests/avl.test/declaration_string.sh

```
#!/bin/bash
avl -o declaration_string declaration_string.avl
```

file: avl/tests/avl.test/display_bool.avl

```
int main(){

    display bool b  = true;
```

```
    <begin_display>
        b;
    <end_display>

    hide b;

    <begin_display>
        b;
    <end_display>

    return 0;
}
```

file: avl/tests/avl.test/display_bool.sh

```
#!/bin/bash
avl -o display_bool display_bool.avl
```

file: avl/tests/avl.test/display_bool_array.avl

```
int main(){

    display bool array[5] = {true,true,false,false,false};
    <begin_display>
        swap(array,1,2);
    <end_display>

    hide array;
    <begin_display>
        swap(array,1,2);
    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/display_bool_array.sh

```
#!/bin/bash
avl -o display_bool_array display_bool_array.avl
```

file: avl/tests/avl.test/display_char.avl

```
int main(){

    display char c = 'd';
    <begin_display>
        c;
    <end_display>

    hide c;
    <begin_display>
        c;
    <end_display>

    return 0;
}
```

file: avl/tests/avl.test/display_char.sh

```
#!/bin/bash
avl -o display_char display_char.avl
```

file: avl/tests/avl.test/display_char_array.avl

```
int main(){

    display char array[5] = {'1','2','3','4','5'};
    <begin_display>
        swap(array,1,2);
    <end_display>

    hide array;
    <begin_display>
```

```
        swap(array,1,2);
    <end_display>
    return 0;

}
```

```
file: avl/tests/avl.test/display_char_array.sh

#!/bin/bash
avl -o display_char_array display_char_array.avl
```

```
file: avl/tests/avl.test/display_index.avl

int main(){

    display int array[5] = {1,2,3,4,5};
    display index i = 1;
    display index j = 2;
    <begin_display>
        array[i];
        array[j];

        swap(array,i,j);
    <end_display>

    hide array;


    <begin_display>
        swap(array,i,j);
    <end_display>

    hide i;
    hide j;


    <begin_display>
        swap(array,i,j);
```

```
    <end_display>


    display array;
    <begin_display>
        swap(array,i,j);
    <end_display>


    return 0;
}



file: avl/tests/avl.test/display_index.sh

#!/bin/bash
avl -o display_index display_index.avl


file: avl/tests/avl.test/display_int.avl

int main(){

    display int i = 1;
    <begin_display>
        i;
    <end_display>

    hide i;
    <begin_display>
        i;
    <end_display>

    return 0;
}



file: avl/tests/avl.test/display_int.sh
```

```
#!/bin/bash
avl -o display_int display_int.avl
```

file: avl/tests/avl.test/display_int_array.avl

```
int main(){

    display int array[5] = {1,2,3,4,5};
    <begin_display>
        swap(array,1,2);
    <end_display>

    hide array;
    <begin_display>
        swap(array,1,2);
    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/display_int_array.sh

```
#!/bin/bash
avl -o display_int_array display_int_array.avl
```

file: avl/tests/avl.test/display_string.avl

```
int main(){

    display string s = "hello world";
    <begin_display>
        s;
    <end_display>
    hide s;

    <begin_display>
        s;
    <end_display>
```

```
    return 0;
}
```

file: avl/tests/avl.test/display_string.sh

```
#!/bin/bash
avl -o display_string display_string.avl
```

file: avl/tests/avl.test/empty.avl

file: avl/tests/avl.test/empty.sh

```
#!/bin/bash
avl -t empty.avl
```

file: avl/tests/avl.test/expression_bool.avl

```
int main(){

    bool b = true;
    print b;
    b  = false;

    hide b;
    display b;

    print b;

}
```

file: avl/tests/avl.test/expression_bool.sh

```
#!/bin/bash
avl -o expression_bool expression_bool.avl
```

file: avl/tests/avl.test/expression_bool_array.avl

```
int main(){

    bool array[5] = {true, true, false, false, false};
    swap(array,1,2);
    hide array;
    display array;
    <begin_display>
        swap(array,1,2);
        array[4] = true;
    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/expression_bool_array.sh

```
#!/bin/bash
avl -o expression_bool_array expression_bool_array.avl
```

file: avl/tests/avl.test/expression_char.avl

```
int main(){

    char c = 'c';
    print c;
    c = 'd';
    print c;

    hide c;
    display c;
    return 0;
```

```
}
```

file: avl/tests/avl.test/expression_char.sh

```
#!/bin/bash
avl -o expression_char expression_char.avl
```

file: avl/tests/avl.test/expression_char_array.avl

```
int main(){

    char array[5] = {'a','b','c','d','e'};
    swap(array,1,2);
    hide array;
    display array;
    <begin_display>
        swap(array,1,2);
        array[4] = 'f';
    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/expression_char_array.sh

```
#!/bin/bash
avl -o expression_char_array expression_char_array.avl
```

file: avl/tests/avl.test/expression_index.avl

```
int main(){

    index i = 1;
    i = 2;

    hide i;
```

```
    display i;
    print i;
    return 0;

}
```

file: avl/tests/avl.test/expression_index.sh

```
#!/bin/bash
avl -o expression_index expression_index.avl
```

file: avl/tests/avl.test/expression_int.avl

```
int main(){

    int a = 1;
    print a;
    a = 2;
    print a;
    hide a;
    display a;

    return 0;
}
```

file: avl/tests/avl.test/expression_int.sh

```
#!/bin/bash
avl -o expression_int expression_int.avl
```

file: avl/tests/avl.test/expression_int_array.avl

```
int main(){

    int array[5] = {1, 2, 3, 4, 5};
```

```
    swap(array,1,2);
    hide array;
    display array;
    <begin_display>
        swap(array,1,2);
        array[4] = 6;
    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/expression_int_array.sh

```
#!/bin/bash
avl -o expression_int_array expression_int_array.avl
```

file: avl/tests/avl.test/expression_string.avl

```
int main(){

    string s = "123";
    print s;
    s = "abc";
    print s;
    hide s;
    display s;

    return 0;
}
```

file: avl/tests/avl.test/expression_string.sh

```
#!/bin/bash
avl -o expression_string expression_string.avl
```

file: avl/tests/avl.test/function_bool.avl

```
bool test(bool x){
    x = !x;
    return x;

}

int main(){

    bool b = true;
    print test(b);
    print b;
    return 0;

}
```

file: avl/tests/avl.test/function_bool.sh

```
#!/bin/bash
avl -o function_bool function_bool.avl
```

file: avl/tests/avl.test/function_bool_array.avl

```
void test(bool x[]){

    index l = len(x);
    if( l <= 1)
        print x[0];
    else{
        print x[l-1];
        test(x[0:l-1]);
    }

    return;

}
```

```
int main(){

    bool array[] = {true,true,false,false,false};
    <begin_display>

        test(array);

    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/function_bool_array.sh

```
#!/bin/bash
avl -o function_bool_array function_bool_array.avl
```

file: avl/tests/avl.test/function_char.avl

```
char test(char x){

    x++;
    return x;

}

int main(){

    char c = 'd';
    print test(c);
    return 0;

}
```

file: avl/tests/avl.test/function_char.sh

```
#!/bin/bash
avl -o function_char function_char.avl
```

file: avl/tests/avl.test/function_char_array.avl

```
void test(char x[]){

    index l = len(x);
    if( l <= 1)
        print x[0];
    else{
        print x[l-1];
        return test(x[0:l-1]);
    }

    return;

}

int main(){

    char array[] = {'1','2','3','4','5'};
    <begin_display>

        test(array);

    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/function_char_array.sh

```
#!/bin/bash
avl -o function_char_array function_char_array.avl
```

file: avl/tests/avl.test/function_empty.avl

```
void empty(){

}

int main(){

    empty();

}
```

file: avl/tests/avl.test/function_empty.sh

```
#!/bin/bash
avl -o function_empty function_empty.avl
```

file: avl/tests/avl.test/function_index.avl

```
index test(index x){

    x++;
    return x;

}

int main(){

    index i = 1;
    test(i);
    return 0;

}
```

file: avl/tests/avl.test/function_index.sh

```
#!/bin/bash
avl -o function_index function_index.avl
```

file: avl/tests/avl.test/function_int.avl

```
int test(int x){
    x++;
    return x;
}


int main(){

    int a = 1;
    print test(a);
    print a;
    return 0;
}
```

file: avl/tests/avl.test/function_int.sh

```
#!/bin/bash
avl -o function_int function_int.avl
```

file: avl/tests/avl.test/function_int_array.avl

```
void test(int x[]){
    index l = len(x);
    if( l <= 1 )
```

```
        print x[0];
    else{
        print x[l-1];
        test(x[0:l-1]);
    }

    return;

}

int main(){

    int array[] = {1,2,3,4,5};
    <begin_display>
        test(array);
    <end_display>
    return 0;

}
```

file: avl/tests/avl.test/function_int_array.sh

```
#!/bin/bash
avl -o function_int_array function_int_array.avl
```

file: avl/tests/avl.test/function_string.avl

```
string test(string x){

    x = "hello anthoer world";
    return x;
}

int main(){

    string s= "hello world";
    print test(s),s;
    return 0;
```

```
}
```

file: avl/tests/avl.test/function_string.sh

```
#!/bin/bash
avl -o function_string function_string.avl
```

file: avl/tests/avl.test/hello_world.avl

```
int main() {
    <begin_display>
    <begin_display>
    display char str[] = "Hello world!";
    display int a[] = {1, 2, 3, 4};
    <end_display>
    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/hello_world.sh

```
#!/bin/bash
avl -o hello_world hello_world.avl
```

file: avl/tests/avl.test/init_test.sh

```
#!/bin/bash
```

file: avl/tests/avl.test/insertion_sort.avl

```
int main() {
    display int a[] = {5, 2, 7, 3, 6, 8};
    <begin_display>
    for (display index i = 1; i < len(a); i = i + 1) {
        display index j = i - 1;
        while (j >= 0 && a[j] > a[j + 1]) {
            swap(a, j + 1, j);
            j = j - 1;
        }
    }
    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/insertion_sort.sh

```
#!/bin/bash
avl -o insertion_sort insertion_sort.avl
```

file: avl/tests/avl.test/kmp.avl

```
index[] get_overlap(char pattern[]) {
    <begin_display>
    display index result[len(pattern)];
    display index k = 0;
    for (index i = 1; i < len(pattern); i = i + 1) {
        while (k > 0 && pattern[k] != pattern[i])
            k = result[k - 1];
        if (pattern[k] == pattern[i]) {
            k = k + 1;
            result[i] = k;
        }
    }
    <end_display>
    return result;
}
```

```
index kmp(char target[], char pattern[], index overlap[]) {
    <begin_display>
    display index q = 0;
    for (index i = 0; i < len(target); i = i + 1) {
    while (q > 0 && pattern[q] != target[i])
        q = overlap[q - 1];
        if (pattern[q] == target[i])
            q = q + 1;
        if (q == len(pattern))
            return i - m + 1;
    }
    <end_display>
    return -1;
}

int main() {
    <begin_display>
    display char target[] = bananobano;
    display char pattern[] = nano;
    display int overlap[] = get_overlap(pattern);
    index result = kmp(target, pattern, overlap);
    <end_display>
    print(result);
}
```

file: avl/tests/avl.test/kmp.sh

```
#!/bin/bash
avl -o kmp kmp.avl
```

file: avl/tests/avl.test/Makefile.am

```
AM_TESTS_ENVIRONMENT = . $(srcdir)/init_test.sh;
AM_TESTS_FD_REDIRECT = 9>&2

TESTS = \
            cast.sh \
```

```
constant.sh \
declaration_bool.sh \
declaration_bool_array.sh \
declaration_char.sh \
declaration_char_array.sh \
declaration_index.sh \
declaration_int.sh \
declaration_int_array.sh \
declaration_string.sh \
display_bool.sh \
display_bool_array.sh \
display_char.sh \
display_char_array.sh \
display_index.sh \
display_int.sh \
display_int_array.sh \
display_string.sh \
empty.sh \
expression_bool.sh \
expression_bool_array.sh \
expression_char.sh \
expression_char_array.sh \
expression_index.sh \
expression_int.sh \
expression_int_array.sh \
expression_string.sh \
function_bool.sh \
function_bool_array.sh \
function_char.sh \
function_char_array.sh \
function_empty.sh \
function_index.sh \
function_int.sh \
function_int_array.sh \
function_string.sh \
hello_world.sh \
init_test.sh \
insertion_sort.sh \
kmp.sh \
operator_bool_mid.sh \
operator_bool_post.sh \
```

```
            operator_char_mid.sh \
            operator_char_post.sh \
            operator_char_unary.sh \
            operator_index_mid.sh \
            operator_index_post.sh \
            operator_index_unary.sh \
            operator_int_mid.sh \
            operator_int_post.sh \
            operator_int_unary.sh \
            operator_string_mid.sh \
            quick_sort.sh \
            statement_empty.sh \
            statement_for.sh \
            statement_if.sh \
            statement_while.sh \
            statement_scope.sh \
            subarray_int.sh \
            mergeSort.sh \
            maxSubarray.sh

XFAIL_TESTS = \
            cast.sh \
            statement_scope.sh \
            empty.sh
```

```
file: avl/tests/avl.test/maxSubarray.avl

void findMaxCrossSubarray(int A[], int mid, int cross[]) {
    int leftsum = 0;
    bool init = false;
    int sum = 0;
    int max_left = 0;

    int count = mid - cross[0];
    <begin_display>
    for (index i = mid - cross[0]; i >=0; i--) {
        sum = sum + A[i];
        if (!init || sum > leftsum) {
            leftsum = sum;
```

```
            max_left = count + cross[0];
            init = true;
        }
        count--;
    }
    <end_display>

    int rightsum = 0;
    init = false;
    sum = 0;
    int  max_right = 0;

    count = mid + 1 - cross[0];
    <begin_display>
    for (index j = mid + 1 - cross[0]; j < len(A); j++) {
        sum = sum + A[j];
        if (!init || sum > rightsum) {
            rightsum = sum;
            max_right = count + cross[0];
            init = true;
        }
        count++;
    }
    <end_display>

    cross[0] = max_left;
    cross[1] = max_right;
    cross[2] = leftsum + rightsum;
}

void findMaxSubarray(int A[], int ret[]) {
    if (ret[0] >= ret[1]) {
        ret[2] = A[0];
        return;
    }

    <begin_display>
    display index low = 0;
    display index high = len(A) - 1;
    display index mid = (low + high) / 2;
```

```
    int m = (ret[0] + ret[1]) / 2;

    int left[] = {0,0,0};
    left[0] = ret[0];
    left[1] = m;

    int right[] = {0,0,0};
    right[0] = m + 1;
    right[1] = ret[1];

    int cross[] = {0,0,0};
    cross[0] = ret[0];
    cross[1] = ret[1];

    findMaxSubarray(A[low : (mid + 1)], left);
    findMaxSubarray(A[(mid + 1) :  (high + 1)], right);
    findMaxCrossSubarray(A, m, cross);

    if (left[2] >= right[2] && left[2] >= cross[2]) {
        ret[0] = left[0];
        ret[1] = left[1];
        ret[2] = left[2];
    }
    else if (right[2] >= left[2] && right[2] >= cross[2]) {
        ret[0] = right[0];
        ret[1] = right[1];
        ret[2] = right[2];
    }
    else {
        ret[0] = cross[0];
        ret[1] = cross[1];
        ret[2] = cross[2];
    }
    <end_display>
}


int main() {
    display int A[] = {1,3,4,-8,-3,2,-1,0,4};
    int ret[] = {0,0,0};
    ret[1] = len(A) - 1;
```

```
    <begin_display>
    findMaxSubarray(A, ret);
    <end_display>

    index s = ret[0];
    index e = ret[1];

    <begin_display>
    display int B[] = A[s:(e+1)];
    <end_display>

    return 0;
}
```

file: avl/tests/avl.test/maxSubarray.sh

```
#!/bin/bash
avl -o maxSubarray maxSubarray.avl
```

file: avl/tests/avl.test/mergeSort.avl

```
void mergeSort(int A[]) {
    if (len(A) <= 1)
        return;

    display index s = 0;
    display index e = len(A) - 1;
    display index m = (s + e) / 2;

    <begin_display>
    mergeSort(A[s:(m+1)]);
    mergeSort(A[(m+1):(e+1)]);
    <end_display>

    int tmp[] = A[s:(m+1)];
    display int L[len(tmp)];
```

```
for (index i = 0; i < len(tmp); i++) {
    L[i] = tmp[i];
}

tmp = A[(m+1):(e+1)];
display int R[len(tmp)];
for (index i = 0; i < len(tmp); i++) {
    R[i] = tmp[i];
}

display index l = 0;
display index r = 0;

<begin_display>
for (index i = s; i <= e; i++) {
    if (l == m - s + 1) {
        while (r < e - m) {
            A[i] = R[r];
            ++i;
            ++r;
        }
        break;
    }

    if (r == e - m) {
        while (l < m - s + 1) {
            A[i] = L[l];
            ++i;
            ++l;
        }
        break;
    }

    if (L[l] <= R[r]) {
        A[i] = L[l];
        ++l;
    } else {
        A[i] = R[r];
        ++r;
    }
}
```

```
    <end_display>
}



int main() {
    <begin_display>
    display int A[] = {2, 4, 1, 10, 8, 7, 5, 5, 3, 3};
    mergeSort(A);
    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/mergeSort.sh

```
#!/bin/bash
avl -o mergeSort mergeSort.avl
```

file: avl/tests/avl.test/operator_bool_mid.avl

```
int main(){

    bool a = true;
    bool b = false;
    print a && b;
    print a || b;
    return 0;
}
```

file: avl/tests/avl.test/operator_bool_mid.sh

```
#!/bin/bash
avl -o operator_bool_mid operator_bool_mid.avl
```

file: avl/tests/avl.test/operator_bool_post.avl

```
int main(){

    bool a = true;
    print (!a);

}
```

file: avl/tests/avl.test/operator_bool_post.sh

```
#!/bin/bash
avl -o operator_bool_post operator_bool_post.avl
```

file: avl/tests/avl.test/operator_char_mid.avl

```
int main(){

    char c = 'c';
    int b = 2;
    int a  = 'b';
    print c*b, c/b;
    print c+b,4-b;
    print c>a, c<a, c>=a, c<=a;
    print c!=a, c==a;
    print c == 'c', c != 'c';

    return 0;
}
```

file: avl/tests/avl.test/operator_char_mid.sh

```
#!/bin/bash
avl -o operator_char_mid operator_char_mid.avl
```

file: avl/tests/avl.test/operator_char_post.avl

```
int main(){

    char a = 'c';

    print a++;
    print a;
    print a--;
    print a;

}
```

file: avl/tests/avl.test/operator_char_post.sh

```
#!/bin/bash
avl -o operator_char_post operator_char_post.avl
```

file: avl/tests/avl.test/operator_char_unary.avl

```
int main(){

    char a = 'c';
    print ++a;
    print a;
    print --a;
    print a;

    return 0;
}
```

file: avl/tests/avl.test/operator_char_unary.sh

```
#!/bin/bash
avl -o operator_char_unary operator_char_unary.avl
```

file: avl/tests/avl.test/operator_index_mid.avl

```
int main(){


    index i = 1;
    index j = 2;
    int a = 1;

    print i*j,i/j;
    print i*a,i/a;
    print i+j, i-j;
    print i+a, i-a;
    print i>j, i<j, i>=j, i<=j;
    print j>i, j<i, j>=i, j<=i;

    index n = 1;
    print i>n, i<n, i>=n, i<=n;
    print i>a, i<a, i>=a, i<=a;
    print i!=j,i==j;
    print i!=n,i==n;
    print i!=a,i==a;


    return 0;
}
```

file: avl/tests/avl.test/operator_index_mid.sh

```
#!/bin/bash
avl -o operator_index_mid operator_index_mid.avl
```

file: avl/tests/avl.test/operator_index_post.avl

```
int main(){
```

```
    index i = 1;
    print i++;
    print i;
    print i--;
    print i;
    return 0;
}
```

file: avl/tests/avl.test/operator_index_post.sh

```
#!/bin/bash
avl -o operator_index_post operator_index_post.avl
```

file: avl/tests/avl.test/operator_index_unary.avl

```
int main(){

    index i = 0;
    print --i;
    print i;
    print ++i;
    print i;
    print +i;
    print -i;

    return 0;

}
```

file: avl/tests/avl.test/operator_index_unary.sh

```
#!/bin/bash
avl -o operator_index_unary operator_index_unary.avl
```

file: avl/tests/avl.test/operator_int_mid.avl

```
int main(){

    int a = 3;
    int b = 2;
    int c = 3;
    int d = 4;
    int e = c%d;
    print a*b, a/b, a%b;
    print a+b, a-b;
    print a>b, a<b, a>=b, a<=b;
    print a!=b, a==b;
    print a>c, a<c, a>=c, a<=c;
    print a!=c, a==c;
    print a>d, a<d, a>=d, a<=d;
    print a!=d, a==d;
    return 0;
}
```

file: avl/tests/avl.test/operator_int_mid.sh

```
#!/bin/bash
avl -o operator_int_mid operator_int_mid.avl
```

file: avl/tests/avl.test/operator_int_post.avl

```
int main(){

    int a = 1;
    print +a;
    print -a;
    print ++a;
    print a;
    print --a;
    print a;
    return 0;
}
```

file: avl/tests/avl.test/operator_int_post.sh

```
#!/bin/bash
avl -o operator_int_post operator_int_post.avl
```

file: avl/tests/avl.test/operator_int_unary.avl

```
int main(){

    int a = 1;
    print a++;
    print a;
    print a--;
    print a;

    return 0;

}
```

file: avl/tests/avl.test/operator_int_unary.sh

```
#!/bin/bash
avl -o operator_int_unary operator_int_unary.avl
```

file: avl/tests/avl.test/operator_string_mid.avl

```
int main(){

    string a = "hello ";
    print a;
    string b = "world";
    print b;
    string c = a + b;
    print c;
```

```
    return 0;
}
```

file: avl/tests/avl.test/operator_string_mid.sh

```
#!/bin/bash
avl -o operator_string_mid operator_string_mid.avl
```

file: avl/tests/avl.test/quick_sort.avl

```
void quicksort(int a[]) {
    if (len(a) <= 1)
        return;
    display index i = -1;
    display index j = 0;
    display index k = len(a) - 1;
    display index e = k;
    <begin_display>
    while (j < k) {
        if (a[j] >= a[e]) {
            j = j + 1;
        }
        else {
            swap(a, i + 1, j);
            i = i + 1;
            j = j + 1;
        }

    }
    swap(a, i + 1, e);
    quicksort(a[0 : (i+1)]);
    quicksort(a[(i + 2) : (k + 1)]);
    <end_display>
}

int main() {
    <begin_display>
```

```
    display int a[] = {5, 2, 3, 6, 1, 7, 4, 9, 8};
    quicksort(a);
    <end_display>
    return 0;
}
```

file: avl/tests/avl.test/quick_sort.sh

```
#!/bin/bash
avl -o quick_sort quick_sort.avl
```

file: avl/tests/avl.test/statement_empty.avl

```
int main(){


    int a = 1;
    {

    }

    return 0;
}
```

file: avl/tests/avl.test/statement_empty.sh

```
#!/bin/bash
avl -o statement_empty statement_empty.avl
```

file: avl/tests/avl.test/statement_for.avl

```
int main(){

    /*empty, line 4 failed*/
    for(int i = 0; i<10; i++);
```

```
for(int i = 0; i<10; i++){

}

/*regular for*/
for(int i = 0; i<10; i++){
    print i;
}

/*changing i inside*/
for(int i = 0; i<10;){
    print i;
    i = i+2;
}

/*decleare i outside for*/
int i = 0;
for(i;i<10;i++)
    print i;
for(i;i>0;)
    print i--;

/*testing continue,break*/
for(i;i<10;i++){
    if(i == 2)
        continue;
    else if( i == 5)
        break;
    print i;
}


/*testing nested for*/
for(i=0;i<3;i++){
    for(int j = 0; j<2; j++)
        print j;
}

/*testing return*/
for(i=0;i<10;i++){
```

```
        print i;
        if(i==5)
            return 0;
    }


    return 0;
}
```

file: avl/tests/avl.test/statement_for.sh

```
#!/bin/bash
avl -o statement_for statement_for.avl
```

file: avl/tests/avl.test/statement_if.avl

```
int main(){

    int a = 1;
    int b = 3;
    if(a < b){
        if(a<b)
            print a;
        if(a>b)
            print b;
    }else{
        print "I should be displayed";

    }

    if(a > b){
        print "I should be displayed";
    }else{
        print "correct!";
    }

    return 0;
}
```

file: avl/tests/avl.test/statement_if.sh

```
#!/bin/bash
avl -o statement_if statement_if.avl
```

file: avl/tests/avl.test/statement_scope.avl

```
int main(){

    int a = 1;
    {
        int j = 1;
    }
    print j;
    return 0;

}
```

file: avl/tests/avl.test/statement_scope.sh

```
#!/bin/bash
avl -t statement_scope.avl
```

file: avl/tests/avl.test/statement_while.avl

```
int main(){

    int i = 0;

    /*empty while, failed*/
    while(i++ < 10);
    while(i-- > 0){}
```

```
/*empty do while, failed*/
do;
while(i++<10);

/*normal while*/
while(i>0){
    print i;
    i--;
}

while(i<10)
    print i++;

/*normal do while*/
do print i--;
while ( i > 0);

do{
    print i;
    i++;
}while ( i < 10 );

/*break , continue and return*/

while(i > 0){
    i--;
    if(i == 2)
        continue;
    else if(i==5)
        break;
    print i;


}

i = 0;
while(i++<3){
    int j = 0;
    while(j<2)
        print j++;
}
```

```
    i=0;


    while(i < 10){
        print i++;
        if(i == 5)
            return 0;
    }



    return 0;
}



file: avl/tests/avl.test/statement_while.sh

#!/bin/bash
avl -o statement_while statement_while.avl


file: avl/tests/avl.test/subarray_int.avl

void subarray(int x[]){
    index l = len(x);
    if(l<=1)
        return;

    <begin_display>
        subarray(x[0:l]);
    <end_display>



}

int main(){

    display int array[10];
    int j = 0;
```

```
    for(index i = 0; i < 10; i++) {
        array[i] = j;
        j++;
    }
    <begin_display>
        subarray(array);
    <end_display>

    return 0;

}
```

file: avl/tests/avl.test/subarray_int.sh

```
#!/bin/bash
avl -o subarray_int subarray_int.avl
```

file: avl/tests/error.test/array_malvalue.avl

```
int main() {
    bool a = true;
    int test[10];
    test[1] = a;

    return 0l
}
```

file: avl/tests/error.test/array_malvalue.sh

```
#!/bin/bash
avl -t array_malvalue.avl
```

file: avl/tests/error.test/array_nosize.avl

```
int main() {
    bool a[];
    return 0;
}
```

file: avl/tests/error.test/array_nosize.sh

```
#!/bin/bash
avl -t array_nosize.avl
```

file: avl/tests/error.test/break.avl

```
int main() {
    int i = 0;
    while(i < 10) {
        print i;
    }
    break;

    return 0;
}
```

file: avl/tests/error.test/break.sh

```
#!/bin/bash
avl -t break.avl
```

file: avl/tests/error.test/display_unmatch.avl

```
int main() {
```

```
    <begin_display>
    int a = 6;
    print a;
    <end_display>
    <end_display>
    return 0;
}
```

file: avl/tests/error.test/display_unmatch.sh

```
#!/bin/bash
avl -t display_unmatch.avl
```

file: avl/tests/error.test/dup_break.avl

```
int main() {
    for (int i = 0; i < 10; i++) {
        break;
        break;
    }
    return 0;
}
```

file: avl/tests/error.test/dup_break.sh

```
#!/bin/bash
avl -t dup_break.avl
```

file: avl/tests/error.test/func_malformat.avl

```
int test() {
    return 1;
}
```

```
int main(){
    test + 1;
    return 0;
}
```

file: avl/tests/error.test/func_malformat.sh

```
#!/bin/bash
avl -t func_malformat.avl
```

file: avl/tests/error.test/func_malparam.avl

```
int test(int b) {
    return b + 1;
}

int main() {
    test(true);
    return 0;
}
```

file: avl/tests/error.test/func_malparam.sh

```
#!/bin/bash
avl -t func_malparam.avl
```

file: avl/tests/error.test/func_malscope.avl

```
void test_b() {

    int test_a(int a) {
        return a+1;
```

```
    }

    int a = 1;
    int b = test_a(a);

    print b;
}

int main() {

    test_a();

    return 0;
}
```

file: avl/tests/error.test/func_malscope.sh

```
#!/bin/bash
avl -t func_malscope.avl
```

file: avl/tests/error.test/func_malvalue.avl

```
int test(bool b) {
    return b;
}

int main() {
    test(true);
    return 0;
}
```

file: avl/tests/error.test/func_malvalue.sh

```
#!/bin/bash
avl -t func_malvalue.avl
```

file: avl/tests/error.test/func_missvalue.avl

```
int test(int a) {
    int b = a + 1;
}

int main() {
    test(1);
    return 0;
}
```

file: avl/tests/error.test/func_missvalue.sh

```
#!/bin/bash
avl -t func_missvalue.avl
```

file: avl/tests/error.test/func_numparam.avl

```
int test(int a, int b) {
    return a + b;
}

int main() {
    int a = 1;
    int b = 2;
    int c = test(a,b,3,4,5);
    return 0;
}
```

file: avl/tests/error.test/func_numparam.sh

```
#!/bin/bash
```

```
avl -t func_numparam.avl
```

```
file: avl/tests/error.test/func_undefine.avl

int main(){
    int a = test();
    print a;
    return 0;
}

int test() {
    return 1;
}
```

```
file: avl/tests/error.test/func_undefine.sh

#!/bin/bash
avl -t func_undefine.avl
```

```
file: avl/tests/error.test/init_test.sh

#!/bin/bash
```

```
file: avl/tests/error.test/Makefile.am

AM_TESTS_ENVIRONMENT = . $(srcdir)/init_test.sh;
AM_TESTS_FD_REDIRECT = 9>&2

TESTS = \
            func_malformat.sh \
            func_malparam.sh \
            func_malscope.sh \
            func_malvalue.sh \
```

```
            func_missvalue.sh \
            func_undefine.sh \
            func_numparam.sh \
            break.sh \
            dup_break.sh \
            display_unmatch.sh \
            array_nosize.sh \
            array_malvalue.sh

XFAIL_TESTS = \
            func_malformat.sh \
            func_malparam.sh \
            func_malscope.sh \
            func_malvalue.sh \
            func_missvalue.sh \
            func_undefine.sh \
            func_numparam.sh \
            break.sh \
            dup_break.sh \
            display_unmatch.sh \
            array_nosize.sh \
            array_malvalue.sh
```

```
file: avl/tests/libavl.test/.gitignore

avlint_assign1
avlint_add1
avlint_add2
avlint_add3
avlint_sub1
```

```
file: avl/tests/libavl.test/avlint_add1.cpp

#include <iostream>
#include <cstdlib>
```

```
#include "AvlTypes.h"
using namespace std;

int main(int argc, char *argv[])
{
    AvlInt a = atoi(argv[1]);
    int b = atoi(argv[2]);

    cout << a + b << endl;

    return 0;
}
```

file: avl/tests/libavl.test/avlint_add1.sh

```
#!/bin/bash

SUCCESS=0

program="./avlint_add1"
declare -i n=20
declare -a input=("0 0" "1 2" "-1 -2" "-1 2" "-2 1")

source genRandom.sh
genRandom $n 2
input=("${input[@]}" "${rArray[@]}")
len=${#input[@]}

for i in $(seq 1 $(( len - 1 )))
do
    num1=`echo ${input[$i]} | awk '{print $1}'`
    num2=`echo ${input[$i]} | awk '{print $2}'`
    expected=$(( num1 + num2 ))
    output=`$program ${input[$i]}`

    if [[ "$output" != "$expected" ]]
    then
        SUCCESS=1
    fi
```

```
done

exit $SUCCESS
```

file: avl/tests/libavl.test/avlint_add2.cpp

```
#include <iostream>
#include <cstdlib>
#include "AvlTypes.h"
using namespace std;

int main(int argc, char *argv[])
{
    AvlInt a = atoi(argv[1]);
    AvlInt b = atoi(argv[2]);

    cout << a + b << endl;

    return 0;
}
```

file: avl/tests/libavl.test/avlint_add2.sh

```
#!/bin/bash

SUCCESS=0

program="./avlint_add2"
declare -i n=20
declare -a input=("0 0" "1 2" "-1 -2" "-1 2" "-2 1")

source genRandom.sh
genRandom $n 2
input=("${input[@]}" "${rArray[@]}")
len=${#input[@]}

for i in $(seq 1 $(( len - 1 )))
```

```
do
    num1=‘echo ${input[$i]} | awk ’{print $1}’‘
    num2=‘echo ${input[$i]} | awk ’{print $2}’‘
    expected=$(( num1 + num2 ))
    output=‘$program ${input[$i]}‘

    if [[ "$output" != "$expected" ]]
    then
        SUCCESS=1
    fi
done

exit $SUCCESS
```

file: avl/tests/libavl.test/avlint_add3.cpp

```
#include <iostream>
#include <cstdlib>
#include "AvlTypes.h"
using namespace std;

int main(int argc, char *argv[])
{
    int a = atoi(argv[1]);
    AvlInt b = atoi(argv[2]);

    cout << a + b << endl;

    return 0;
}
```

file: avl/tests/libavl.test/avlint_add3.sh

```
#!/bin/bash

SUCCESS=0
```

```
program="./avlint_add3"
declare -i n=20
declare -a input=("0 0" "1 2" "-1 -2" "-1 2" "-2 1")

source genRandom.sh
genRandom $n 2
input=("${input[@]}" "${rArray[@]}")
len=${#input[@]}

for i in $(seq 1 $(( len - 1 )))
do
    num1=‘echo ${input[$i]} | awk ’{print $1}’‘
    num2=‘echo ${input[$i]} | awk ’{print $2}’‘
    expected=$(( num1 + num2 ))
    output=‘$program ${input[$i]}‘

    if [[ "$output" != "$expected" ]]
    then
        SUCCESS=1
    fi
done

exit $SUCCESS
```

file: avl/tests/libavl.test/avlint_assign1.cpp

```
#include <iostream>
#include <cstdlib>
#include "AvlTypes.h"
using namespace std;

int main(int argc, char *argv[])
{
    int num = atoi(argv[1]);

    AvlInt a = num;

    cout << a << endl;
```

```
    return 0;
}
```

file: avl/tests/libavl.test/avlint_assign1.sh

```
#!/bin/bash

SUCCESS=0

program="./avlint_assign1"
declare -i n=20
declare -a input=("0")

source genRandom.sh
genRandom $n 1
input=("${input[@]}" "${rArray[@]}")
len=${#input[@]}

for i in $(seq 1 $(( len - 1 )))
do
    num=‘echo ${input[$i]} | awk ’{print $1}’‘
    expected=$(( num ))
    output=‘$program ${input[$i]}‘

    if [[ "$output" != "$expected" ]]
    then
        SUCCESS=1
    fi
done

exit $SUCCESS
```

file: avl/tests/libavl.test/avlint_sub1.cpp

```
#include <iostream>
#include <cstdlib>
#include "AvlTypes.h"
```

```
using namespace std;

int main(int argc, char *argv[])
{
    AvlInt a = atoi(argv[1]);
    int b = atoi(argv[2]);

    cout << a - b << endl;

    return 0;
}
```

file: avl/tests/libavl.test/avlint_sub1.sh

```
#!/bin/bash

SUCCESS=0

program="./avlint_sub1"
declare -i n=20
declare -a input=( "0 0" "3 1" "1 3" "-1 -2" "-2 -1" "-1 2" "-2 1" "1 -2")

source genRandom.sh
genRandom $n 2
input=("${input[@]}" "${rArray[@]}")
len=${#input[@]}

for i in $(seq 1 $(( len - 1 )))
do
    num1=`echo ${input[$i]} | awk '{print $1}'`
    num2=`echo ${input[$i]} | awk '{print $2}'`
    expected=$(( num1 - num2 ))
    output=`$program ${input[$i]}`

    if [[ "$output" != "$expected" ]]
    then
        SUCCESS=1
    fi
done
```

```
exit $SUCCESS
```

file: avl/tests/libavl.test/genRandom.sh

```
#!/bin/bash

declare -a rArray=()

# Argument 1: #elements in the array
# Argument 2: #numbers in each element
genRandom ()
{
    rArray=()
    declare -i n=$(( $1 - 1 ))

    for i in $(seq 0 $n)
    do
        element=$(( ( RANDOM % 20000 ) - 10000 ))

        if [[ $2 -ge 2 ]]
        then
            for j in $(seq 2 $2)
            do
                num=$(( ( RANDOM % 20000 ) - 10000 ))
                element="$element $num"
            done
        fi

        rArray[$i]="$element"
    done
}
```

file: avl/tests/libavl.test/init_test.sh

```
#!/bin/bash
```

file: avl/tests/libavl.test/Makefile.am

```
check_PROGRAMS = \
                 avlint_assign1 \
                 avlint_add1 \
                 avlint_add2 \
                 avlint_add3 \
                 avlint_sub1

avlint_assign1_SOURCES = avlint_assign1.cpp
avlint_add1_SOURCES = avlint_add1.cpp
avlint_add2_SOURCES = avlint_add2.cpp
avlint_add3_SOURCES = avlint_add3.cpp
avlint_sub1_SOURCES = avlint_sub1.cpp

AM_CPPFLAGS = -Wall -Werror -std=c++11 -I$(top_builddir)/lib -I/opt/local/include
AM_LDFLAGS = -L$(top_builddir)/lib -L/opt/local/lib -lavl -lglut -lGL

AM_TESTS_ENVIRONMENT = . $(srcdir)/init_test.sh;
AM_TESTS_FD_REDIRECT = 9>&2

TESTS = \
        avlint_assign1.sh \
        avlint_add1.sh \
        avlint_add2.sh \
        avlint_add3.sh \
        avlint_sub1.sh
```

file: avl/tests/Makefile.am

```
SUBDIRS = libavl.test avl.test error.test
```

10.2 Git Log

```
commit 2caddd3ca010f8bd79753f51a0dcae794d720c4b
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun May 11 14:18:33 2014 -0400

    code.sh updated

commit 01d34e46f6004be8026cde861c26a21b8b1faee1
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun May 11 14:14:59 2014 -0400

    script updated

commit befac159e4c864953ae79bebca4c36657fa3e89e
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun May 11 14:09:05 2014 -0400

    script updated

commit b8ebe38cd51157b5033ccbb20bb46d081884fc0c
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun May 11 14:05:07 2014 -0400

    report done

commit 57e527d63d8aaa1992062592c5190633a86753b7
Merge: 747ea9c 9428692
Author: seemuch <shining.app@gmail.com>
Date:   Sun May 11 04:14:26 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 747ea9c4343c4a6ac71ea6a2ffe2f0fe41bc9a9f
Author: seemuch <shining.app@gmail.com>
Date:   Sun May 11 04:13:58 2014 -0400

    Add type checking

commit 94286929e9379ad71400246d0291b43adacb8251
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun May 11 02:52:13 2014 -0400
```

```
    remove debug code

commit ee6790a3a4c1d3ba8a3f07705b233b77e113c91b
Merge: 2ed37fb eb37825
Author: seemuch <shining.app@gmail.com>
Date:   Sun May 11 02:18:53 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 2ed37fb2f2a413a70aa21b0d1231e49e51d953ff
Author: seemuch <shining.app@gmail.com>
Date:   Sun May 11 02:18:37 2014 -0400

    Add type check

commit eb3782589b59b91454c2967afa314d3f8691df01
Merge: 3ec8584 2726622
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sat May 10 23:36:56 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 3ec85845b7f97579c7022abc0c0950739b389cc6
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sat May 10 23:36:43 2014 -0400

    update

commit 2726622fa9665534dcc751206dbeb07bf9028f68
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sat May 10 23:28:36 2014 -0400

    add pop, push, dequeue in AvlArray class

commit aaba1bc1a686ea55b4158e8873a255479859c038
Merge: 09859b9 8315fb4
Author: stella <zhangqx10@gmail.com>
Date:   Sat May 10 22:09:59 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl
```

```
commit 09859b926075bdaab7ba1d8b1095696d5b897029
Author: stella <zhangqx10@gmail.com>
Date:   Sat May 10 22:09:22 2014 -0400

    add sleep for each statement when display

commit 8315fb478d7d8b7e9ae86a1d696dbe9d11c1762d
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sat May 10 21:32:20 2014 -0400

    update mergesort

commit 866ff1671f371c85330fc8d93afc8fdc4e88d8dc
Merge: c3d1a77 1652f08
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sat May 10 17:40:35 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit c3d1a77b782adfcdf969e7d46cd96c6d20c3aab0
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sat May 10 17:40:25 2014 -0400

    temp

commit 1652f08fc15ae00774f41886477bf781abb9b194
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sat May 10 17:25:09 2014 -0400

    modified merge_sort_array.avl

commit bb63bad726e49a2c5ed03f6ed6c9b1c28154a587
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sat May 10 16:38:34 2014 -0400

    nothing

commit 4ed6a7e64ba42a5f01360087c1fab921630d4771
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sat May 10 15:27:27 2014 -0400
```

```
    fixed Issue #26

commit 44f5488ed845db213207f6e8711155d4514ce062
Merge: 2fa368b c41f57a
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sat May 10 14:18:13 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 2fa368b92f7b35bb96400dd08a50f84827de04e3
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sat May 10 14:18:00 2014 -0400

    mergesort and maxsubarray

commit c41f57a4927000bda422dc5aadd49fc67fa7a37d
Merge: 184857c d8faaa1
Author: seemuch <shining.app@gmail.com>
Date:   Sat May 10 14:02:16 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 184857cadeb5cb54be4eca0f7c741fc3787c809f
Author: seemuch <shining.app@gmail.com>
Date:   Sat May 10 14:01:36 2014 -0400

    Modified struct identifier.

commit d8faaa174e8da7bf91c8bd1ad652d6acabd33da1
Author: stella <zhangqx10@gmail.com>
Date:   Sat May 10 12:53:56 2014 -0400

    bug fixed test case

commit 2336afebb8bdadf7f1aac4f4733fcc963f722163
Author: stella <zhangqx10@gmail.com>
Date:   Sat May 10 12:48:47 2014 -0400

    bug fixed empty statement
```

```
commit 3d786e73df88bf22a640c2437c914dfc012c9e34
Merge: 89e49fd 2234350
Author: stella <zhangqx10@gmail.com>
Date:   Sat May 10 12:20:13 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 89e49fdd2e328399b5e15e5090e22e76218eb3c8
Author: stella <zhangqx10@gmail.com>
Date:   Sat May 10 12:19:55 2014 -0400

    add empty statement

commit 22343501b161e43a9599039bb4cb5affc2abe052
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sat May 10 11:37:35 2014 -0400

    modified most of testcases

commit a2b4af05e948ad8797ed692bea28c7965bfa9405
Author: stella <zhangqx10@gmail.com>
Date:   Sat May 10 01:07:30 2014 -0400

    fix test case

commit aed7a062253d4c778903ba4dff298ba3a3f4c4ad
Merge: 0f8e249 c166a62
Author: stella <zhangqx10@gmail.com>
Date:   Sat May 10 00:48:45 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 0f8e2496e8d00f5ccfbfa785aec54ca23dd26632
Author: stella <zhangqx10@gmail.com>
Date:   Sat May 10 00:48:12 2014 -0400

    fix para declaration

commit c166a629b0255bce47752305ab9ea383b046a591
Merge: 3bfca76 9bd3b2f
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
```

```
Date:   Sat May 10 00:41:04 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 3bfca76455a73e7b5c31d702b2d0647a58e60b66
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sat May 10 00:40:31 2014 -0400

    modified the error in test cases

commit 9bd3b2f04b8e387bf32bb3c8d7f447a87bf8be37
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sat May 10 00:38:23 2014 -0400

    add a new sample code

commit ef577d338c6816d3481117ad9e710de15983ddd3
Merge: 2b2db8d bbb3ed1
Author: wqfish <wuqinfan@gmail.com>
Date:   Sat May 10 00:28:38 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 2b2db8d4f6d9d630a78dec6266caa351113110b1
Author: wqfish <wuqinfan@gmail.com>
Date:   Sat May 10 00:28:10 2014 -0400

    libavl tests init_test.sh updated

commit bbb3ed165803dc1b7db1cdf7a02195dd40f47fc8
Merge: 34423ae 1b87c74
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Fri May 9 23:54:25 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 34423ae23cee402f09db1aa43def82f4ccb9db40
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Fri May 9 23:54:12 2014 -0400

    more failed test cases
```

```
commit 1b87c742237f75f77ee79bd467e7903b0802cde7
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 23:49:06 2014 -0400

    fix char expression

commit a67361a2dd57eced0ae71c55e42ccd05740b21aa
Merge: d2860a6 1c32a4d
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 23:28:30 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit d2860a6cb58f7c3b21769dd7cd32a9ef6ffdc853
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 23:26:01 2014 -0400

    add string

commit 1c32a4de57094bedb371a6eec301d4ada7825b09
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri May 9 23:25:24 2014 -0400

    type fixed

commit 1f5d2076659d1a10dfc5e4ea52f8124a0bb385d1
Merge: d26698f 920c6d7
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 22:54:31 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit d26698f718bafa44d3c03f3620c1e10f773c72ba
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 22:53:21 2014 -0400

    fix bug assignment in declaration

commit 920c6d738c2c7e177b4eac72bbd94946959ee87c
Author: Qinfan Wu <wuqinfan@gmail.com>
```

```
Date:   Fri May 9 22:53:09 2014 -0400

    fixed gcc link problem on Ubuntu

commit 8d64711d079fb7b3bdcfe18a2c20c94f95d3f325
Merge: 58937c2 fb27457
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri May 9 22:43:16 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 58937c21c857ee8fcecf4e68bf291aeb9927216a
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri May 9 22:43:01 2014 -0400

    add support for multiple objects

commit fb27457edcd95c00e480c8dc9f5fb74dda2eaad0
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Fri May 9 22:38:14 2014 -0400

    modified some test cases end remove -werror option

commit 73a822b5e766fb64688bf8e8ce56b8a14f628a24
Merge: 3d908e4 dc0d3f1
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Fri May 9 21:59:31 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 3d908e44403dd358c45bdcffaa758516ca0dedc5
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Fri May 9 21:59:11 2014 -0400

    correct makefile

commit dc0d3f13c289992171ef102e9373d63c430bf0e0
Merge: 65cc1b3 0ef5feb
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 21:54:02 2014 -0400
```

```
    Merge branch 'master' of https://github.com/wqfish/avl

commit 65cc1b3d3b6fdec77fa682a40af20e12c00aea90
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 21:53:44 2014 -0400

    add true and false

commit 0ef5feb431b1eb82e9de3e42621e169ca5ba4e10
Merge: 8f817fe 9d1b832
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Fri May 9 21:44:27 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 8f817fe0487b84752e4802531e3f112d6449cf8c
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Fri May 9 21:43:46 2014 -0400

    add tokens in scanner.l

commit 9d1b832428b16399e9c451519d1d740026728a47
Author: wqfish <wuqinfan@gmail.com>
Date:   Fri May 9 21:35:36 2014 -0400

    bug fixed

commit 1a44d09ffe13ed4b6731467924a29f3ba1937b14
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 21:21:54 2014 -0400

    fix bugs

commit c37c1162f87ca7ebeccf43d9928cc086ce108192
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 21:13:43 2014 -0400

    fix return type of main

commit fcedb42453d7c67b96a2a06174e804622829bc4c
Merge: c8d088d 0e481aa
```

```
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 20:37:26 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 0e481aa9b0f9c04638db589797a2b71fbd626b8c
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri May 9 20:36:50 2014 -0400

    bug fixed

commit c8d088db7dbeb6b40ad7823b5abaac601a24e622
Merge: 94e1b72 6b1139e
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 20:32:27 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 94e1b7273f553270efde446b13e7ba538cccfb2d
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 20:21:40 2014 -0400

    delete cast

commit 6b1139e4b661fed5475e781ed8a60809b5a49610
Author: seemuch <shining.app@gmail.com>
Date:   Fri May 9 20:11:59 2014 -0400

    Bug fixed

commit 876e2a894098f0f4be8be03e3e2cd8c142690833
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Fri May 9 20:10:08 2014 -0400

    modified the Makefile.am in tests/

commit b9012e7f6e15462e1508a8d950cb01643696c280
Merge: 8dc8aa5 96d6a6c
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 20:02:31 2014 -0400
```

```
    Merge branch 'master' of https://github.com/wqfish/avl

commit 8dc8aa58c8b68bb2321aaa256d89f7b73dfc0929
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 20:01:29 2014 -0400

    scanner string add sepcial char

commit 96d6a6ce1cecd920c1490ceebaed365cc60ab2c5
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri May 9 19:56:44 2014 -0400

    add sample code: merge_sorted_array

commit 65d939181cc2aee9e8d22488710b553ff36713b4
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri May 9 19:51:42 2014 -0400

    updated

commit 914412809fa9864d2104431ef4808c2bc208a36e
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 19:48:14 2014 -0400

    change file name

commit 6071e670ee98508661c528a92c1dc7c2604e332d
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 19:47:10 2014 -0400

    change function name

commit 09b7dcedda41adedbc88920e7c80bd66accf7c5a
Merge: e7bec3d 6d41746
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 18:49:35 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit e7bec3df10603eef7c30692b1674a62dfd4d8d04
Author: stella <zhangqx10@gmail.com>
```

```
Date:   Fri May 9 18:48:22 2014 -0400

    fix bugs in for loop, declaration and return

commit 6d41746fd266783e5a177136ec58593b768a6b2b
Author: seemuch <shining.app@gmail.com>
Date:   Fri May 9 18:21:30 2014 -0400

    But fixex.

commit 84997e08c61c3ce46184eede59fbf42de07f4343
Author: seemuch <shining.app@gmail.com>
Date:   Fri May 9 17:20:13 2014 -0400

    Bug fixes

commit b8a4d644ef1724998933d0ba36cbb100ba85d27f
Author: seemuch <shining.app@gmail.com>
Date:   Fri May 9 17:16:27 2014 -0400

    Bug fixes based on Wu's comments.

commit 405abbdb0f33cf6ad7fb03ae838fce32d3f1d9f4
Author: seemuch <shining.app@gmail.com>
Date:   Fri May 9 16:58:16 2014 -0400

    Bug fix for glutBitmapString

commit 0accec5fb7fd6cb98db6956f7ef9914c705d12e4
Merge: 44501f8 6038883
Author: seemuch <shining.app@gmail.com>
Date:   Fri May 9 16:49:58 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

    Conflicts:
     sample_code/sorting.cpp

commit 44501f896ad51629da29df6fd67cf2b5bba0ad2c
Author: seemuch <shining.app@gmail.com>
Date:   Fri May 9 16:48:11 2014 -0400
```

```
    aaa

commit 60388833670e85fc5bcf6f775de5470f180534d3
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 15:17:03 2014 -0400

    test

commit 31ff85abeff0a4609a0d4e501041fb63d7e43992
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 14:41:47 2014 -0400

    temp

commit aa86cd77574527426b4f6aef7d9a0a294c842448
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 13:11:15 2014 -0400

    change head file

commit 0170a582b3ea242395f546d2e9aacc28aa424ae9
Merge: 3211633 a83bca1
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 13:05:22 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 3211633e4231ef6b831ad781c31747a49235c859
Author: stella <zhangqx10@gmail.com>
Date:   Fri May 9 13:02:33 2014 -0400

    fix bugs. begin to use symble table

commit a83bca19ff758349dd1b8b5b3e65396d3aebc9fe
Author: wqfish <wuqinfan@gmail.com>
Date:   Fri May 9 12:54:09 2014 -0400

    Update README.md

commit 8c2d7b38969396cf5a8472e52c6bdf628f942a52
```

```
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun May 4 14:10:24 2014 -0400

    bug fixed

commit bbb500bb3691e87d1afb1da124eca56a208df9fb
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun May 4 14:06:58 2014 -0400

    bug fixed

commit 9a3c5b6f21e98ea48a175b712848a0a9949a7cd6
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sat May 3 22:09:12 2014 -0400

    modify the last line problem

commit 31f4598b61004b7a00c5a1d530b79d3059794f40
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sat May 3 21:49:31 2014 -0400

    split many test cases, and add some new cases

commit b766cc637978ee5413b9c7af92fb50d10108aed0
Author: stella <zhangqx10@gmail.com>
Date:   Sat Apr 26 16:53:45 2014 -0400

    add default case

commit 9d484d7175c26c9afcbbb63138553b38f7d3c48b
Author: stella <zhangqx10@gmail.com>
Date:   Sat Apr 26 16:49:33 2014 -0400

    add default to switch case

commit 1966f67a19d120de3b0333c1ce38c30f761ba934
Merge: 32ffd50 004beba
Author: stella <zhangqx10@gmail.com>
Date:   Sat Apr 26 16:23:59 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl
```

```
    Conflicts:
     sample_code/sorting.cpp
     src/Makefile.am

commit 32ffd50dd9bbf098106da1d83dabbedd37b272c3
Merge: 747e67e 5d831a6
Author: stella <zhangqx10@gmail.com>
Date:   Sat Apr 26 16:20:47 2014 -0400

    Merge branch 'shining'

    Conflicts:
     sample_code/sorting.cpp

commit 004beba7e9e5dc7e14fcda6fc0e904d60c79ccb6
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Apr 22 22:08:01 2014 -0400

    implementation of AvlSleep updated

commit 5d831a6b3d18836b7ce11d60e37602080d110728
Merge: bd39b0b df518c9
Author: stella <zhangqx10@gmail.com>
Date:   Sun Apr 20 18:04:18 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

    Conflicts:
     src/parser.y

commit bd39b0b6c1ca5268f94a326792a9d8e7a92d65b2
Author: stella <zhangqx10@gmail.com>
Date:   Sun Apr 20 17:59:30 2014 -0400

    modify parser.y

commit 3499ef1d701242b4b6e7790d0dc7ba5627eb50c8
Author: stella <zhangqx10@gmail.com>
Date:   Sun Apr 20 17:50:41 2014 -0400
```

```
    simple compiler. err at begin and end of functions

commit a8de1f18e86f8ff3cdedb6ed9105288d94a57d3e
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sun Apr 20 01:03:34 2014 -0400

    added AvlString

commit cfd1fc2bae78f30890e96b5e0f63721f2a1ce527
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Thu Apr 17 22:25:04 2014 -0400

    AvlBool: chage display value

commit 07aa758b2d7915ea9659a5bb88dba0f8e121102c
Merge: f286664 36d6006
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Thu Apr 17 22:02:47 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit f2866647bad0251a287b7ff41ad25614ba1f8f15
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Thu Apr 17 22:02:24 2014 -0400

    Add AvlBool

commit 36d6006fc4b41c3fd9deee911662244229c88906
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Apr 17 11:39:54 2014 -0400

    adjust CPPFLAGS

commit 52586b1c9b9864470dbfb031c5466b913c4b0948
Author: Shining Sun <shining.app@gmail.com>
Date:   Wed Apr 16 12:47:45 2014 -0400

    Bug fixes

commit 44c97112e49f86d05b8bade70ef074cab952cea4
Merge: df518c9 6f02d21
```

```
Author: Shining Sun <shining.app@gmail.com>
Date:   Wed Apr 16 12:04:00 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

    Conflicts:
     sample_code/sorting.cpp

commit df518c9d767e32b676f42982d3dcd90d8c3dafbe
Author: Shining Sun <shining.app@gmail.com>
Date:   Wed Apr 16 11:56:32 2014 -0400

    Bug fixes.

commit a546eea94ba56803ab6c2c3d8b0fc0f7d04ab21c
Merge: 702b246 91fa3b1
Author: Shining Sun <shining.app@gmail.com>
Date:   Wed Apr 16 11:51:33 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

    Conflicts:
     src/parser.y

commit 702b246ecbd581b43272b4494258cf3bd688e58e
Author: Shining Sun <shining.app@gmail.com>
Date:   Wed Apr 16 11:47:10 2014 -0400

    Added AvlIndex and AvlChar

commit 6f02d21aa2bdd92a335709d41580e27415649285
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Apr 15 17:34:13 2014 -0400

    scanner updated

commit 52d6a54cccf71c04eab312a7dc4c1c9dc4843740
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Apr 15 16:26:06 2014 -0400

    lex rules updated
```

```
commit 4881a6a56ecb48b2bd81963d3f351168f7395400
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Apr 15 15:16:18 2014 -0400

    bug fixed

commit f1d81b673e3236c9dd27b50a01c12c9c0f7cd875
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Mon Apr 14 23:52:49 2014 -0400

    modify the makefile to run testcases

commit 94d6f9aa09edca874d1f80e0c6464b4342f4cb3b
Merge: 35e3427 4e8ace7
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Mon Apr 14 23:47:37 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

    Conflicts:
     src/scanner.l

commit 35e342731404d9e0b4e1956c34fe618b7a9347de
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Mon Apr 14 23:39:08 2014 -0400

    added some testcases , hehe

commit 4e8ace71cd22a89d212a0a0d86bbb85d16c7288c
Merge: 708973d 6c5c445
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Mon Apr 14 21:41:10 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 708973d875a7991817e1e2976f6f051703928591
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Mon Apr 14 21:40:44 2014 -0400

    sample code typo fixed
```

```
commit 6c5c4458f2949896c04d313e2e4b4747d56b6711
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Apr 14 21:36:51 2014 -0400

    variable name typo fixed

commit 99b1a46d41aba0519ae760a42aade2517256885e
Merge: ac8ab33 91fa3b1
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Mon Apr 14 20:00:36 2014 -0400

    Merge branch 'shining'

commit 91fa3b17c707cba0260d677169581749a1726945
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Apr 10 17:29:00 2014 -0400

    updated

commit ac8ab33a4395305a57253c8b3eacfa32be049b2b
Author: wqfish <wuqinfan@gmail.com>
Date:   Thu Apr 10 17:13:07 2014 -0400

    Add astyle for code formatting

commit 9fd0f1ec4c0f9056c42f39870fad57ccf2927cd8
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Apr 10 17:10:52 2014 -0400

    add code formatting for target code

commit b97a5768798e58d1e150876c4e99abf28bc867b3
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Apr 10 14:36:18 2014 -0400

    fix: not checking return value of mkstemp()

commit 063ca3be81b6e356250f16ba842a606c07f38afe
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Apr 9 23:06:51 2014 -0400
```

```
    typo fixed

commit 3dd8ec1b7f54f3ed99c829459a024969d72a169f
Author: Shining Sun <shining.app@gmail.com>
Date:   Tue Apr 8 10:25:58 2014 -0400

    Complete parser.y

commit 00a91ba3d23b187339bb1dfd6faaabb05349b107
Author: stella <zhangqx10@gmail.com>
Date:   Tue Apr 8 10:09:11 2014 -0400

    delete main function

commit aeb36ad9f1b93a3c9f7f74754fd4d0441b9b46ce
Author: Shining Sun <shining.app@gmail.com>
Date:   Mon Apr 7 19:27:50 2014 -0400

    Complete parser.y

commit fd8450c40457f5bb12c11c5c68a84dd7e6ac0a19
Merge: 74fd021 873c73b
Author: Shining Sun <shining.app@gmail.com>
Date:   Mon Apr 7 19:18:42 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

commit 74fd0213ab39cf593adf0e8780f54a09957f3707
Author: Shining Sun <shining.app@gmail.com>
Date:   Mon Apr 7 19:17:47 2014 -0400

    Modified parse.y

commit 873c73b0a8a75f7e0e4b2f360809e7c4ef4f9cd9
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Apr 7 14:08:53 2014 -0400

    remove comment() completely from scanner.l

commit 30e0be689c9220eb319e3f0954207978cb491ae0
```

```
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Apr 7 14:06:26 2014 -0400

    updated comment implementation in scanner

commit f7fcbb65347d19348d8ffa643dcd138823d423eb
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 22:44:53 2014 -0400

    Bug fixex.

commit 22d3df90975a387479f85f9909c977b9846e4649
Merge: 1ea7ecd 1f033f0
Author: stella <zhangqx10@gmail.com>
Date:   Sun Apr 6 21:13:31 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

commit 1f033f0f1ab180924f5f249970986067d2fcecf0
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 21:12:29 2014 -0400

    Modified makefile and scanner.l

commit 1ea7ecdc0210a9ed7b78fa9ddc2c7a4b59278514
Merge: 674d454 4dab6ad
Author: stella <zhangqx10@gmail.com>
Date:   Sun Apr 6 21:11:51 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

    Conflicts:
     src/parser.y

commit 4dab6ad105b1254aefb1638339391d5d8fd3c0b5
Merge: f9548f8 3fcbd84
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 21:10:28 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining
```

```
    Conflicts:
     src/parser.y

commit 674d454cb31a7b2636294277681689ea808f7514
Merge: c70a4a2 3fcbd84
Author: stella <zhangqx10@gmail.com>
Date:   Sun Apr 6 21:10:00 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

commit c70a4a2f268a98693aa447ac30747b8b16218c3a
Merge: 1dd4e39 0838743
Author: stella <zhangqx10@gmail.com>
Date:   Sun Apr 6 21:09:40 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

    Conflicts:
     src/include.h

commit f9548f85dc42c04dc840108a1a5941f7c266783e
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 21:05:48 2014 -0400

    asdfasfd

commit 3fcbd84bdc8f062f1741f608a0a6023ed35b8278
Merge: 99ec7a8 0838743
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sun Apr 6 21:00:53 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

commit 0838743e449cc8f3c02169c8ececa4e64bee0895
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 20:59:37 2014 -0400

    Bug fixes

commit 99ec7a8b8b63c90b2e4e1c234fa0a2eb8d9e21e8
Author: MackeyZheng <bearzheng2011@gmail.com>
```

```
Date:   Sun Apr 6 20:59:24 2014 -0400

    updated statement

commit 1dd4e39ebe0b87e877e3af7e877e544613df6863
Author: stella <zhangqx10@gmail.com>
Date:   Sun Apr 6 20:54:00 2014 -0400

    create tree

commit 8ba55f568a064ef8f3c77498ac1c9b4c81ed486d
Merge: 6a008d3 6667eb8
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sun Apr 6 20:53:12 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

commit 6a008d3ae46f61c72d96d42a557b8a6779451c4e
Merge: de2f0ea 6b26684
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sun Apr 6 20:52:34 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

    Conflicts:
     src/include.h
     src/parser.y

commit 6667eb82b293d97ff3882a9e51e150b4057255f7
Merge: 18526ae 6b26684
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 20:51:21 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

commit 18526aed04ade6dc081b39836d83f134d200febb
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 20:51:03 2014 -0400

    Modified .y
```

```
commit de2f0eaf9fdb83a75363fab441618400e986ccc7
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sun Apr 6 20:46:49 2014 -0400

    modified include.h and parser.y in the section of expression

commit 6b266847b8efc776251c7b950df343f3504ab1b7
Merge: 8eaddda e5c354a
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sun Apr 6 20:45:10 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

    Conflicts:
     src/include.h

commit 8eaddda465bd0000cff5918a5263d91209a28a9a
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sun Apr 6 20:44:08 2014 -0400

    statement

commit e5c354a605362a613aa76d5b2c788f84a0119faa
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 20:25:35 2014 -0400

    Modified include.h and .y

commit 000ace602fb19afbba4a78f18ddccd1aba45a1eb
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 20:18:09 2014 -0400

    Modified .y

commit 07f72e29d98a6d52ba39cdf077c9eef97e440ff8
Merge: a52065b e2537d1
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sun Apr 6 20:17:02 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining
```

```
    Conflicts:
     src/include.h

commit a52065ba17e0282e7acf33b15c0d32fcd1898d6e
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sun Apr 6 20:14:51 2014 -0400

    haha

commit e2537d1750a245391ceb7307741d3ad554fa7343
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 20:14:20 2014 -0400

    Add creator.

commit ada2cbb3f68fe9c98d7960802e18769dcac8e715
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 20:10:34 2014 -0400

    Operator node function defined.

commit a0a78718d8f9f04c575bf07e405f6fd34f9b4c03
Merge: 84af314 116a191
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sun Apr 6 20:00:31 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

commit 7fb304dd0422d1705b700df379e4d3a09d197ad4
Merge: 5439b7a 116a191
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sun Apr 6 19:59:41 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

    Conflicts:
     src/include.h
     src/parser.y

commit 84af314ae0c3e05c8354f715490cef45941e7b0b
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
```

```
Date:   Sun Apr 6 19:58:10 2014 -0400

    added some operators

commit 5439b7aad53e92575d3b282c8d2943863df8466f
Author: MackeyZheng <bearzheng2011@gmail.com>
Date:   Sun Apr 6 19:55:50 2014 -0400

    yu zheng

commit 116a19150422d66f0d7788aae0a53002e5ab66da
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 19:55:06 2014 -0400

    Operator API updated.

commit 0a6ff9ddc18d9009d31e83bc9bd07d0f16b4f9a6
Merge: e002789 6e50ddd
Author: stella <zhangqx10@gmail.com>
Date:   Sun Apr 6 19:43:58 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

    Conflicts:
     src/include.h

commit e002789497cbd1ad4e1b219b9380aba0ce922baf
Author: stella <zhangqx10@gmail.com>
Date:   Sun Apr 6 19:39:40 2014 -0400

    change include.h

commit 6e50ddd7dccbe658f6470ab18381584fed8a7827
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 19:26:58 2014 -0400

    Modified included.

commit 4e8ddc7d2c5f05d9dc59101efb0e7a71eef90fea
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 19:07:31 2014 -0400
```

```
    Modified include.h

commit fbee3b4da53412cd635a23a5320d78bccf1bb50d
Merge: 8d65cb4 410faa7
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 19:06:09 2014 -0400

    Merge branch 'shining' of https://github.com/wqfish/avl into shining

    Conflicts:
     src/include.h

commit 8d65cb447e427e848460efe7953e21f72cb83f69
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 19:04:50 2014 -0400

    Modified include.h

commit 490970c82eb0f3f7838e85b5ccaf8f7f91080712
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Apr 6 15:26:19 2014 -0700

    implemented a list for symbol tables

commit 4a8b96ef18b7f8e458c2187b3c52703ca178b633
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Apr 6 14:08:28 2014 -0700

    implemented a symbol table

commit 10a2f5a0dcbd42be53af491b28d748305c6036d7
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Apr 6 13:03:22 2014 -0700

    modified interface of sym_list

commit 410faa7358ab6091e1ecb9d9d9a772d5c699acc1
Author: Jiuyang <zhaojiuyangsjtu@gmail.com>
Date:   Sun Apr 6 15:49:20 2014 -0400
```

```
    added oprNodeType

commit c0622101b9325ca7b2964428b0517eb2d11bb427
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Apr 6 12:36:20 2014 -0700

    implemented a linked list for symbols

commit 1eaaeedd6fb092d1c674219b3e4517e4ff953f4f
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 14:49:18 2014 -0400

    Change include.h

commit a097694c73c685e2dbfa6d52444609c06ca243a4
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 14:37:48 2014 -0400

    More .y

commit f052c723e5cc64aeff0fd3e27dbf448740e08111
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 13:37:49 2014 -0400

    Add include.h

commit cd693b97ad887299f21cddeda6582fc03b962015
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Apr 6 12:44:14 2014 -0400

    Getting started on .y and .l

commit 747e67e3a46ff475e2c3addcce863173fe56714d
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Apr 6 09:35:36 2014 -0700

    sample code updated

commit b42e98f344c16873ef5de81d7627a44a64da3ea7
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sat Apr 5 10:37:30 2014 -0700
```

```
    add -Werror option to configure.ac

commit 114a457f3722fb225603fc72de76aef289b9ff54
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Apr 3 00:53:55 2014 -0400

    sample code updated

commit 19897a07b2135428d76912241631f0e75b3baf91
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Apr 2 22:01:08 2014 -0400

    updated scanner

commit 23d68107e3013a246302c28332e26c9bd6924719
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Apr 2 21:59:12 2014 -0400

    reduce # random numbers in test cases

commit d621f64f80218751247544e44083525ce727490f
Merge: 09e2359 51d1fa7
Author: stella <zhangqx10@gmail.com>
Date:   Wed Apr 2 21:31:51 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 51d1fa73e511e0f3db8db0fdbb7cb46281f77921
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Apr 2 21:28:15 2014 -0400

    remove -Werror option from sample_code

commit 399451d21676a2e1f8197194e64690d261fc1929
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Apr 2 21:22:49 2014 -0400

    remove "-Werror" from configure.ac

commit 09e235953054348cfd71476dee73bcb4e85caff1
```

```
Merge: 81cf015 41a54b9
Author: stella <zhangqx10@gmail.com>
Date:   Wed Apr 2 21:21:08 2014 -0400

    Merge commit '41a54b91176c7d8d7992f3d3b74a4bc2f3d3716f'

commit 81cf0155d77c4c5151d6c822588656a26c9086b0
Author: stella <zhangqx10@gmail.com>
Date:   Wed Apr 2 21:20:58 2014 -0400

    backup

commit 46695a6cd8d29a67b754fd5827d724ee38a7d66f
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Apr 2 21:20:31 2014 -0400

    typo fixed

commit 41a54b91176c7d8d7992f3d3b74a4bc2f3d3716f
Author: stella <zhangqx10@gmail.com>
Date:   Wed Apr 2 21:07:17 2014 -0400

    Update parser and scanner.

commit e31b2ae3012d4a463a3cce927d1b406462da3f92
Author: Shining Sun <shining.app@gmail.com>
Date:   Wed Apr 2 19:29:48 2014 -0400

    Erase debug print.

commit 85c5ccb844940351d8ef5a9e8eecc430822e33d0
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Mar 30 21:10:40 2014 -0400

    Display array name.

commit d01ec22c1acc8a2e02193239fe2c2f5bf8875586
Merge: 5ac0ca7 ec57d9b
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Mar 30 20:36:49 2014 -0400
```

```
    Merge branch 'master' of https://github.com/wqfish/avl

    Conflicts:
     lib/AvlTypes.h

commit 5ac0ca7c72418faa704e78732968f14c611f8be7
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Mar 30 20:32:22 2014 -0400

    Show indices in AvlArray

commit ec57d9b51979f79af972037d3c470b4c0501f875
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri Mar 28 22:25:03 2014 -0400

    add support for print

commit cd7a58955de783d3a87093481416ee2ed49f997d
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri Mar 28 15:26:09 2014 -0400

    sample code updated

commit 3da6fbaf6bcfacf29714796f1fa2194113f1739c
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri Mar 28 14:43:15 2014 -0400

    implementation of avlSleep() updated

commit 1e9e2c1825e9375a78e0a1a83b69f9406744b907
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Mar 26 20:18:30 2014 -0400

    update support for <begin_display> and <end_display>

commit acccf9dc335a51fad00291469e38b2db66a5172d
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Mar 26 19:43:19 2014 -0400

    update support for <begin_display> and <end_display>
```

```
commit 5f3d1ee2d8d32f4ae98d8d1762ec69815f7c1661
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Mar 26 17:46:02 2014 -0400

    place the array in the middle of the window

commit 0b8c36b4ee3d2bebba57b35261cde403fc207b92
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Mar 26 17:10:01 2014 -0400

    add highlight support

commit b82710ba7cd72a5a8152b31d1e54c1ec3e013749
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Mar 25 14:53:37 2014 -0400

    for loop updated

commit 1170a66ebab935a559c5e2f87b170bcb4747a97e
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Mar 25 14:39:07 2014 -0400

    remove AvlArray::width()

commit 815bcc0387e80a9789cf7cd4922a02632891c0c8
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Mar 25 14:22:46 2014 -0400

    private derivation -> a vector member for AvlArray

commit 2e8e352a9230412beb4848488cd70bb2f855cccf
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Mar 25 14:01:08 2014 -0400

    change update() policy for AvlInt

commit ef3906141485a1cafe1bee563f36b69e722656ce
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Mar 25 10:57:27 2014 -0400

    fix potential race conditon in moveObject
```

```
commit de67b58ece0cd57a863a95d294ed89b9d233be4c
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Mar 25 09:39:19 2014 -0400

    add .gitignore files

commit 8a5fca6fba1b09d5476842d1639f8817082efca6
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Mar 25 09:25:17 2014 -0400

    modified makefiles

commit c8cae9e2d954df5282fec6d63a463087b0026e7f
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 24 16:39:15 2014 -0400

    missing -lavl option when invoking g++

commit 6ecb17b75756a15d308ebeae58c5c09e2e986c32
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 24 16:32:28 2014 -0400

    updated

commit 76281fcfcb95dc5b2f09a78db1ba1f62954aeedf
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 24 16:20:00 2014 -0400

    modify some variable names to avoid conflicts with users

commit 07802b811f6ff364d0e1d0e81dedc8677b459e1c
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 24 13:46:57 2014 -0400

    remove pthread completely

commit 34c2cd520f9d2e342ca884cceaeaaeb3a4d82135
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 24 13:12:38 2014 -0400
```

```
    add condition variable in sample code

commit a077ef6c28a639d31efbaed896dea519b6e0635f
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 24 11:54:29 2014 -0400

    change 'bool shouldUpdate' to mutex

commit 53178f3f961bbeaad2d01ea7f0a3724e829bb4bc
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 24 11:02:28 2014 -0400

    add some sample target code

commit 670d5c0c8bb65a80400ab433e184c7d1e15c3afa
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 24 10:54:06 2014 -0400

    add support for <begin_display> and <end_display>

commit 57eca827ac3b8e6d010205d8632559cf19f723a9
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 24 00:00:33 2014 -0400

    overload ++ and -- operator for AvlInt

commit accc8dc02efe72c99f9ddfee4278b859d63e7c75
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 23:48:31 2014 -0400

    remove pthread, add standard c++ thread

commit 2f2a3d9f9203ae3d7c9aee2bd6268d1940702eb5
Merge: 5acdaf7 f3f2f07
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 20:20:12 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 5acdaf7f5fd3b7a7560a17ccc73cdf8a7f454f38
Author: Qinfan Wu <wuqinfan@gmail.com>
```

```
Date:   Sun Mar 23 20:19:42 2014 -0400

    add subarray method for AvlArray

commit f3f2f070091663fd01496f902f2c258dc478fc83
Merge: ffa86f5 b610d7f
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Mar 23 20:08:43 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit ffa86f5d8f34c445b7becd46d7861f2436ffbd4f
Author: Shining Sun <shining.app@gmail.com>
Date:   Sun Mar 23 20:04:18 2014 -0400

    Bug fix.

commit b610d7f0cc78cdfb35aefc225cb40e3fbaf38dd2
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 18:37:21 2014 -0400

    In AvlArray, change pointers from T* to shared_ptr<T>

commit 6d5aff122991400aba77e766d552d09276d68a12
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 18:02:28 2014 -0400

    New implementation: every element in AvlArray is a pointer to an AvlObject.

commit 40a7f4c95ecec1dc23deaed61bc31c0c475f36c9
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 16:02:55 2014 -0400

    add delay in AvlVisualizer::addObject

commit c7fb66d4070aa000972a0a2d6f443ef2a0ca6b8a
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 15:41:54 2014 -0400

    add swap_element for AvlArray
```

```
commit 7bd8442c299e9f28c5c0da3e0fbb8cae08e0903a
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 13:14:45 2014 -0400

    overload relational operators for AvlInt

commit de2f1d50b171af5c8afcd46dffef7e9223be73ba
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 11:55:48 2014 -0400

    fixed the issue that multiple avl instances are sharing the same
    intermediate c++ file.

commit b6e5784ad915b673017d607bd9916e2884175a96
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 23 11:25:43 2014 -0400

    Update README.md

commit 0d944a5d6d65c7abe89072901b531698591b7612
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 11:19:42 2014 -0400

    bug fixed

commit 7d8d082d171e52648009e8b62196746c8a6cfea4
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 11:16:47 2014 -0400

    add delay value for AvlObject

commit 91261dbe6439625470b6537ae541cb31ef34d47d
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 11:07:03 2014 -0400

    add avlSleep()

commit cb8798fa1c1fc33afb6faf4adec65a63ff37a6b2
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 23 11:06:15 2014 -0400
```

```
    updated

commit 438281b2b6e803700df2630731e8a286a33f4c66
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 23 10:30:32 2014 -0400

    Update README.md

commit d91bd5f41d2739462b158c368ba4347e4413fed6
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 23 09:16:41 2014 -0400

    Update README.md

commit f7a35ed4db60c5d527fe0f894d4d422ca9def942
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 23 01:05:10 2014 -0400

    Update README.md

commit d95b3eb856be5ed92a962b05a88c477c81b691c9
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 23 01:04:18 2014 -0400

    Update README.md

commit 64058064d4f28448ca70469f052e6ca4f1eceb4e
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sat Mar 22 22:29:24 2014 -0400

    typo fixed

commit e4a7e9cb9e4cab68b757b7667fb20555fa63f5b1
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sat Mar 22 22:09:38 2014 -0400

    typo fixed

commit e49f4a7074b8cd55ba01b234f3d097da81cea8f0
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sat Mar 22 22:01:57 2014 -0400
```

```
    new scripts

commit 0d4f966bdf2a9e8a5f516b22e649c0a0a36a3b3b
Author: Shining Sun <shining.app@gmail.com>
Date:   Sat Mar 22 21:56:11 2014 -0400

    Four sample codes, by Shining Sun.

commit e1cd9382d81eec89d9342d744fed3d1477cfcb03
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sat Mar 22 21:03:35 2014 -0400

    Add mutex support for AvlObject

commit e43b64e75bc1a8561e7a8c54f368a2e6a73a2929
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sat Mar 22 19:57:57 2014 -0400

    bug fixed.

commit 5be3e2f7e19205bb348657eb5af6393dccb6fee2
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sat Mar 22 19:45:30 2014 -0400

    add test scripts for avl

commit 8727ce42fcf11cd8dd9e2fefd36bf6ee3487198d
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 23:12:08 2014 -0400

    test cases updated

commit 96bbaf03e6add8f2f709f3226b812244295c2185
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 21:08:34 2014 -0400

    bug fixed

commit 9a61d21fbfb8faf63c9b8c736ab714c6a707c142
Author: Qinfan Wu <wuqinfan@gmail.com>
```

```
Date:   Thu Mar 20 20:58:49 2014 -0400

    bug fixed

commit 2e6cd121e33b6f1583d21514c8cf806d915a086c
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 20:43:04 2014 -0400

    test scripts updated

commit 9418c050040180ed7efc25d046ccc89c158da883
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 20:39:39 2014 -0400

    More efficient random array generation.

commit 185ecc7f3bfed885a09f12e54df4554775d25c1c
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 20:23:26 2014 -0400

    test scripts updated

commit 9e36068c98aceec56bb116ffc1a6efd570b3ff20
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 19:34:06 2014 -0400

    test case updated.

commit 8d6ac7a01f566dbefd6ee81257074251efc7d206
Author: wqfish <wuqinfan@gmail.com>
Date:   Thu Mar 20 16:23:44 2014 -0400

    Update avl.c

commit 349d458b2f0610a4bf129f219999443210efb74a
Author: wqfish <wuqinfan@gmail.com>
Date:   Thu Mar 20 15:56:53 2014 -0400

    Update README.md

commit 1f7d27aeda0abf520f3d27cebb9e88d772ffc61c
```

```
Author: wqfish <wuqinfan@gmail.com>
Date:   Thu Mar 20 15:45:08 2014 -0400

    Update README.md

commit 9dcf0ec61aee3678d76bb18bae4894f2ff599f6b
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 15:39:00 2014 -0400

    Add new test case

commit 7a8a9d50339a02bc8a3635d8d3e7bf28b1ed4ebf
Author: wqfish <wuqinfan@gmail.com>
Date:   Thu Mar 20 15:08:24 2014 -0400

    Rename AvlInt_assign1.sh to avlint_assign1.sh

commit 69fffad8cd8af836bab74dbb1a1eabdafe61ce0b
Author: wqfish <wuqinfan@gmail.com>
Date:   Thu Mar 20 15:08:04 2014 -0400

    Rename AvlInt_assign1.cpp to avlint_assign1.cpp

commit dd75a87faaecdcd6f1399b88ae68c5361b4397ce
Merge: 8215a3d b4ca60a
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 15:06:59 2014 -0400

    Merge branch 'master' of https://github.com/wqfish/avl

commit 8215a3d50e8c81a0d1d0a906ae0c74ac29d3a040
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 15:06:35 2014 -0400

    add a test case

commit b4ca60a6a250e71467d8df4a805700bb3dc85a79
Author: wqfish <wuqinfan@gmail.com>
Date:   Thu Mar 20 14:51:59 2014 -0400

    Update README.md
```

```
commit 04e64705f7ae03b319138a4bf01abff5bcd4d2e4
Author: wqfish <wuqinfan@gmail.com>
Date:   Thu Mar 20 14:49:28 2014 -0400

    Update README.md

commit d20d1001d57a814875a2be8e4f4215dc66b83e6e
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 14:38:31 2014 -0400

    test case updated

commit 5b07ec526747e280135073dc1ee5776537a3f0bc
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Thu Mar 20 12:07:53 2014 -0400

    Add initial testsuites

commit b6d2df491da0e5cf17f5b1f4c3d15588807bb420
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Mar 19 16:56:26 2014 -0400

    bug fixed

commit 04089a9285f373acef5fbec3164ec315fa6f7695
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Mar 19 15:06:20 2014 -0400

    AvlArray updated

commit 9838beca37dc4c8cf6f7ee9189ba2ce2c878ca7c
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Wed Mar 19 10:54:32 2014 -0400

    Add partial array visualization support

commit 987b10fbc29dbb63e82ee888d9db8e932c4165db
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Mar 18 11:13:46 2014 -0400
```

```
    g++ options updated

commit 8f13f045348d67e694dbf564bb4e0d0bbab7a006
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Tue Mar 18 10:21:16 2014 -0400

    CFLAGS and CPPFLAGS updated.

commit 1e4d3c97634ee59cb8ea9b1e25056eafd54a782b
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 17 19:10:30 2014 -0400

    updated AM_CFLAGS and AM_CPPFLAGS in Makefile.ams.

commit 4a91337abd9aa66dc4215c5aa207b1add997aaba
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 17 16:21:23 2014 -0400

    Built shared library.

commit ed6a6873039b6690b5e19774a5df05290c71932a
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 17 11:35:23 2014 -0400

    operators of AvlInt updated.

commit a5e2dca1e78c4d6f90c91c0e2ff3904daefd39a7
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Mar 17 00:34:15 2014 -0400

    Basic opengl implemented!

commit 7493ee69f0b2edf7e87e81e8dc678aa92f3c6e15
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 16 23:04:41 2014 -0400

    Added object rendering.

commit 5d6ba5d98f6e494b9011c0b82f91b6c014d2e0c5
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 16 17:05:45 2014 -0400
```

```
    Added AvlObject and AvlInt classes.

commit 76a152237f1bc7cf3333ac46616599a5c09c9fdd
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 16 15:58:09 2014 -0400

    define -> const

commit 6307fa34bf44dd4e82a0fb68d0bceaebf318a06b
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 16 15:12:26 2014 -0400

    Update README.md

commit 80c37deeb72e895b8e837ec3729fc80eacf99d03
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 16 15:08:46 2014 -0400

    Update README.md

commit 0cf338d3f0da4668d9a6d85f7f24873ca3972788
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 16 15:05:24 2014 -0400

    Update README.md

commit 4f80b9b773d3739780eb2f769052db6e3bba667b
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 16 15:01:23 2014 -0400

    Update README.md

commit 2de3fec2f3b5123aa90ae8df1eaa60838b36c712
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 16 15:00:25 2014 -0400

    Update README.md

commit dcfdfff2c64baf4c1dbf644180fffd3074578938
Author: wqfish <wuqinfan@gmail.com>
```

```
Date:   Sun Mar 16 14:58:42 2014 -0400

    Update README.md

commit ec0e18ddbfa887ccb24332d8f76d7e0c7867cf40
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 16 14:57:35 2014 -0400

    Update README.md

commit 956c93487c72a1c3a57d3505d5c463264bba2b95
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 16 14:54:52 2014 -0400

    Update README.md

commit 34441f408b5c6bca997078502a54544be079ed80
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 16 14:51:15 2014 -0400

    README updated.

commit 565793d67b3798ea87f8473ba520b3f40dcb2e97
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 16 14:22:11 2014 -0400

    updated

commit be135fbcd76bdc16b4b0b2ce34a99f9298eb3cf1
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 16 14:20:13 2014 -0400

    Updated Makefiles.

commit 5d91174e5135327ee6efa4013b43e80433cd3268
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 16 14:18:44 2014 -0400

    Added avl libs.

commit c19d8efae0f8acc216c6b8f09abff38a4df61955
```

```
Author: wqfish <wuqinfan@gmail.com>
Date:   Fri Mar 14 20:54:18 2014 -0400

    updated.

commit 5aaf691d9a77240e3b01354dcd3f4ae312209364
Author: wqfish <wuqinfan@gmail.com>
Date:   Fri Mar 14 19:03:02 2014 -0400

    Bug fixed.

commit 1bf2fdcd7e1dab5dbafe5c59e5cb93ae6ace0fb5
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri Mar 14 15:24:53 2014 -0400

    updated

commit e897611f7163992cfce6cd46d0e91817161568f8
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri Mar 14 15:07:53 2014 -0400

    updated

commit dd8383e53b885914360b04a0a7ab18cbfa1910db
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri Mar 14 15:00:21 2014 -0400

    Bug fixed.

commit 2eaaa0507a4ee2a3567f7b22baa7eb1e18dd506e
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri Mar 14 14:43:46 2014 -0400

    Added g++ for compiling the parsing result.

commit df0042a959f4903ede3b7dedc96202077e9b9e48
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri Mar 14 14:08:54 2014 -0400

    Change input/output from stdin/stdout to files.
```

```
commit 85c0a48cbb1b6fb014d23e6238c920fb5215e13f
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Fri Mar 14 13:32:48 2014 -0400

    Command-line parsing updated.

commit 53f67db5fe9d37974609e658e7dbc8b07a58967c
Author: wqfish <wuqinfan@gmail.com>
Date:   Fri Mar 14 02:01:12 2014 -0400

    Move command-line parsing into a single function.

commit 45ec98e02104be7b9f18a1936923305a155dbc78
Author: wqfish <wuqinfan@gmail.com>
Date:   Fri Mar 14 00:25:13 2014 -0400

    Added command-line options parsing.

commit 4bf0f2ad3e00423388674cc2f8404a485356b6c8
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 9 19:43:07 2014 -0400

    Modified readme.

commit 06e04a99af06af326e6d9d90bd7726ed89c0880a
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Sun Mar 9 19:41:33 2014 -0400

    Added build instructions.

commit 786029dfcc339d034eca05aa4538299d32fbe991
Author: wqfish <wuqinfan@gmail.com>
Date:   Sun Mar 9 19:26:15 2014 -0400

    Build environment initialized.

commit 41d4abe016878ab7de14899b6f821801ee3c9887
Author: Qinfan Wu <wuqinfan@gmail.com>
Date:   Mon Feb 24 19:43:13 2014 -0500

    Added main function.
```

```
commit 1092302511c8c744001af4e85c78a01c5e4b5488
Author: wqfish <wuqinfan@gmail.com>
Date:   Mon Feb 24 16:40:23 2014 -0800

    Initial commit
```