COMS 4115: PROGRAMMING LANGUAGES AND TRANSLATORS

PAL: PDF AUTOMATION LANGUAGE

Anshuman Singh(as4916)
Diksha Vanvari(dhv2108)
Vinay Gaba(vhg2105)
Viral Shah(vrs2119)

Contents

Introduction **4**

Language Tutorial **5**

Program Structure 5
Import Statements 5
Program Compilation 6
Command Line Interface 6
File Compilation 6
Program Execution 7
Lexical Conventions 7
Identifiers 7
Keywords 7
Punctuation 8
Literals 8
Comments 9
Data Types 9
Primitive Data Types 9
List Data Type 11
Map Data Type 12
Predefined Constructs 14
Operators 15
Arithmetic Operators 15
Logical and Relational Operators 16
Special Operators 18
Expressions 19
Literals 19
List Access 19
Map Access 19
Binary Operator 20

Unary Operators . . . 20
Negation . . . 20
Function Call . . . 20
Nested Expressions . . . 20
Statements . . . 21
Declarative Statements . . . 21
Assignment Statement . . . 21
Function Call Statements . . . 21
Control Flow Statements . . . 22
Predefined Functions . . . 24
Function Definition and Declaration . . . 28
Function Call . . . 29
Scoping . . . 29
Standard Library . . . 30

Architectural Design **33**
Scanning . . . 34
Parsing . . . 34
Analysis . . . 34
Code Generation . . . 34
Run . . . 35

Project Plan **36**
Project Processes . . . 36
Planning . . . 36
Specification . . . 36
Development . . . 36
Testing . . . 37
Style Guide . . . 37
Team Responsibilities . . . 37
Project Timeline . . . 37
Development Environment . . . 38
Project Log . . . 38

Test Plan **87**

Lessons Learned **89**

Anshuman Singh 89

Diksha Vanvari 89

Vinay Gaba 89

Viral Shah 89

Appendix **90**

1. Introduction

Portable Document Format (PDF) is the file standard for the electronic exchange of documents. According to estimates by Adobe executives, there might be up to 2.5 trillion PDF documents existing in the world. The reason for its popularity is its platform-agnostic behaviour of passing and sending information that wonâĂŹt be skewed or altered. Our aim is to expand the range of operations performed on this popular data source through the means of PAL. There are many solutions available which are similar in nature to PAL but very often they do not fulfill the exact functionality as needed and are generally complicated, which requires a learning curve. We intend to simplify these interactions with PAL while at the same time also enable powerful operations which can fulfill operational needs.

2. Language Tutorial

2.1 Program Structure

Every program in the language must be structured, comprising of import statements (optional), the main function (necessary), and function declarations (optional). When a library is imported by another program, the function declarations made in the library are imported into the current program. However the main function of the imported library is ignored and the overwritten by the main function of the importing program. For nested import statements, the function declarations of each imported library are imported, and the main function is overwritten by that of the primary importing program.

```
import ("stdlib.pal");
import ("userlib.pal");

main() {
    i : int;
    i = func(i);
}
func (i : int) {
    return i;
}
```

2.2 Import Statements

Import Statements are used to import the standard library that the language provides, as well as user defined libraries. If there are any import statements, then they must appear at the beginning of program. It makes all the functions from the imported module accessible in the current program, and the functions can be accessed without prepending the module name. Import Statements are included as follows:

```
import ("stdlib.pal");
import ("userlib.pal");
```

2.3 PROGRAM COMPILATION

Command Line Interface

A PAL Program can be written using the command line interface. Every input line is parsed and syntactically checked at input time. At the end of input, the written program is semantically analyzed and the java code is generated for the same on successful compilation.

The Command Line Interface can be invoked using the following command:

```
$pal -j

The program can then be input line by line:

import ("stdlib.pal");

main() {
    i : int;
    i = func(i);
}

func (i : int) {
    return i;
}
```

File Compilation

A PAL program can be written as a '.pal' file. The '.pal' file can be compiled using the compile utility provided by the language. On successful compilation, a PAL program generates the corresponding '.java' file.

The Compile Utility can be invoked as:

```
$pal -f program.pal
```

2.4 Program Execution

A successfully compiled PAL program generates the corresponding '.java' file. The generated '.java' file can be executed using the run utility provided by the language.

The Run Utility can be invoked as:

```
$run -o output.java
```

2.5 Lexical Conventions

Identifiers

Identifiers are a sequence of letters, digits and underscores. All identifiers must begin with a letter and not use any reserved keywords.

```
pdfvar : pdf;
pagevar : page;
list1 : list int;
list2 : list string;
english_dictionary : map string,string;
```

Keywords

The following identifiers are reserved as keywords and cannot be used in any other manner:

import	main	function
int	boolean	float
line	tuple	true
import	main	function
false	list	map
image	if	else
for	while	return
continue	break	length
getpages	split	readtable
readtextfrompdf	drawpiechart	drawbarchart
loadpdf	readtextfile	renderpdf

Punctuation

Punctuations are special characters which are neither operators nor identifiers. They have their own significance.

: -> type declarator ; -> statement end , -> argument delimiter âĂIJâĂİ -> string delimiter

Literals

A literal is a notation for representing a fixed value in code.

Integer Literal

A positive integer is 1 or more digits from 0 - 9. A negative integer is a âĂŸ-âĂŹ followed by 1 or more digits from 0 - 9. Zero is neither positive nor negative and is represented as 0.
INT = "['0'-'9']+ | '-'['1'-'9']['0'-'9']+"

Float Literal

A float literal consists of an integer part, followed by a point '.' followed by the fractional part. The float literal can either be positive or negative.

FLOAT = ['+"-']?['0' - '9']* '.' ['0'-'9']*

Boolean Literal

A boolean literal can take only two values - true or false.

BOOL = "true|false"

String Literal

A string literal is zero or more ASCII characters written between two double quotes. "
n", "
r", "
t", ", ',
are preceded with an escape sequence character '
'.
STRING = "
"(([' '-'!' '#'-'[' ']'-' '] | '
['
'' 'n' 'r' 't']))*"

Comments

Only single line comments are supported, which are identified by #.

```
# This is a single line comment.
```

2.6 DATA TYPES

Primitive Data Types

boolean

Maybe true or false. Boolean types can only be used with other boolean types, any other operation involving boolean types fails.

```
boolvar : bool = true;
```

int

An integer literal such as 5664 is a 32-bit signed integer. It takes values in the range from -2,147,483,648 to 2,147,483,647.

```
intvar : int = 42;
```

float

A float literal has an integer part followed by a fraction part. It is a 64-bit signed float.

```
floatvar : float = 42.0;
```

string

A string literal is a sequence of ASCII characters. They are enclosed in double quotes, with special characters escaped with a backslash.

```
stringvar1 : string = "This is a string."
stringvar2 : string = "This is \"Hello\" from the other side.
```

pdf

A pdf type represents a logical representation of a physical âĂIJPDFâĂİ document.

```
pdfVar : pdf;
```

page

A page type represents a logical representation of a physical âĂIJPDFâĂİ page. A pdf document consists of 0 to an arbitrary number of lines.

```
pageVar : page;
```

List Data Type

List Declaration

A list can be declared in the same manner as declaring a primitive data type, but the keyword list must be prefixed to the data type of which the list is an ordered collection List Declaration:

```
intlist : list int;
stringlist : list string;
intlist_ : list list int;
stringlist__ : list list list string;
pdflist : list pdf;
tuplelist : list tuple;
```

List Access

An element of a list can be accessed, and can be assigned or assigned to another variable or literal of the same type as the list element or passed as an argument to a function accepting the same type as that of the list element.

```
i : int;
i = 1;
intlist : list int;
intlist[0] = i;
i = intlist[0];
i = func(intlist[0]);
```

List Addition

An element can be added to a list, provided that it is of the same type as that of which the list is an ordered collection.

```
i : int;
i = 1;
intlist : list int;
```

```
intlist[0] = i;
intlist_ = list list int;
intlist_ += intlist;
```

List Removal

An element can be removed from a list, by mentioning the index of the element, which is to be removed, in the list. This index can be specified using a valid integer expression.

```
i = 1;
intlist : list int;
intlist[0] = i;
intlist_ = list list int;
intlist_ += intlist;
intlist_ -= [0];
```

Map Data Type

Map Declaration

A map can be declared in the same manner as declaring a primitive data type, but the keyword map must be prefixed to the data types of which the map is a dictionary. The data types of the key and the value must be separated by a comma.

```
int_int_map : map int,int;
int_string_map : map int,string;
page_listline_map : map page,list line;
```

Map Access

A value of a key can be accessed, and can be assigned to another variable or literal of the same type as the type of the value or passed as an argument to a function accepting the same type as that of the valuet. A map can be accessed using a key present in the map.

```
i : int;
i = 1;
s : string;
int_string_map : map int,string;
s = int_string_map:=i;
```

Map Addition

An key value pair can be added to a map, provided that the pair is of the same type as that of which the map is a dictionary.

```
i : int;
i = 1;
s : string;
s = âĂIJvalueâĂİ;
int_string_map : map int,string;
int_string_map += i,s;
```

Map Removal

An element can be removed from a map, by mentioning the key, which is to be removed, in the map. This key must be of the same type as that of the dictionary key type of the map.

```
i : int;
i = 1;
s : string = âĂIJvalueâĂİ;
int_string_map : map int,string;
int_string_map += i,s;
Int_string_map -= i;
```

Predefined Constructs

line

A line type represents the lowest level of physical space which is used to draw strings on a page. Line accepts a string along with fonts style as string, size as integer, left and top margin as integers, and the width of line as an integer. Based on the values of input parameters and the pdf configurations, the line variable stores an index which points to the last position of the string which has been written out to the pdf.

```
lineVar : line(string, string, int, int, int, int);
```

tuple

A tuple represents the association of a pdf with a page. Before using a pdf and a page as part of a tuple, they need to be defined and the page needs to be added to the pdf, otherwise the construct gives an error.

```
pdfVar : pdf;
pageVar : page;
tupleVar : tuple(pdfVar,pageVar);
```

image

An image is a high level construct that lets you hold a representation of a jpeg/png image. It will also be helpful in generation of charts that would also be returning an image of the chart which can then be placed in the pdf. Image accepts the name, width and height as integers and left and top margins as integers.

```
imageVar : image(string, int, int, int, int);
```

2.7 OPERATORS

Arithmetic Operators

An arithmetic operator is a token that would manipulate the value of integer or floating point operands(s).

```
Operator '+':
Addition
i : int;
i = 1 + 2;
f : float;
f = 1.0 + 2.0;

Operator '-':
Subtraction
i : int;
i = 2 - 1;
f : float;
f = 2.0 - 1.0;

Operator '*':
Multiplication
i : int;
i = 1 * 2;
f : float;
f = 1.0 * 2.0

Operator '/':
Division
i : int;
i = 2 / 1;
f : float;
f = 2.0 / 1.0;
Operator '%':
Modulo
i : int;
```

```
i = 2 % 1;
f : float;
f = 2.0 % 1.0;

Precedence:
'* / %''
'+ -'
```

Logical and Relational Operators

A relational operator is used to determine how two operands compare to each other. It can be used to test for equality, inequality, and comparison of operand values. A logical operator is used to perform logical operations on operands.

```
Operator '==':
Equality
i1 : int;
i2 : int;
i1 = 2;
i2 = 1;
if (i1 == i2) {
    i1 = i1 + 1;
}

Operator '!=':
Inequality
i1 : int;
i2 : int;
i1 = 2;
i2 = 1;
if (i1 != i2) {
    i1 = i1 + 1;
}
Operator '>':
Greater Than
i1 : int;
```

```
i2 : int;
i1 = 2;
i2 = 1;
if (i1 > i2) {
    i1 = i1 + 1;
}

Operator '>=':
Greater Than or Equals
i1 : int;
i2 : int;
i1 = 2;
i2 = 1;
if (i1 >= i2) {
    i1 = i1 + 1;
}

Operator '<':
Lesser Than
i1 : int;
i2 : int;
i1 = 2;
i2 = 1;
if (i1 < i2) {
    i1 = i1 + 1;
}

Operator '<=':
Lesser Than or Equals
i1 : int;
i2 : int;
i1 = 2;
i2 = 1;
if (i1 <= i2) {
    i1 = i1 + 1;
}
```

```
Operator '!':
Logical NOT
b1 : bool;
b2 : bool;
b1 = true;
b2 = false;
if (!b1) {
    b2 = true;
}

Operator '&&':
Logical AND
b1 : bool;
b2 : bool;
b1 = true;
b2 = false;
if (b1 && b2) {
    b2 = true;
}

Operator '||':
Logical OR
b1 : bool;
b2 : bool;
b1 = true;
b2 = false;
if (b1 || b2) {
    b2 = true;
}
```

Logical and relational operators have a lower precedence than arithmetic operators.

Special Operators

```
Operator '+':
Concatenation
i : int;
```

```
s : string;
r1 : string;
r2 : string;
r1 = s + i;
r2 = i + s;
Operator '.':
Concatenation
pdfVar : pdf;
pageVar : page;
lineVar : line(âĂIJThis is a lineâĂİ, âĂIJTIMES_ROMANâĂİ, 18, 10, 10, 60);
imageVar : image(âĂIJImg.pngâĂİ, 1920, 1080, 20, 20);
tupleVar : tuple(pdfVar,pageVar);
tupleVar = tupleVar.lineVar;
tupleVar = tupleVar.imageVar;
```

2.8 EXPRESSIONS

Literals

```
i : int;
i = 2;
```

List Access

```
i : int;
i = 1;
intlist : list int;
intlist[0] = i;
i = intlist[0];
```

Map Access

```
i : int;
i = 1;
```

```
s : string;
int_string_map : map int,string;
s = int_string_map:=i;
```

Binary Operator

```
i : int;
i = 1 + 2;
```

Unary Operator

```
i : int;
i = 1 + 2;
i = -i;
```

Negation

```
b1 : bool;
b2 : bool;
b1 = true;
B2 = !b1;
```

Function Call

```
b1 : bool;
b2 : bool;
b1 = negate(b2);
```

Nested Expressions

```
i : int;
i = (1 * 2) + (3 * 4);
```

2.9 STATEMENTS

Declaration Statements

A declaration statement specifies the name and datatype of the variable being declared. In addition, it may also initialize the variable.

```
stringvar : string;
stringvar : string = "Initialized.";
```

Some special data types(such as lines, tuples and images) are declared and initialized in the following manner:

```
linevar : line(stringVar, "TIMES_ROMAN" , 12 , 100, 700, 500);
```

Assignment Statements

The assignment statement comprises of an expression on the right hand side of an assignment operator which evaluates the value denoted by the expression and assigns it to the variable on the left hand side of the equals sign.

```
stringvar : string = str \âĂŘ 15;

listvar : list string;
listvar += "Hello";
listvar -= "Hello";

mapvar : map string,string;
mapvar += "Hello", "World";
```

Function Call Statements

The function call statement is used for function calls which donâĂŹt return any value.

```
renderpdf("This is a test string", "test-funccall1.pdf");
```

Control Flow Statements

Conditional execution of partial blocks of code is enabled using control flow statements that facilitate decision making, looping and branching. The decision making statements include the if, else, elif statements. The looping statements include the for and while statements. The branching statements include the break, continue and return statements.

If, Else, Elif

These statements enable conditional execution of partial blocks of code by evaluating the given condition and executing the corresponding block of code. If the condition is true, then it executes the statements in the first block as limited by the parentheses, else it executes the statements in the second block as limited by the parentheses.

For, While

These statements enable conditional execution of partial blocks of code by evaluating an expression against a given condition, and executing the corresponding block of code if the condition is true.

For Loop

While the condition mentioned by the second expression is true, the loop continues iterations, each time executing the statements in the block following 'do' limited by the '' parentheses.

```
i :int;

for ( i = 1 ; i <= 10 ; i = i + 1)
{
    <statement1>
    <statement2>
}
```

While Loop

While the condition mentioned by the expression is true, the loop continues iterations, each

time executing the statements in the block following 'do' limited by the '' parentheses.

```
i : int = 1;
n : int = 10 ;
while ( i != n )
{
    <statement1>
    i = i + 1;
}
```

Break, Continue, Return

These statements enable conditional execution of partial blocks of code by specifying the termination of execution of the corresponding block of code.

Break

A break statement within a 'for' or 'while' statement terminates the looping of the innermost looping statement it is nested within.

```
i : int = 1;
n :int = 10;

while ( i != n )
{
    <statement1>
    if (< condition1 >)
        {
            <statement2>
            break;
        }
    i = i + 1;
}
```

Continue

A continue statement within a 'for' or 'while' statement skips the current iteration of the innermost looping statement it is nested within, skipping to the end of it and evaluating the conditional expression that controls the loop.

```
i : int = 1;
n :int = 10;

while ( i != n )
{
    <statement1>
    if (< condition1 >)
        {
            <statement2>
            break;
        }
    i = i + 1;
}
```

Return
A continue statement exits from the current method and the control flow returns to the point of function invocation. The return statement is followed by a return value of the type indicated in the function definition.

```
return result;
```

2.10 PREDEFINED FUNCTIONS

length()
The length function is an overloaded function that takes either a string, list or a map as input and returns the number of characters in the string, number of elements in the list or the number of key value pairs in the map respectively.

Input parameters:
string – The string whose length needs to be returned map – The map whose size needs to be returned
list – The list whose size needs to be returned

Return Type:
int – The size of the datatype returned as an integer
getpages()
The getpages function takes a pdf as input and returns a list of pages from the pdf.

Input parameters:
pdf – the pdf from which you want to extract pages

Return Type:
list page – the function returns a list of pages of the pdf
split()
The split function helps in splitting a pdf file into multiple pdf's.

```
pdfList = split(pdf , listVar );
```

Input parameters:
pdf – The pdf file that needs to be split
list – A list of integers that specify which page numbers to split the pdf on

Return Type:
list pdf– A list of pdf files that were split from the original pdf
readtable()
The readtable function helps in reading a table from a specified location in a pdf file.

```
data = readtable(pdflocation , pagenumberlist );
```

Input parameters:
pdflocation – The pdf file that needs to be split
pagenumberlist – A list of integers that specify which page numbers to split the pdf on

Return Type:
list list string – Every row is a list and values of the columns together make up a list of string.
readtextfrompdf()
The readtextfrompdf function helps in reading text from a specified location in a pdf file.

```
data = readtextfrompdf ( pdflocation , pagenumberlist );
```

Input parameters:
pdflocation – The pdf file that needs to be split
pagenumberlist – A list of integers that specify which page numbers to split the pdf on

Return Type:
string – The string representation of the text content in the pdf file.

drawpiechart()

The drawpiechart function helps in creating a pie chart from a datalist passed to it.

```
data = drawpiechart ( datalist, attributemap );
```

Input parameters:
datalist– list list string of data
attributemap – map of the pie chart properties

Return Type:
image – The image representation of the data.

drawbarchart()

The drawbarchart function helps in creating a pie chart from a datalist passed to it.

```
data = drawbarchart ( datalist, attributemap );
```

Input parameters:
datalist– list list string of data
attributemap – map of the bar chart properties

Return Type:
image – The image representation of the data

loadpdf()

The loadPDF function helps in loading a PDF file into the program.

```
pdfvar : pdf;
```

```
pdfvar = loadPDF (" Area52 . pdf ");
```

Input parameters:
string – location of the file

Return Type:
string – pdf object of file

readtextfile()
The readtextfile function helps in reading text from a specified location in a pdf file.

```
data = readtextfile ( pdflocation , pagenumberlist );
```

Input parameters:
pdflocation – The pdf file that needs to be split

Return Type:
string – The string representation of the text content in the pdf file.

renderpdf()
The renderpdf function takes in two arguments, a pdf and a disk location and saves this pdf to the specified location.

```
data = readtextfile ( pdflocation , pagenumberlist );
```

Input parameters:
pdf – the pdf that you want to save
string – the disk location you want to save the pdf to

Return Type:
No return value is associated with renderpdf

print()
The print function accepts a type as input and prints out the string representation of the type.

```
print "ABC";
```

Input parameters:
string – the value that needs to be printed to the output stream

Return Type:
No return value is associated with renderpdf

substr()
The substr function accepts a type as input and prints out the string representation of the type.

```
stringVar = substr ( stringVar , startIndex, endIndex);
```

Input parameters:
stringVar – the string variable from which the string needs to be extracted
stringIndex – the start index for the start of the string to be returned
endIndex – the end index for the end of the string to be returned

Return Type:
string – the string returned between the start and end index.

2.11 FUNCTION DEFINITION AND DECLARATION

Users can define their own function in PAL. A function is declared and defined at the same time. A function is defined by specifying the function name followed by the function parameters and finally the function return type. Functions are declared by specifying a list of statements after the function definition. Functions must be defined before they are used, however a function may recursively itself.

```
function_name (parameter : parametertype) : returntype {
   <statement1>
   <statement2>
```

```
   return result;
}
```

2.12 FUNCTION CALL

Functions are called using the following syntax. Users can specify the function name and the list of input parameters. Once a function is called, the program execution is halted until the function execution is completed.

```
variable : type;
variable = function_name (parameter : parametertype);
```

2.13 SCOPING

Scope refers to the variables and functions available at any given instance in the program. All variables are local and are available within the function in which they have been declared. In addition, a variable declared within a given block of code is available only within the scope of that block. The scope of a block is limited by the surrounding parentheses. Thus, variables declared within a control flow statement is available only within the scope of the control flow statement, as opposed to variables declared in the beginning of a function definition, which are available throughout the function body. If more than one variable is declared with the same identifier, then the variable declared in the most nested scope, limited by parentheses, prevails the variable declared before it, within the scope. A function is available throughout the file in which it has been defined and can be invoked by another function irrespective of the order of function definition. Every program must have a main function. The program starts execution from the main function. On successful execution, the main function returns an integer value as specified in the return statement, else it returns -1.

```
main() {
    <statement>

    result : int;
    result = getResult(<expression>);
```

```
    n : int = 10;
    while (result != n) {
        <statement>
        result = result + 1;
    }

    return 0;
}


getResult (parameter : int) : int {
    <statement>
    <statement>

    return parameter;
}
```

2.14 STANDARD LIBRARY

The standard library provides high level constructs which take away the pain of writing the pdf with low level constructs like line. We have the following functions defined in the standard library:

1. Write Paragraph

```
write_paragraph(tupleVar : tuple, stringVar : string, startMargin : int,
    startHeight : int, fontSize : int, fontType : string, endHeight : int,
    width : int) : string
```

write_paragraph is a function that accepts a tuple, the content to be written, the size of the margin on the left, the y-coordinate where we want to start writing, the font and font size and the y-coordinate where we want to end the paragraph. We also specify the width of the lines to be written in the paragraph. This function writes the content and returns the content that could not fit in the paragraph.The function is a wrapper over the line construct allowing us to write a whole chunk instead of a line at a time.

2. Write Two Column Layout

```
write_two_column_layout(tupleVar:tuple, stringVar:string,
    fontType:string, fontSize:int) : string
```

write_two_column_layout is a function that accepts a tuple, the content to be written, font and font size and writes the content in a two column layout on the page that is bound with the tuple. The programmer need not specify the X and Y coordinates of where to start writing and where to stop. The function writes all the text it can in two columns and returns the remaining content.

3. Write Three Column Layout

```
write_three_column_layout(tupleVar:tuple, stringVar:string,
    fontType:string, fontSize:int) : string
```

write_three_column_layout is a function that accepts a tuple, the content to be written, font and font size and writes the content in a two column layout on the page that is bound with the tuple. The programmer need not specify the X and Y coordinates of where to start writing and where to stop. The function writes all the text it can in two columns and returns the remaining content.

4. Write 4 Grid Layout

```
write_4grid_layout(tupleVar:tuple, stringVar:string, fontType:string,
    fontSize:int): string
```

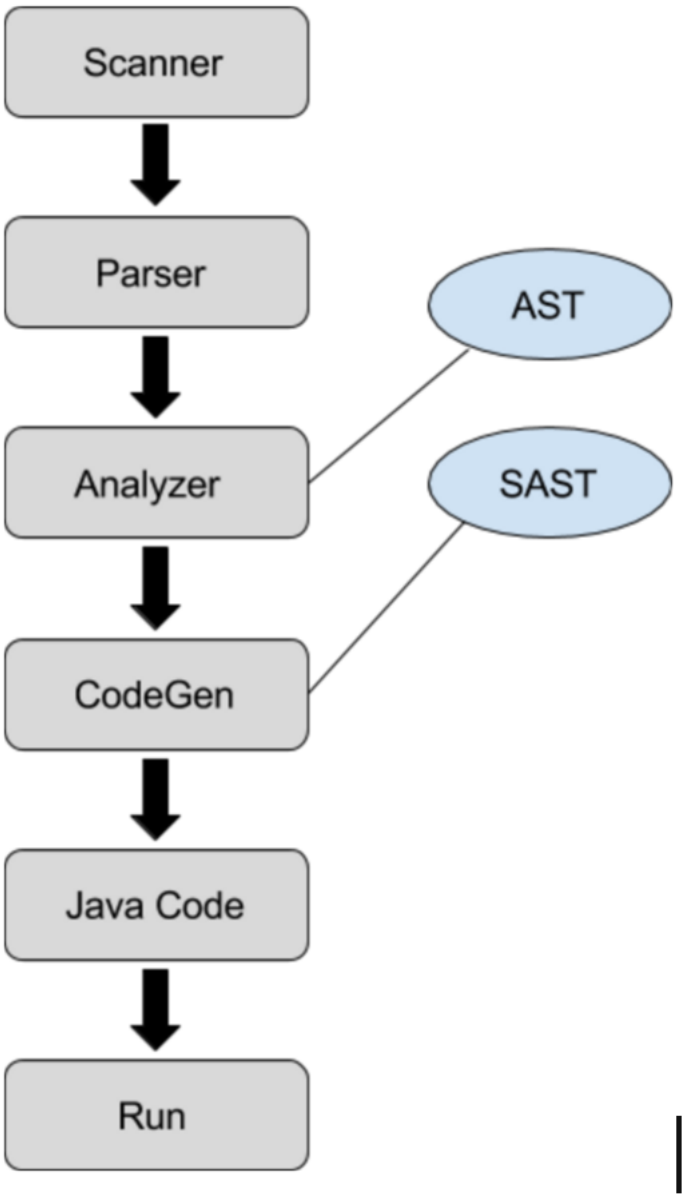
write_4grid_layout is a function that accepts a tuple, the content to be written, font and font size and writes the content in four grids on the page that is bound with the tuple. The programmer need not specify the X and Y coordinates of where to start writing and where to stop. The function writes all the text it can in 4 grids and returns the remaining content.

5. Write Pages

```
write_pages(stringVar : string, fontSize : int, fontType : string,
    layoutType:string) : pdf
```

write_pages is a function that accepts the content to be written, font, font-size and layout and writes the content in the specified layout on as many pages as it takes to fit the content. All these pages are added to a pdf and the function returns the pdf.

3. Architectural Design



The compilation and execution steps can be divided as follows:

- Scanning
- Parsing

- Analyzing
- Code Generation (Java)
- Running (Java)

Scanning, parsing and analyzing were handled by Anshuman and Viral, while Code Generation and Java Implementation were handled by Diksha and Vinay.

3.1 SCANNING

The scanner tokenizes the input into PAL readable units. This process involves discarding whitespace and comments. Illegal character combinations are caught. The scanner was written with ocamllex.

3.2 PARSING

The parser generates an abstract syntax tree (AST) from the tokens provided by the scanner. Syntax errors are caught here. The parser was written with ocamlyacc. The AST describes the statements and their associated expressions, but it is not typesafe.

3.3 ANALYSIS

The analyzer walks the abstract syntax tree produced by the parser, and generates a typesafe, semantically checked abstract syntax tree (SAST). This process detects all type mismatches, including the passing of wrongly-typed parameters and bad assignments. The semantic checking portion checks for other errors, such as scope errors, and the reassignment of special functions.

3.4 CODE GENERATION

The code generator walks the SAST produced by the analyzer and generates java code corresponding to the program. While this module generates java code, it does not compile it; that is achieved by the run utility.

3.5 RUN

The generated '.java' file can be executed using the run utility provided by the language.

4. Project Plan

Without careful design, planning and organization, the language and its compiler would surely be doomed. Heeding the warning, we started early and outlined a roadmap and simple procedures that helped us to successfully drive this project to completion.

4.1 Project Processes

Planning

In order to ensure that all of our team members were in sync, and that we agreed upon common deliverables, we consistently met each week, on Tuesdays and Fridays and sometimes additionally over the weekend. During our meetings, we focussed on design considerations, interfaces and on resolving challenges faced during implementation. We would conclude the meetings with a glance at our roadmap, and with a plan for the next week. The whiteboard in the apartment had never been put to such extensive use.

Specification

Our Language Reference Manual became the running specification for our language. As we proceeded with the implementation details, we faced certain challenges that made us revisit our language design and constructs. With each revision of the reference manual and with each iteration of the compiler, our language grew.

Development

The development took part in three phases. Having defined the language and its constructs, the first phase involved the implementation of a scanner and a conflict free parser. In the second phase, we developed the semantic analyzer and the code generator in parallel, agreeing upon the interface. The third phase involved the robust testing of a newly implemented feature, finding and working on areas of improvement.

Testing

At a lower level, the code is well documented and annotated with debug logs, in order to test errors and regressions. The parser, the analyzer and the code generator are rich with debug information to ensure quick debugging and regression testing. At a higher level, our integration test suite offers testing capabilities for our language and is described in greater detail in Section 5.

4.2 Style Guide

We used the following rules when writing our code to ensure maximum readability:

1. Each line of code should remain under 100 characters
2. Write utility functions for commonly reused code
3. Use camelcase for function and identifier names

4.3 Team Responsibilities

Divide and conquer was essential in driving the project to success. In order to facilitate splitting up work, we first identified our interfaces between modules, and then we made sure to get everyone onboarded with git hosted by GitHub.
We split up our group into two smaller focused teams:
Front End: Anshuman, Viral - Scanner, Parser, Analyzer
Back End: Diksha, Vinay - Java Code Generation, Java Libraries and Utilities, Testing

4.4 Project Timeline

The project timeline was carefully laid out:
Feb 10th - Project Proposal Due
Feb 20th - PAL Syntax Created
Mar 2nd - Scanner/ Parser Unambiguous
Mar 7th - Language Reference Manual Due
Mar 23rd - Architectural Design and Basic Features
Apr 6th - Hello World Compiles

Apr 30th - Integration Tested
May 6th - Final Project Report
May 9th - Final Project Slides
May 11tth - Final Project Due

4.5 Development Environment

1. We developed on a variety of environments including Mac OS X and Ubuntu.
2. We used OCaml version 4.00.1, OCamllex, and OCamlyacc for the compiler itself.
3. We used git hosted on github for version control.
4. We used shell scripts and makefiles to ease the work of compiling and testing the code.

4.6 Project Log

```
commit f8462066ce7518da8b71a030aa61b5a1b894357c
Merge: c7adb72 e3cbb7d
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Wed May 11 01:30:28 2016 -0400

    Merge pull request #59 from vanvaridiksha/master

    Cleaned final demo code

commit e3cbb7d2aa649d584195c9f49a60565de00bc453
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Wed May 11 01:27:39 2016 -0400

    Cleaned final demo code

commit c7adb72cb1e42b32e12093b7224331f331b9eb2a
Merge: 0700916 aad816f
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Wed May 11 01:16:53 2016 -0400
```

```
    Merge pull request #58 from vanvaridiksha/master

    Added final demo program. Fixed bugs in codegen and analyzer

commit aad816f2291f1edcc796e3340bb1f1d66e9ddd74
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Wed May 11 01:16:33 2016 -0400

    Added final demo program. Fixed bugs in codegen and analyzer

commit 070091680c1c825fe87d467f3542083b9384aa86
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue May 10 19:51:26 2016 -0400

    Made changes to the finaldemo

commit 2ac1fa12d860ee099ebd5ddca20ae710ea0f3eb5
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Tue May 10 00:07:58 2016 -0400

    Added Print Command

commit f35596f3dd3f7515b73e698c0995f93f648f72f4
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Mon May 9 22:36:34 2016 -0400

    Fixed Function without Arguments

commit e245c71011484ce26281c061f3af0fd71525a1fa
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Mon May 9 22:01:04 2016 -0400

    String and Int Concatenation

commit 63c90549b03d7a29a277dd684e1a932183cb1966
Merge: 673e807 b526be8
```

```
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Mon May 9 21:44:56 2016 -0400

    Merge pull request #57 from ANSSIN/master

    Added list init and substr

commit b526be88415f8fd913b426a057e88f897e792883
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Mon May 9 21:44:18 2016 -0400

    Added list init and substr

commit 673e8074b8599a37727654a727491b90b31f07ba
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun May 8 17:36:27 2016 -0400

    Added TestCase: test-writepage1.pal

commit 870feb7a41e80d70c72d37de068bbcad1e3deaec
Merge: 1ca2422 b428ae3
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sat May 7 21:38:19 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 1ca2422114ca7cde05b16b04cbac31135b0d3a48
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sat May 7 21:21:19 2016 -0400

    Added TestCase: test-writeparagraph1.pal

commit b428ae3d2534946cf1d0f3f862c22e6df4f8bde0
Merge: dc58436 9ab36cf
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Sat May 7 21:10:49 2016 -0400
```

```
    Merge pull request #56 from ANSSIN/master

    Added list remove and map remove test cases.

commit 9ab36cfafc2ab2a949ceafe2161ca8955b16a33f
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Sat May 7 21:10:05 2016 -0400

    Added list remove and map remove test cases.

commit dc58436df83148ef97df85b1f57aaedadd8a36dd
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sat May 7 20:36:59 2016 -0400

    Added TestCase: test-mapfind1.pal

commit 12b6e0d780d098a18b9319c37de313c37309fadf
Merge: 9127a52 2eeea45
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Fri May 6 16:09:11 2016 -0400

    Merge pull request #55 from ANSSIN/master

    Fixed negative numbers issue. Had to add UOP for negs in Parser.

commit 2eeea45b0eb14db525321afae38de9cb9a9800e9
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Fri May 6 16:07:59 2016 -0400

    Fixed negative numbers issue. Had to add UOP for negs in Parser.

commit 96d17cc0569d6c3cd3aa8e55f3295cb4952d6968
Merge: 4fbd76c 9127a52
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Fri May 6 16:04:31 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL
```

```
commit 9127a52f32aa207e15044614276feef4cb4a7df6
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Fri May 6 02:31:18 2016 -0400

    Added testcase- test-mapadd1.pal

commit 43f0781f789460073e36d6f72f6482deac4faae9
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu May 5 22:37:48 2016 -0400

    Added 3 test cases. Fixed column layout overflow error. Changed codegen
        list app method to call add() instead of append().

commit 0b7473132c49837b2ff0f5df2f003d7f382ef156
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu May 5 18:35:27 2016 -0400

    Added parameters to the write_pages method to accept page layout. Added PDF
        to init assign in codegen. Added demo pal file for write_page_demo

commit 397a94aa98960f2b392020973ef96ac4271d5c43
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu May 5 18:00:41 2016 -0400

    Implemented write_pages method to be able to write multiple pdfs

commit ae5e8e88ec017304b2aa276cf3a52ef7c302cc21
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu May 5 17:59:54 2016 -0400

    Fixed issue of parameters being reversed

commit 3c22355058fefc426f62180972cb3bcf781a118a
Merge: afd4fe6 0ad34f6
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Thu May 5 12:52:49 2016 -0400
```

```
    Merge pull request #54 from ANSSIN/master

    Fixed a major bug in List. Assigning the right types by merging the tâĂę

commit 4fbd76c635d2ef8f848caab754b876a5b46dd791
Merge: 0ad34f6 afd4fe6
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Thu May 5 12:52:10 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 0ad34f661eabfa5cd74667220b33214e93d163a6
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Thu May 5 12:51:34 2016 -0400

    Fixed a major bug in List. Assigning the right types by merging the type
        maps correctly and generating the right code.

commit afd4fe6a392712701e6584e48b93c600c308af24
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Thu May 5 12:10:10 2016 -0400

    Golden Program Change

commit 2e2c617aff9df82ea1385ecc732bba1ff07eea4d
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu May 5 11:52:53 2016 -0400

    Added write_pages_demo

commit e64f7ed7f2fc568327da136a1648f4066393663d
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu May 5 11:38:08 2016 -0400

    Added write_pages method in stdlib
```

```
commit 80463ead3fdf14659e51293a668a840f025c3b76
Merge: 5820dd8 578ec12
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Thu May 5 03:05:24 2016 -0400

    Merge pull request #53 from vanvaridiksha/master

    Added getpages and split functions.Fixed bugs in javagen

commit 578ec127aec3040c73e590d660e0e65af4aa6934
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Thu May 5 03:04:42 2016 -0400

    Added getpages and split functions.Fixed bugs in javagen

commit 5820dd8d4544c6bfec70a4e23f21a463aa68c9fc
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu May 5 01:27:35 2016 -0400

    Added support for different fonts. PAL now supports 14 different fonts

commit 8f54c0edc8bac4c21a1d4852359f0e75a83a6b8a
Merge: 5ab7d91 65c2a45
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Thu May 5 01:26:09 2016 -0400

    Merge pull request #52 from vanvaridiksha/master

    Fixed a bug in import

commit 65c2a4524741f59444f2d22485e2a51bff3f36dd
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Thu May 5 01:25:39 2016 -0400

    Fixed a bug in import

commit 5ab7d91183e7d5d4c60be9b2258505221cb77f0f
```

```
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Thu May 5 00:15:28 2016 -0400

    Fixed tmap passing

commit 3d655d57c7dbdcb6d6612e10225fdc79e7d2a692
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu May 5 00:10:21 2016 -0400

    Made changes to the testcases.

commit cd01c7fb4d51f9f770c9e362ca4580fcdc30de5e
Merge: 882ce20 340f916
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Wed May 4 23:54:23 2016 -0400

    Merge pull request #50 from ANSSIN/master

    Fixed test cases

commit 340f9161ada7a44c75cfee4354d3a1f5dbf472ce
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Wed May 4 23:53:58 2016 -0400

    Fixed test cases

commit 882ce20f966ef9a1715ab5c4b2338c5f6bb674a1
Merge: 6d67185 364c46f
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Wed May 4 23:50:08 2016 -0400

    Merge pull request #49 from vanvaridiksha/master

    Added readtext in codegen. Added standard library

commit 364c46f0b7c46c34b26a04a273b40f4af32b1d6e
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
```

```
Date:   Wed May 4 23:49:35 2016 -0400

    Added readtext in codegen. Added standard library

commit 6d671856c0552110a834b4c77d332ad2ba7eab87
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed May 4 23:27:18 2016 -0400

    Fixed errors in automated text suite. Created new folders and made changes
        to diff.sh. Enabled logging of results of the test cases

commit 2af15052a77e06a7a86f51c718c1d6abc19a7528
Merge: 3bc4086 68ca198
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed May 4 22:00:49 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 68ca19850cff3c871ef2698c918e0942f0faf1ff
Merge: 04f8093 c178609
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Wed May 4 18:09:17 2016 -0400

    Merge pull request #48 from ANSSIN/master

    Wrapper for Test Suite to run all test cases. Pdftk not working on myâĂę

commit c1786090a2eb33e4c670d3e34f2605b2ccdeba82
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Wed May 4 18:08:01 2016 -0400

    Wrapper for Test Suite to run all test cases. Pdftk not working on my
        machine. Need to validate once.

commit 04f8093e7df4bc38a3cf44061c105212525acf37
Merge: 8049dc8 af1ac2d
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
```

```
Date:   Wed May 4 03:29:11 2016 -0400

    Merge pull request #47 from vanvaridiksha/master

    Added code for drawbarchart in codegen.ml. Fixed bugs in Java gen. EdâĂę

commit af1ac2d41c0c21ea22374b6467d90ae55d5dc735
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Wed May 4 02:58:46 2016 -0400

    Added code for drawbarchart in codegen.ml. Fixed bugs in Java gen. Edited
        makefile.

commit 3bc408677f1cac23c06c55d16a417f57ebfa639b
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed May 4 02:09:53 2016 -0400

    Added differences folder

commit 8049dc801fb40735318307b3dabf9518b9c6e76e
Merge: 795a0c7 c12f3ca
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Wed May 4 01:13:16 2016 -0400

    Merge pull request #46 from vanvaridiksha/master

    Fixed bugs in drawchart and readtable. Modified makefile.

commit c12f3ca56e6eff0fe03b9c2b4cad2085f616c924
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Wed May 4 01:12:34 2016 -0400

    Fixed bugs in drawchart and readtable. MOdified makefile.

commit 795a0c7c6734e70c36df3490ddc4f0c35ee55a3d
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed May 4 00:47:50 2016 -0400
```

```
    Added missing jar file to fix bug

commit c154feb08fa388c1bbb701b0da67bbffa759acd3
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed May 4 00:40:51 2016 -0400

    Added appropriate imports

commit a52d9c02769aa7cbcf407c38259560bf344df5d1
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed May 4 00:38:09 2016 -0400

    Added jar file

commit 0c2788896d2221100094d827f983e99a834f6551
Merge: 7fa9a24 8d5e293
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed May 4 00:37:10 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 8d5e293a898a731f6763038e435348f6e10edecb
Merge: e4d0d67 19f9c3c
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Wed May 4 00:22:47 2016 -0400

    Merge pull request #45 from vanvaridiksha/master

    Added imports in codegen. Added split and getpages functions. Pal tesâĂę

commit 19f9c3ca4c11adcfbe9ce234d8683fdb13e39a28
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Wed May 4 00:22:19 2016 -0400

    Added imports in codegen. Added split and getpages functions. Pal test code
        for testing pie charts and read table
```

```
commit e4d0d6714a52fcffbb27c8706eb99708d00e55e6
Merge: a4cbdfa 69aaf25
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Wed May 4 00:22:10 2016 -0400

    Merge pull request #44 from ANSSIN/master

    Adding return types for getpages and split

commit 69aaf251bc614b3cff9147e6d247d7c3663e6eab
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Wed May 4 00:21:43 2016 -0400

    Adding return types for getpages and split

commit 7fa9a24d217d7b22d22187b1cb425f2352cd5cd7
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed May 4 00:19:13 2016 -0400

    Changed type of chart type to png

commit a4cbdfa3f1a6debfb741886929be5f7433a2e3bf
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed May 4 00:18:06 2016 -0400

    Added drawbarchart and imports to Util

commit b3dbd5c54629ef08e9181f9ae9fa968b1d834668
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed May 4 00:04:08 2016 -0400

    Added drawpiechart method in codegen

commit 0b3b97877ca06a0cec380ed038773c7f9343291f
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue May 3 23:58:52 2016 -0400
```

```
    Added drawpiechart method in Util

commit 4f1b58ea72e00e2b41fd5940ef556d9d51263dbb
Merge: 50fef1f 7bebcb9
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Tue May 3 23:55:28 2016 -0400

    Merge pull request #43 from ANSSIN/master

    Adding corrected return type for readTable

commit 7bebcb96b529f681c155a432464f3b3c727a0248
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Tue May 3 23:36:20 2016 -0400

    Adding corrected return type for readTable

commit 50fef1f66c3e3669350d9e8a9e2473832cc5fcd6
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Tue May 3 23:25:17 2016 -0400

    Fixed Return Type

commit 121bbf9224acf3e5ff1eba0c80cfefc84adb6504
Merge: 4e43520 f2aeb60
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue May 3 22:48:54 2016 -0400

    Merge pull request #42 from vanvaridiksha/master

    Added readtable function in codegen and java implementation in Util.

commit f2aeb6057098c02d2eff4f5d8bebf99a2689fd92
Merge: 7a0361d 4e43520
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Tue May 3 22:47:10 2016 -0400
```

```
    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 7a0361dce3c4e3b3155912f10d4bec2a773589ef
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Tue May 3 22:47:06 2016 -0400

    Added readtable function in codegen and java implementation in Util.

commit 4e435204fdb0f62a7e6c939bc4face04ef1a06a8
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Tue May 3 22:37:51 2016 -0400

    Added TypeMap Merging

commit c705c302d42a806c681d6ff095f3b2ef199a91ae
Merge: 7eefd5c 254f142
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Tue May 3 22:02:54 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 7eefd5c38ff4d76379c46d4fcd1f664a3183ce4d
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Tue May 3 22:02:51 2016 -0400

    Resolving merge conflict

commit 254f1428655c1efbfd79c7c8c00cf02ce825cf8f
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Tue May 3 22:01:29 2016 -0400

    Correcting Warnings

commit 0d2267de12a83dc41f7e4ec2ea90387fea368d16
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue May 3 21:47:43 2016 -0400
```

```
    Added readtable, drawpiechart and readfile to predefined functions in
        analyzer

commit 49c7c83fc4befecf1d5443aadce3a36b79dc27ab
Merge: 8c1bdab a8c3cca
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sat Apr 30 20:15:21 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 8c1bdaba2e9ac945461d2eb621fb48af9c747f13
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sat Apr 30 20:15:07 2016 -0400

    Added details to the README file

commit a8c3cca32b57522798c53c9bf8aaf3b12dead85f
Merge: 77987dc c69730c
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Sat Apr 30 01:31:51 2016 -0400

    Merge pull request #41 from ANSSIN/master

    Fixed all warnings in codegen and parser conflicts. Who the man?

commit c69730c188d0b8aca97bf1152229863e9579a695
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Sat Apr 30 01:31:25 2016 -0400

    Fixed all warnings in codegen and parser conflicts. Who the man?

commit 77987dcab040cf9dc7286a235549f54a430cdaeb
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Fri Apr 29 19:43:49 2016 -0400

    Common List Elements Test
```

```
commit 4657dc5d548e159c9f381cfe6fb6c4cc4c2ef1c9
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Fri Apr 29 19:04:08 2016 -0400

    Fixed List Access Errors

commit 83902452d6259c94f2f346bc93ff627b7b13d402
Merge: 5cfdf8d cedf573
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:  Fri Apr 29 18:55:30 2016 -0400

    Merge pull request #40 from ANSSIN/master

    Fixed bug in List Assign and Negation Operator

commit cedf573a6baf01193bf19289bf4ebcd412dfae62
Author: Anshuman Singh <as4916@columbia.edu>
Date:  Fri Apr 29 18:54:53 2016 -0400

    Fixed bug in List Assign and Negation Operator

commit 5cfdf8d9ffcef70a9f013f75b5fbc0b31dfda8e8
Merge: 2a66427 bae6c4a
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:  Fri Apr 29 18:45:33 2016 -0400

    Merge pull request #39 from ANSSIN/master

    Added List Assign

commit bae6c4a15ea913e8debd7aa99c375197ec10aacc
Author: Anshuman Singh <as4916@columbia.edu>
Date:  Fri Apr 29 18:45:11 2016 -0400

    Added List Assign
```

```
commit 2a664276417e1f04f625e11cfee824cbd366a7a3
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Wed Apr 27 15:34:15 2016 -0400

    Comma delimited function formals

commit 04ac94c8d23ca9d2cb985b73475f3e0638a40ad7
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Wed Apr 27 03:42:30 2016 -0400

    Modified 3 column layout to have equidistant columns

commit f3bd42a885cbcf31c7cd42f6f33268ef1bac3840
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Wed Apr 27 02:34:44 2016 -0400

    Made changes to 3 column layout

commit aa0635e2854f540a64d19f4082d39c985865829d
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Wed Apr 27 02:08:06 2016 -0400

    Added logic for 2 columns and 3 columns layout

commit ac7e776450ddea0de96a7465799f4a3ba0179504
Merge: 03382ba 1c8f375
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:  Wed Apr 27 01:04:59 2016 -0400

    Merge pull request #38 from vanvaridiksha/master

    Fixed bugs in functionCallExpression and FunctionCallStatement

commit 1c8f375d43755c282ed240d9a86029c775e908e1
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:  Wed Apr 27 01:04:12 2016 -0400
```

```
    Fixed bugs in functionCallExpression and FunctionCallStatement

commit 03382bac2335442f5a71635bdd691b36f1500dc2
Merge: f1bbf56 0a964f5
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Wed Apr 27 00:05:53 2016 -0400

    Merge pull request #37 from vanvaridiksha/master

    Added function declaration to codegen

commit 0a964f5d097d055eb21ea34dfd12f0f8bf331a50
Merge: 0e47bdb f1bbf56
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Wed Apr 27 00:05:19 2016 -0400

    Added function declaration to codegen. Merged conflicts.

commit 0e47bdba495e881de83dfa748ca5781b5471d664
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Wed Apr 27 00:02:12 2016 -0400

    Added function declaration to codegen

commit f1bbf56cd3470893bdbe677f2dad5e5ce7aec52c
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Apr 26 23:12:37 2016 -0400

    Added multi line comments

commit b5d02fa1efffe104aeaa88a52b116d23920dfc67
Merge: 26a13ad 099af5d
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Apr 26 23:08:04 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL
    It is importantr
```

```
commit 26a13ad6cc9d329e07a4dd8871b4430e6e381a44
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Apr 26 23:06:10 2016 -0400

    Added default values for Line and Image decl

commit 099af5df284f891e6a8cd9839e0e0b2e312ebbae
Merge: 3ed12a5 46ef274
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Tue Apr 26 22:33:30 2016 -0400

    Merge pull request #36 from ANSSIN/master

    Added Map Access

commit 46ef27463f19f08d453c6e058931a08320a2aab9
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Tue Apr 26 22:32:39 2016 -0400

    Added Map Access

commit 3ed12a5ed3dc00e3e5b6d8d2d469dde9bbb7bb3c
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Tue Apr 26 22:15:09 2016 -0400

    Redefined Function Declaration

commit c7eb4d96ee901120bf63eff6d418e66c37f9d9ce
Merge: b181d87 28fd117
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Tue Apr 26 21:53:29 2016 -0400

    Merge pull request #35 from vanvaridiksha/master

    Added pal functions for writing a paragraph and writing text in grid âĂę
```

```
commit 28fd117aec7a0010edb4d50f8c9fdc75a127ac0c
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Tue Apr 26 21:52:47 2016 -0400

    Added pal functions for writing a paragraph and writing text in grid layout.

commit b181d873904d34ca2a1faa4e117be65a5a0ca8df
Merge: 9928865 e0a6613
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Tue Apr 26 21:47:32 2016 -0400

    Merge pull request #34 from ANSSIN/master

    Added map and list code gen. Peace.

commit e0a661374344e21e6a428444c802214be8c88ae3
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Tue Apr 26 21:46:29 2016 -0400

    Added map and list code gen. Peace.

commit 9928865f611ee0e4f598df94d45b33205b799e92
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Apr 26 20:35:03 2016 -0400

    Added code for three column layout

commit 5f22209507150c995f38d994b2ca1974f1a17f31
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Apr 26 19:30:35 2016 -0400

    Added code for write_column_layout

commit f2a563dd406b62adc7a30a91847e301451d6025d
Merge: 2f9567b 7aed90e
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Tue Apr 26 18:19:23 2016 -0400
```

```
    Merge pull request #33 from vanvaridiksha/master

    Added width attribute to line

commit 7aed90ecec9665b502e83d3fb44de4111c92946e
Merge: 7284312 cca7aa8
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:  Tue Apr 26 18:18:16 2016 -0400

    Added width attribute to line

commit 2f9567bb46a2067de378ded0b35fdb63a709cfa2
Merge: cca7aa8 7284312
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:  Mon Apr 25 00:13:30 2016 -0400

    Merge pull request #32 from vanvaridiksha/master

    Fixed a bug in codegen

commit 7284312d3afd25372c6169ca813d8639111aea54
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:  Mon Apr 25 00:11:11 2016 -0400

    Fixed a bug in codegen

commit cca7aa8aa4772d1f7ff5f803b632b9add1498f37
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Sun Apr 24 23:31:25 2016 -0400

    Added Support for Import Statements

commit eb02859dcf8ed913354f21f2b285a18a34a9d508
Merge: d74f4ea 4f5a851
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:  Sun Apr 24 23:26:48 2016 -0400
```

```
    Merge pull request #31 from vanvaridiksha/master

    Added pal code for paragraph and add image. Fixed bugs in analyzer

commit 4f5a851d463178dfd982ae89d5e3b12de153b682
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Sun Apr 24 23:26:04 2016 -0400

    Added pal code for paragraph and add image. Fixed bugs in analyzer

commit d74f4eae6fef13cee16fbbdce1786232cf66c036
Merge: fdbd929 f9f33b7
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Sun Apr 24 04:50:14 2016 -0400

    Merge pull request #30 from vanvaridiksha/master

    Fixed java bugs

commit f9f33b7edc10609a70c62190aba66fd8985f3667
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Sun Apr 24 04:49:34 2016 -0400

    Fixed java bugs

commit fdbd9292906d7b14223a111913ea20809f2de979
Merge: 45d5c74 7f65fdf
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 24 04:31:31 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 45d5c74ddd4d052217c4be94ee5fe43ab9168210
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 24 04:31:19 2016 -0400
```

```
    Added Image to Concat Binop

commit 7f65fdf4938606d5de6e1a765c20f2b0bc0c733a
Merge: 20b5ff6 e6bf964
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Sun Apr 24 04:24:22 2016 -0400

    Merge pull request #29 from vanvaridiksha/master

    Added an Image class to javagen. Added codegen for image declaration and
        adding an image to a pdf.

commit e6bf9646ae052db27446c3d55b4e2301987f3293
Merge: 4877f89 20b5ff6
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Sun Apr 24 04:22:40 2016 -0400

    Created an Image class in javagen. Added codegen logic for image loading
        and adding image to a pdf

commit 4877f89b22c84074b2ea656232906f9e65481513
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Sun Apr 24 04:17:35 2016 -0400

    Created an Image class in javagen. Added codegen logic for image loading
        and adding image to a pdf

commit 20b5ff6a32727421128d8f569b9a5fe4d903fd2a
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 24 03:50:36 2016 -0400

    Added logic to append string type to TUop when the uop is linebuffer

commit da941d4ea4774b76cc793f5dc27644e84607f43f
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 24 03:33:58 2016 -0400
```

```
    Added linebuffer logic in codegen

commit d36e1bcf7d0b855b850f442050d3b0edfc1cd049
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 24 02:56:01 2016 -0400

    Added linebuffer logic in codegen

commit 3b737d5159890da9850f2d6bf963b858d65cedd1
Merge: 74cb143 b615abb
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Sun Apr 24 02:55:34 2016 -0400

    Merge pull request #28 from vanvaridiksha/master

    Added text wrapping logic to Util and Line. Fixed bugs in code gen

commit b615abbf3125aae6e843a45b23ff5d5b5514964e
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Sun Apr 24 02:55:13 2016 -0400

    Added text wrapping logic to Util and Line. Fixed bugs in code gen

commit 74cb143bee8e738a18fdd7045aeb70a5d3222cb8
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 24 02:18:26 2016 -0400

    Added linebuffer logic in parser and lexer

commit 863e4b4351f952e87e76975bef403647fd61e557
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 24 00:39:20 2016 -0400

    Commented code for List

commit a084a1db3297263a6ea66625e5c705e559a54fe9
Merge: 0cb8f41 07ade4f
```

```
Author: Diksha Vanvari <vanvari.diksha@gmail.com>
Date:   Fri Apr 22 03:24:42 2016 -0400

    Merge pull request #27 from vanvaridiksha/master

    Added length, readfile functions; continue and break control statemenâĂę

commit 07ade4f137e1ef4c048995ea368d8ae7470a388e
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Fri Apr 22 03:19:58 2016 -0400

    Added length, readfile functions; continue and break control statements;
        Java code for wrap text

commit 0cb8f411934e3548fa40f1f33940650271038441
Merge: cb88039 228dc15
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Fri Apr 22 03:16:17 2016 -0400

    Merge pull request #26 from ANSSIN/master

    Added continue, break, function declarations, function expressions anâĂę

commit 228dc1525ab0a4ce79beaa07f31d6a641cd8b7d2
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Fri Apr 22 03:15:45 2016 -0400

    Added continue, break, function declarations, function expressions and
        removed warnings from Analyzer

commit cb880391bb593470a3f57f33ace7b3881d7d511f
Merge: b8f75bf fd51496
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Fri Apr 22 00:14:43 2016 -0400

    Merge pull request #25 from ANSSIN/master
```

```
    Added image

commit fd51496bbb2736f595269a9a6e37fb7fb892272c
Author: Anshuman Singh <as4916@columbia.edu>
Date:  Fri Apr 22 00:13:31 2016 -0400

    Added image

commit b8f75bf8b7c9051dd8dba0b0229e197ba8c1f0d9
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Thu Apr 21 21:22:56 2016 -0400

    Finished List Remove

commit 399ea3d125ce933a0ec9cb278e9c4e9e3427ead8
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Thu Apr 21 12:52:15 2016 -0400

    Finished Map Remove

commit 5af41d7906b47a29c7d02ec0ab4fd447463331ca
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Thu Apr 21 01:00:24 2016 -0400

    Finished List and Map Add

commit 343cd64f4200520b065dd44f86709911c0f27216
Merge: d72d270 32a73d7
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:  Wed Apr 20 23:53:00 2016 -0400

    Merge pull request #24 from ANSSIN/master

    Added Map Access and resolved merge issues

commit 32a73d7f9ebc7a15d58e04fa3778b2482b8f9886
Author: Anshuman Singh <as4916@columbia.edu>
```

```
Date:  Wed Apr 20 23:52:27 2016 -0400

    Resolved merge issues with parser

commit d72d2708bc3361a4d200be12c3e1227860df545d
Merge: f486e7d 65863e2
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:  Wed Apr 20 23:50:23 2016 -0400

    Merge pull request #23 from ANSSIN/master

    Added Map Access

commit 65863e2b825a7ddf8eb736460c607e63f6b1c579
Merge: 124bfa0 f486e7d
Author: Anshuman Singh <as4916@columbia.edu>
Date:  Wed Apr 20 23:47:45 2016 -0400

    Resolved merge issues

commit 124bfa01de857e884cb39a2754c565002ba1ce9d
Author: Anshuman Singh <as4916@columbia.edu>
Date:  Wed Apr 20 23:34:31 2016 -0400

    Refactoring Code

commit f486e7d103a27125f85261fb5a240fe9d2646f11
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Wed Apr 20 21:43:04 2016 -0400

    Finished List Access RHS

commit 3d49c38a8f69b1dd19801afcc0c43a8889f2abe1
Merge: c89bbce 73f8ea8
Author: Anshuman Singh <as4916@columbia.edu>
Date:  Wed Apr 20 21:17:26 2016 -0400
```

```
    Refactoring Code

commit 73f8ea8f37d4317d0539fa41ef81c6898d9075e2
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Wed Apr 20 20:15:17 2016 -0400

    Finished List Access LHS

commit c89bbcef74c8208cf1bde1146710868a860fa58c
Merge: 964139a 09a8d4f
Author: Anshuman Singh <as4916@columbia.edu>
Date:  Wed Apr 20 17:35:17 2016 -0400

    Dynamic Typing with String Map

commit 09a8d4ff6fe3587d020c38df5a231796eb95f87c
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Wed Apr 20 17:00:44 2016 -0400

    Interim List Change

commit 964139a8f4f0ac3acadffd480fed8822cc736922
Author: Anshuman Singh <as4916@columbia.edu>
Date:  Wed Apr 20 01:46:04 2016 -0400

    Dynamic Typing by adding it to environment

commit b244fe2a3e93b679600bb8c688802c44463cd869
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Wed Apr 20 01:13:26 2016 -0400

    Dynamic Typing

commit b400d94c52f4c9dc2acc77e798739accb2c2e4c3
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Sun Apr 17 21:04:36 2016 -0400
```

```
    Initial Commit for List

commit f6def3819b03d2790170e319ecbfb877fe1a3688
Merge: d7c2f22 caaf67f
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 17 03:33:12 2016 -0400

    Merge pull request #21 from vanvaridiksha/master

    Added codegen for if-elif-else

commit caaf67f9659ca192abec594ca81983d312f0542e
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Sun Apr 17 03:32:04 2016 -0400

    Added codegen for if-elif-else

commit d7c2f22d7b7bf7fb08176d15a616aa6aad9cf368
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sat Apr 16 23:49:19 2016 -0400

    Made changed to diff.sh to check whether a pdf matches the golden copy

commit 88df2b12796f6b5ba4ae57a92b32d1ceb055a015
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Apr 13 12:31:59 2016 -0400

    Added diff command to the Makefile to find the difference between two pdf's

commit 83959ecb30b95d55903d5f521ff171308a2f783c
Merge: ea68556 35762f4
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Apr 13 11:50:40 2016 -0400

    Merge pull request #20 from ANSSIN/master

    Test Cases for Test Suite
```

```
commit ea685567cd088066c627c05402743f1ba0a4b871
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Apr 13 11:49:54 2016 -0400

    Added automated comparison of pdf files and modified Makefile to compile
        java generated code

commit 35762f4c6a5feead5c68940ffb730d200e8e0279
Author: Anshuman Singh <as4916@columbia.edu>
Date:   Wed Apr 13 10:27:37 2016 -0400

    Test Cases for Test Suite

commit e61d4fbfb42f383212693a5bd4fc30aba04e09d3
Merge: 3ea56bd 7092e30
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Apr 13 01:30:55 2016 -0400

    Merge pull request #19 from vanvaridiksha/master

    Test Cases

commit 7092e301fe20ca2e9e21dff76f08bb1b2b8979a2
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Wed Apr 13 01:19:09 2016 -0400

    Added test cases

commit 070e9933dd78ed713083ef3910c17a4f5e2a67b2
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Wed Apr 13 01:17:38 2016 -0400

    Added test cases

commit 3ea56bd80dd2c05859de81cfa886ef372cb5e053
Author: Vinay Gaba <vinaygaba@gmail.com>
```

```
Date:   Tue Apr 12 19:48:31 2016 -0400

    Added test pdf's

commit c807440c8e6403e24276867905a5ef1ee9263417
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Apr 12 18:45:37 2016 -0400

    Commented If Loops to be able to run the code

commit 679666d2cd58f2edc9e661f0e685243b637f9990
Merge: fbbb966 efda9ab
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Apr 12 17:06:40 2016 -0400

    Merge pull request #18 from vanvaridiksha/master

    Added codegen logic for for, while, if-else

commit efda9abc56315281834db4c0497027118faefc5e
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Tue Apr 12 17:00:15 2016 -0400

    Added codegen logic for for, while, if-else

commit fbbb96657235a7b3713abc5628afa55954dbc73b
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Fri Apr 8 21:37:26 2016 -0400

    Added InitAssign and Nested If ELse in Analyzer & Cleaned Sast and Ast

commit 33b19f8b185ca25dfaf03604ad1cad678c8802e8
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Fri Apr 8 18:48:14 2016 -0400

    Added While Statement in Analyzer
```

```
commit 33e79c79f3e9c680d0e82c87ab20334301b18724
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Fri Apr 8 18:18:38 2016 -0400

    Added For Statement in Analyzer

commit ca0e788bc44b061a4e6a944e17adfa336709cb30
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Fri Apr 8 17:33:04 2016 -0400

    Correct Binop Expression Types

commit 5f8badd99481fc4bcaf187900f98d6f88035607d
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Fri Apr 8 17:21:23 2016 -0400

    Made changes to gitignore

commit beff861dba5d113b44ecef35bdf5391057c7d7ff
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Fri Apr 8 17:05:12 2016 -0400

    Refactored folders

commit 2f4bee8706f73a648165c8b377e258df215b2fae
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Fri Apr 8 17:03:32 2016 -0400

    Added gitignore

commit e3989647f7a5b622e8be41cf72e44c8a6a3d7ebf
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Fri Apr 8 15:39:05 2016 -0400

    Added all other binop operations

commit 7399665731bdcfe711ff0823532463d6777e34f9
```

```
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Fri Apr 8 15:06:46 2016 -0400

    Added other binop operations: Add, Div, Mul, Sub

commit 8c538ebae105dce5d2e64a93c337d527117590b3
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Tue Apr 5 15:47:40 2016 -0400

    Added Java POJOs and helper classes in javagen

commit 559cc45e44044e6ec1950479d63df0706f8ba750
Merge: 67d6cf9 4f1539c
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Tue Apr 5 15:39:46 2016 -0400

    Merge pull request #17 from vanvaridiksha/master

    Fixed assignment for tuple and var

commit 4f1539cb35f6194da932837ebd7a52a55934b701
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:  Tue Apr 5 15:38:13 2016 -0400

    Fixed assignment for tuple and var

commit 67d6cf9347cd04939829d20472ff4a6999efe36f
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Tue Apr 5 15:30:52 2016 -0400

    Added addLineToTuple method in Util.java

commit 392fc47c79c09d904b132dbdd424091b6a7282ee
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Tue Apr 5 15:10:27 2016 -0400

    Added Util.java
```

```
commit 2e6666dff9f4639072f3ab469748ff47b0baaf7b
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Tue Apr 5 13:21:47 2016 -0400

    Fixed expression map error in codege. Hello World works successfully

commit 748a0580fdf0362804ee3ed95c9ad93f803dbf81
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Mon Apr 4 20:37:12 2016 -0400

    Fixed codegen. Hello World compiles successfully

commit 00699461fb76fd3db49a4ab9dee38a06059f8a3f
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Sun Apr 3 23:37:21 2016 -0400

    Completed codegen to correctly generate code

commit c1a928f940d30928af7337a9fb0718c51f3ac529
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Sun Apr 3 21:27:19 2016 -0400

    Made changes to codegen to generate primitive pdfbox classes. Changes in
        Analyzer as well

commit 5ead7703d4545fe6ce70b2cd14a925a2272d80bb
Merge: 29b3b56 4b1f430
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Sun Apr 3 20:03:50 2016 -0400

    Merge pull request #16 from vanvaridiksha/master

    Fixed a big in the parser

commit 4b1f430443185576d27823d4bec220b1a092360f
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
```

```
Date:   Sun Apr 3 20:03:12 2016 -0400

    Fixed a big in the parser

commit 29b3b569da95449cf62ec0762693b1d856f9c0b3
Merge: 23d21e2 06a9330
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 3 19:57:11 2016 -0400

    Merge pull request #15 from vanvaridiksha/master

    Updated makefile

commit 06a93300b8b6297146377564fe0ff9610b0fc078
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Sun Apr 3 19:56:42 2016 -0400

    Updated makefile

commit 23d21e28251cf7874d977b05ec4e4ddb63e0c08e
Merge: 430f3c5 dc64aaa
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 3 19:02:21 2016 -0400

    Merge pull request #14 from vanvaridiksha/master

    Completed codegen. Updated sast ast and analyzer

commit dc64aaae96d23b6918e7127394b764d4e1bdfb86
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Sun Apr 3 19:00:26 2016 -0400

    Completed codegen. Updated sast ast and analyzer

commit 430f3c579cc5532ddabeb1baba1c954201397b52
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Sun Apr 3 07:44:17 2016 -0400
```

```
    Makefile and pal changes

commit 522020f81371b289cb6dd9e20e1d2bd96b2e00e0
Merge: 1e0c46f 9ebea5d
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 3 07:32:18 2016 -0400

    Merge pull request #13 from vanvaridiksha/master

    Added code to run analyzer and codegen

commit 9ebea5d998624e56c72557022f2754614eeaf51b
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Sun Apr 3 07:31:23 2016 -0400

    Added code to run analyzer and codegen

commit 1e0c46f1c018ac004c4a9e0c0afa52652011ac35
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Sun Apr 3 07:24:30 2016 -0400

    Maybe Completed Analyzer

commit c627735c9edb1c695bc5be50855f79c4668f2cda
Merge: 9c81815 4fb1da4
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 3 06:24:23 2016 -0400

    Merge pull request #12 from ANSSIN/master

    Modifications to lexer and parser for Tuple

commit 4fb1da462fa1f690824f467e17f2bca2d99d583d
Author: Anshuman Singh <anshuman@Anshumans-MacBook-Pro.local>
Date:   Sun Apr 3 06:21:41 2016 -0400
```

```
    Modifications to lexer and parser for new data types

commit f048a516fd60acaf2e0b0c4f51f63eefb98f2273
Author: Anshuman Singh <anshuman@Anshumans-MacBook-Pro.local>
Date:   Sun Apr 3 04:21:02 2016 -0400

    Added tuple data type

commit 9c818154d6b8e07334bdc0831165e04931362fb8
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 3 03:36:01 2016 -0400

    Added all files that are modified: Line.java, Tuple.java, codegen, parser.ml

commit 1fd88748f260a0459a977855cc102cd0f26cbb39
Merge: 51da755 708400b
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 3 03:23:13 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 51da7555fe4bb580c47031c0c639c555790d6ddb
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 3 03:21:40 2016 -0400

    Made changes PrimitiveObject

commit 45a4d62e173c3529367e6bd863a548b89bf60120
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Apr 3 03:19:49 2016 -0400

    Made changes to hello world

commit 708400b18d45234b719fe12379622ca7a7eb519f
Author: viralshahrf <viralshahrf@gmail.com>
Date:   Sun Apr 3 03:18:13 2016 -0400
```

```
    Updated Analyzer Sast Ast Makefile

commit 79359b88e404e7664d1d02dbd975370617c7bc0a
Merge: 0e6d6a3 2f2aaa1
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Sun Apr 3 01:52:04 2016 -0400

    Resolved merge conflicts in analyzer amd sast. Completed codegen

commit 0e6d6a338c25d137c08bbadd730f99c01f64b62b
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Sat Apr 2 23:33:36 2016 -0400

    Made changes to analyser,sast and ast

commit 2f2aaa159e845dc51395f6d9b1a1705336451e48
Author: viralshahrf <viralshahrf@gmail.com>
Date:  Fri Apr 1 15:05:12 2016 -0400

    Adding the Analyzer and Related Changes in Sast and Ast

commit 5f0fd9ba6d6fc6c9708bafc5e859960879901513
Merge: 2890840 53d0f70
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Thu Mar 31 22:21:08 2016 -0400

    Merge pull request #11 from ANSSIN/master

    First Draft of Analyzer

commit 53d0f706f910dda2fb98bb19451adf7021404348
Author: Anshuman Singh <anshuman@Anshumans-MacBook-Pro.local>
Date:  Thu Mar 31 22:20:17 2016 -0400

    First draft of analyzer

commit 2250611910f217d714957dc7a5f87c6481916864
```

```
Merge: 757b65e 2890840
Author: Anshuman Singh <anshuman@Anshumans-MacBook-Pro.local>
Date:   Thu Mar 31 22:18:54 2016 -0400

    Getting fork upto speed

commit 289084094e0f4e167744e9ce314e4f68f4a5dc69
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Mar 29 22:27:09 2016 -0400

    Made changed to codegen

commit fb4eda3df5f2b433c8a758e0fcd52d353936aec8
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Mar 29 20:16:30 2016 -0400

    Added Primitive Object class

commit 2b19eb092e99ed9a4a46c83fb4675706546d8c5a
Merge: 5d64153 015a215
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Mar 29 19:45:53 2016 -0400

    Merge pull request #10 from vanvaridiksha/master

    Started work on codegen. Completed Sast.

commit 015a21500bc15d2830e6a693c435775ca8da8947
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Tue Mar 29 19:44:41 2016 -0400

    Started work on codegen. Completed Sast.

commit 5d64153fad6650bbbba148908a871aa2aead7207
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Mar 23 20:37:26 2016 -0400
```

```
    Added HelloWorld for pal and java

commit 5053abf2082be15663663267db5384c93c0b01fb
Merge: 25b8cd5 c65d5ea
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Mar 23 20:01:01 2016 -0400

    Merge pull request #9 from vanvaridiksha/master

    Added list data type

commit c65d5ea1f13f485cc8effb613a9563416a54ac3d
Author: Diksha Vanvari <dikshavanvari@dyn-160-39-198-61.dyn.columbia.edu>
Date:   Wed Mar 23 19:58:20 2016 -0400

    Added list data type

commit 25b8cd58e962c41a4febc062d6bcfa4331c13265
Merge: 59bc35b 6335a63
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Mar 23 18:40:24 2016 -0400

    Merge pull request #8 from vanvaridiksha/master

    Fixed bugs in ast, parser and Makefile

commit 6335a63a88940bffe6584ec7fba22aba983d2f59
Author: Diksha Vanvari <dikshavanvari@dyn-160-39-198-61.dyn.columbia.edu>
Date:   Wed Mar 23 18:39:22 2016 -0400

    Fixed bugs in ast, parser and Makefile

commit 59bc35b26a0fd89ed5de93386d69e6e1a65b1f65
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Wed Mar 23 18:22:58 2016 -0400

    Update ast.mli
```

```
commit 5a519a4dc324abe3ec19205342d085555219a690
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:  Wed Mar 23 18:21:58 2016 -0400

    Update parser.mly

    Fixed function declaration lists

commit 9836c79bfff43cc2f1a321aae97a534481d389aa
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Mon Mar 21 15:08:26 2016 -0400

    Fixed Makefile typo

commit fe4433884ea166158d719a364d5b356e35617aed
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Mon Mar 21 15:06:32 2016 -0400

    Fixed Merge Conflicts

commit 22b0e221ac709fd056bf15ec79d844c2a98abf00
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Mon Mar 21 15:04:34 2016 -0400

    Fixing merge conflicts

commit 3d63ae5ed213ece9282e34f7cfc517fe394e96f2
Merge: 41fcd04 806fbce
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Mon Mar 21 15:02:00 2016 -0400

    Creating stash for ast and parser

commit 41fcd041122e7dda9af2f5504c4b56c744d4d5eb
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Mon Mar 21 14:59:39 2016 -0400
```

```
    Creating stash for ast and parser

commit c6aadd08a12de2937b71aee9fb8fa74a83e81a4d
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Mon Mar 21 14:57:55 2016 -0400

    Added Makefile

commit 806fbcecf3c9ba682ff9503796ee8b86bc362525
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Wed Mar 16 22:15:52 2016 -0400

    Update parser.mly

    Ironing out Issues

commit 037ce32466d952ff65f93c2fb82b4e9295a9d8e9
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Wed Mar 16 22:15:28 2016 -0400

    Update ast.mli

    Ironing out issues

commit 757b65eac1191cacddef6fafc6724a01846ec65e
Author: Anshuman Singh <anshuman@Anshumans-MacBook-Pro.local>
Date:   Wed Mar 16 22:09:42 2016 -0400

    Ironing out issues

commit fcff4d6ad9bca8219c6cb99521d54abfd0f38ffc
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Wed Mar 16 21:23:56 2016 -0400

    Update ast.mli
```

```
    Added multiple functions as start of program

commit a989a52ea933f835cbe9be2ffc99e6694a9dc6dc
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Wed Mar 16 21:12:43 2016 -0400

    Update ast.mli

    Adding import statements and changing start of program

commit 578acf20f755971cac2589498760635f0afc05a8
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Mar 16 21:09:23 2016 -0400

    Added logic to parse function calls and if

commit 3735fae723b90f61f1ac413dcccf7200aab19085
Merge: 0ab1fa0 e3075e8
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Mar 16 19:07:28 2016 -0400

    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 0ab1fa0736bcb603056e16e0f32ba4a1bd5a31e7
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Mar 16 19:07:15 2016 -0400

    Made changes in the ast.mli

commit e3075e8708067b108ad29445336c9488fc07e2e7
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Wed Mar 16 19:03:34 2016 -0400

    Update parser.mly

commit 7d04a9c567e33adff63c0690ecd2253b686c3d80
Author: ANSSIN <singh-anshuman@hotmail.com>
```

```
Date:   Wed Mar 16 18:44:16 2016 -0400

    Update ast.mli

commit 41e4d8c6df50d858cc66deec83c3f507cb8b5512
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Wed Mar 16 17:21:35 2016 -0400

    Committing the latest version

commit 50baf682938792aa78c35dd3eb617db5877ac51c
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Mar 15 19:23:23 2016 -0400

    Made changes in the ast.mli based on the LRM

commit 680d57c582f302e129b24b0320322d0c27d0d957
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Tue Mar 15 16:06:08 2016 -0400

    Added operators in expr

commit efbd102e77c863ba32e0a1d9b6dc07af6df45068
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Mon Mar 14 02:51:56 2016 -0400

    Changed the scanner and parser according to the LRM

commit 8456fae7c2924f06f059e719c2df97b83875659b
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu Mar 10 11:25:58 2016 -0500

    Added LRM to docs folder

commit b1a7c45d6acc042f95f3a30206b2624687702ecc
Merge: c139de0 62696e5
Author: Vinay Gaba <vinaygaba@gmail.com>
```

```
Date:   Thu Mar 10 11:22:01 2016 -0500

    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit c139de0dd6c275be7d05a82addfc40e6cb8ee1e2
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu Mar 10 11:21:52 2016 -0500

    Added java method to write across multiple lines

commit aa7425df35278aaaed78c53e8a03b313ebd7b845
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu Mar 10 11:20:34 2016 -0500

    Added tokens based on the LRM

commit 62696e5da81666314cda39bf7a37fe805935f768
Merge: ad60ef5 70c7f2d
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Sun Feb 21 17:57:31 2016 -0500

    Merge pull request #6 from ANSSIN/master

    Split PDFs

commit 70c7f2d24eaf6ed28691da2d3c8c66c349568312
Author: Anshuman Singh <anshuman@Anshumans-MacBook-Pro.local>
Date:   Sun Feb 21 17:55:26 2016 -0500

    Split a pdf into documents and saving documents as different pdfswith
        ability to set split size and input/output file names

commit a8f708eb532eec357b8769375897852086457bcf
Merge: 29e6be2 ad60ef5
Author: Anshuman Singh <anshuman@Anshumans-MacBook-Pro.local>
Date:   Sun Feb 21 17:49:20 2016 -0500
```

```
    Merge branch 'master' of https://github.com/vinaygaba/PAL

commit 29e6be22f6d98851d70487f5ad52b71e3d07e7ac
Author: Anshuman Singh <anshuman@Anshumans-MacBook-Pro.local>
Date:   Sun Feb 21 17:47:17 2016 -0500

    Split a pdf into documents and saving documents as different pdfswith
        ability to set split size and input/output file names

commit ad60ef5132fe1578a3476089617b5f0b221a4d47
Merge: 119399d f3f55ad
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sun Feb 21 17:45:29 2016 -0500

    Merge pull request #5 from vanvaridiksha/master

    Added a util function for drawing a table in a pdf and adding data to it

commit f3f55ad4c80ce2a72a5bdc949e3658ef28b7b1d5
Author: Diksha Vanvari <dikshavanvari@Dikshas-MacBook-Pro.local>
Date:   Sun Feb 21 17:43:41 2016 -0500

    Added a util function for darwing a table in a pdf and adding data to it

commit 119399d8be93bdb4022852137af341fa70f8d50d
Merge: 167221a fea1ecb
Author: ANSSIN <singh-anshuman@hotmail.com>
Date:   Sun Feb 21 17:41:12 2016 -0500

    Merge pull request #4 from ANSSIN/master

    Split PDFs

commit fea1ecb73d6b7f843f9beaa4d4f1b2f1150c4d4e
Author: Anshuman Singh <anshuman@Anshumans-MacBook-Pro.local>
Date:   Sun Feb 21 17:39:47 2016 -0500
```

```
    Split a pdf into documents and saving documents as different pdfswith
        ability to set split size

commit 167221a0b0ed950f9463ecf6af3371361c285040
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Sun Feb 21 17:36:12 2016 -0500

    Added method to merge two pdfs

commit 1daef8d1bd8ecfe2a9f5e1fac80f4c44b686416d
Author: Anshuman Singh <anshuman@Anshumans-MacBook-Pro.local>
Date:  Sun Feb 21 17:32:10 2016 -0500

    Split a pdf into documents and saving documents as different pdfs

commit 80f921742c0b52b7362bf021e8f4706a205b640c
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Sun Feb 21 16:44:44 2016 -0500

    Added PDFBox .jar

commit 5745e40240a1263c36f0b24f13f2ae0637f2d15e
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Sun Feb 21 16:37:51 2016 -0500

    Added Java Project

commit 87add096f57f64e2a26b174323905a332402957b
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Sat Feb 20 18:50:22 2016 -0500

     Concat Working

commit 02813d9f4e7bd24e62f91d638ae85d59d81abeca
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Sat Feb 20 17:34:02 2016 -0500
```

```
    Added Ast.mli

commit e76333aae74b998efef8543d33a86bea2d42594e
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sat Feb 20 15:28:08 2016 -0500

    Added boolean operators to the lexer

commit 839c043ea6dbda2bbc58d106e8a25299942254ae
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sat Feb 20 15:18:03 2016 -0500

    Added boolean operators to the lexer

commit 2d91cf611dee0eb483f3cebc79a86f267c8a7ed8
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Sat Feb 20 15:01:01 2016 -0500

    Added more tokens to our lexer

commit 536e1f6c1d56b1fb9ab79de76a5b4c5460b1e509
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu Feb 11 02:01:15 2016 -0500

    Added more rules in the lexure

commit d6fc2097087adf3444bae25985f667c7d6786098
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu Feb 11 00:47:21 2016 -0500

    Generated a test lexer.

commit 770d0a8c3f4f8195ca3ba0e92ea0a734671bac36
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:   Thu Feb 11 00:29:37 2016 -0500

    Added Proposal
```

```
commit d1eda5cc795d486a9469e6775381d311fb9f356f
Author: Vinay Gaba <vinaygaba@gmail.com>
Date:  Thu Feb 11 00:24:10 2016 -0500

    Initial Commit
```

5. Test Plan

Our testing suite implements test programs for each individual aspect of PAL. There are individual tests for:

- Operators
- For Loops
- If-else statements
- While Loops
- Primitive data types
- List data type
- Map data type
- Predefined Constructs
- Function Declaration
- Function Calling
- Import Statements
- PDF Functionalities
- List Type Inference
- Concatenation Operations
- Scoping

6. Lessons Learned

Anshuman Singh

1. Thinking from a functional programming perspective really accelerated the development of the compiler.
2. Get a working test suite ready as soon as possible. Saves hours in fixing bugs introduced because of regression.
3. Really enjoyed working with OCaml once you are over the learning curve. Just keep at it and you will be surprised with OCaml.

Diksha Vanvari

1. Start early. Some knowledge of OCaml before taking the course would have helped.
2. Do not wait until the Hello World deadline to get Hello World to work.
3. Keep an open mind towards the shift from Object Oriented programming to Functional programming.

Vinay Gaba

1. Start loving OCaml earlier than later
2. Thinking in terms of Object Oriented Programming for this project would be the worst thing you could do to your project.
3. Be prepared to spend a lot of nights churning out OCaml code.

Viral Shah

1. We are too ingrained in Object Oriented Programming. Functional Programming provides new perspective and refreshes you as a programmer.

2. On days when I just didn't want to work on this project, I would sit down, start writing in OCaml, and then keep writing. Never has a programming language seemed so interesting and productive. Often, I found myself wondering how the OCaml compiler worked.

3. Start very early.Learn OCaml even earlier.

Appendix

MAKEFILE

```
.PHONY : make
make :
   ocamlc -c ast.mli
   ocamlyacc -v parser.mly
   ocamlc -c parser.mli
   ocamlc -c parser.ml
   ocamllex lexer.mll
   ocamlc -c lexer.ml
   ocamlc -c sast.ml
   ocamlc -c analyzer.ml
   ocamlc -c codegen.ml
   ocamlc -c pal.ml
   ocamlc -o pal sast.cmo parser.cmo lexer.cmo analyzer.cmo codegen.cmo pal.cmo


.PHONY : clean
clean :
   rm -f pal parser.ml parser.mli lexer.ml bin/*.class \
   *.cmo *.cmi *.output *.class *.java *.log *.csv

.PHONY : compile
compile :

   javac -cp
       "javagen/.:javagen/pdfbox.jar:javagen/jfreechart-1.0.19.jar:javagen/jcommon-1.0.23.jar:
       javagen/Output.java

   java -cp
       "javagen/.:javagen/pdfbox.jar:javagen/jfreechart-1.0.19.jar:javagen/jcommon-1.0.23.jar:
```

```
      Output

.PHONY : diff
diff :
	touch ../test/output.pdf
	mv helloworld.pdf ../test/output.pdf
	cd ../test/;./diff.sh

.PHONY : runall
runall :
	cd ../test/;./runalltests.sh
```

OCaml Code

Scanner

```
{

open Parser

}

(*test*)

let digit = ['0'-'9']
let id = ['a'-'z'] ['a'-'z' 'A'-'Z' '0'-'9' '_']* ['?']?
let ws = [' ' '\r' '\t' '\n']


rule token = parse
    | ws                        {token lexbuf}
    | ','                       { COMMA }
    | ';'                       { SEMICOLON }
    | ':'                       { TYPEASSIGNMENT }
    | eof                       { EOF }
    (* Scoping *)
```

```
| '{'                   { LEFTBRACE }
| '}'                   { RIGHTBRACE }
| '('                   { LEFTPAREN }
| ')'                   { RIGHTPAREN }
| '['                   { LEFTBRAC }
| ']'                   { RIGHTBRAC }
(* Operators *)
| '*'                   { MULOP }
| '/'                   { DIVOP }
| '+'                   { ADDOP }
| '-'                   { SUBOP }
| '%'                   { MODOP }
| "<>"                  { SWAP }
| "<="                  { LEQ }
| ">="                  { GEQ }
| '<'                   { LT }
| '>'                   { GT }
| "=="                  { EQ }
| "!="                  { NEQ }
| "&&"                  { AND }
| "||"                  { OR }
| "!"                   { NOT }
| '='                   { ASSIGN }
| '.'                   { CONCAT }
| "|_"                  { LINEBUFFER }
(* Keywords *)
| "bool"                { BOOLD }
| "true"                { BOOL(true) }
| "false"               { BOOL(false) }
| "int"                 { INTD }
| "float"               { FLOATD }
| "string"              { STRINGD }
| "pdf"                 { PDFD }
| "page"                { PAGED }
| "line"                { LINED }
| "list"                { LISTD }
| "map"                 { MAPD }
```

```
    | "image"                     { IMAGED }
    | "tuple"                     { TUPLED }
    | "if"                        { IF }
    | "elif"                      { ELIF }
    | "else"                      { ELSE }
    | "while"                     { WHILELOOP }
    | "for"                       { FORLOOP }
    | "import"                    { IMPORT }
    | "void"                      { VOID }
    | "null"                      { NULL }
    | "main"                      { MAIN }
    | "return"                    { RETURN }
    | "continue"                  { CONTINUE }
    | "break"                     { BREAK }
    | "function"                  { FUNCTION }
    (* Literals *)
    | digit+ as int               { INT(int_of_string int) }
    | digit+'.'digit+ as float    { FLOAT(float_of_string float) }
    | '"'('\\'_|[^'"'])*'"' as str { STRING(str) }
    (* Identifier *)
    | id as i                         { ID(i) }
    (* Comment *)
    | '#'                         {comment lexbuf}
    | "/*"              { multilinecomment lexbuf }
    | _ as char { raise (Failure("Illegal character " ^ Char.escaped char)) }

and comment = parse
    | '\n'                        {token lexbuf}
    | _                           {comment lexbuf}

and multilinecomment = parse
      "*/" { token lexbuf }
    | _    { multilinecomment lexbuf }
```

Parser

```
%{ open Ast %}
%token SEMICOLON
%token LEFTBRACE LEFTPAREN LEFTBRAC RIGHTBRACE RIGHTPAREN RIGHTBRAC COMMA
%token ADDOP SUBOP MULOP DIVOP MODOP
%token SWAP CONCAT TYPEASSIGNMENT LINEBUFFER
%token EQ NEQ LT GT LEQ GEQ
%token NOT AND OR
%token ASSIGN
%token IF ELIF ELSE WHILELOOP FORLOOP BREAK CONTINUE VOID NULL
%token EOF
%token IMPORT FUNCTION RETURN MAIN
%token CONTINUE
%token BREAK
%token <string> ID
%token IDTEST
%token <string> STRING
%token <int> INT
%token <float> FLOAT
%token <bool> BOOL
%token INTD BOOLD STRINGD FLOATD PDFD PAGED LINED LISTD TUPLED IMAGED MAPD
%left ASSIGN
%left OR
%left AND
%left EQ NEQ
%nonassoc LT LEQ GT GEQ
%left ADDOP SUBOP
%left CONCAT
%left LINEBUFFER
%left MULOP DIVOP MODOP
%nonassoc TYPEASSIGNMENT
%right NOT
%left LEFTBRAC RIGHTBRACK
%left LEFTPAREN RIGHTPAREN
%start program
%type <Ast.program> program
%%
```

```
program:
  import_decl_list main_func_decl_option func_decl_list EOF { { ilist =
     List.rev $1 ; mainf = $2 ; declf = List.rev $3} }

main_func_decl_option:
  MAIN LEFTPAREN RIGHTPAREN body { { body = $4 } }

decl:
  ID TYPEASSIGNMENT data_type { Vdecl(Ast.IdTest($1),$3) }
  | ID TYPEASSIGNMENT LISTD recr_data_type { ListDecl(Ast.IdTest($1), $4) }
  | ID TYPEASSIGNMENT sp_data_type { ObjectCreate(Ast.IdTest($1), $3, []) }
  | ID TYPEASSIGNMENT MAPD data_type COMMA recr_data_type {
     MapDecl(Ast.IdTest($1),$4,$6) }

import_decl_list:
                                { [] }
  | import_decl_list import_decl { $2::$1 }

func_decl_list:
                                 { [] }
  | func_decl_list func_decl     { $2::$1 }

func_decl :
  ID LEFTPAREN decl_list RIGHTPAREN TYPEASSIGNMENT recr_data_type body {
    { rtype = $6 ; name = $1; formals = $3 ; body = $7; }
  }

import_decl:
   IMPORT LEFTPAREN STRING RIGHTPAREN SEMICOLON { Import($3) }

stmt_list:
   /* nothing */ { [] }
 | stmt_list stmt { $2 :: $1 }

decl_list:
   /* nothing */ { [] }
 | decl { [$1] }
```

```
 | decl COMMA decl_list { $1 :: $3 }

expr_list:
   /* nothing */ { [] }
  | expr { [$1] }
  | expr COMMA expr_list {$1 :: $3 }

body:
   LEFTBRACE stmt_list RIGHTBRACE { List.rev $2 }

function_call:
     ID LEFTPAREN expr_list RIGHTPAREN               { ($1, $3) }

stmt:
  | assign_stmt SEMICOLON                                          { $1 }
  | FORLOOP LEFTPAREN assign_stmt SEMICOLON expr_stmt SEMICOLON assign_stmt
     RIGHTPAREN body { For($3, $5, $7, $9) }
  | RETURN expr SEMICOLON                                          { Ret($2) }
  | function_call SEMICOLON                                        { CallStmt(fst
     $1,snd $1) }
  | v_decl                                                         { ($1) }
  | WHILELOOP LEFTPAREN expr_stmt RIGHTPAREN body                  { While($3, $5) }
  | ID TYPEASSIGNMENT sp_data_type LEFTPAREN expr_list RIGHTPAREN SEMICOLON {
     ObjectCreate(Ast.IdTest($1), $3, $5) }
  | ID TYPEASSIGNMENT LISTD data_type LEFTPAREN expr_list RIGHTPAREN SEMICOLON
     { ListInit(Ast.IdTest($1), $4, $6) }
  | IF LEFTPAREN expr_stmt RIGHTPAREN body elifs else_opt    {If({condition =
     $3; body = $5} :: $6, $7)}
  | ID ADDOP ASSIGN expr COMMA expr SEMICOLON                      {
     MapAdd(Ast.IdTest($1), $4, $6) }
  | ID SUBOP ASSIGN expr SEMICOLON                                 {
     MapRemove(Ast.IdTest($1), $4) }
  | ID ADDOP ASSIGN expr SEMICOLON                                 {
     ListAdd(Ast.IdTest($1), $4) }
  | ID SUBOP ASSIGN LEFTBRAC expr RIGHTBRAC SEMICOLON              {
     ListRemove(Ast.IdTest($1), $5) }
```

```
  | controlstmt SEMICOLON                                    { ControlStmt($1)
     }

controlstmt:
  | CONTINUE          { "Continue" }
  | BREAK             {  "Break" }

elifs:
  | {[]}
  | ELIF LEFTPAREN expr_stmt RIGHTPAREN body elifs { {condition = $3; body =
     $5} :: $6 }

else_opt:
  | {None}
  | ELSE body {Some($2)}

recr_data_type:
  | sp_data_type                                                {
     (Ast.TType($1)) }
  | data_type                                                   {
     (Ast.TType($1)) }
  | LISTD recr_data_type                                        {
     (Ast.RType($2)) }

v_decl :
| decl SEMICOLON                      { ($1) }

assign_stmt:
  ID ASSIGN expr                                        {
     Assign(Ast.IdTest($1), $3) }
| ID TYPEASSIGNMENT data_type ASSIGN expr                {
    InitAssign(Ast.IdTest($1),$3,$5) }
| list_access ASSIGN expr                               {
    ListAssign(ListAccess(fst $1, snd $1), $3) }

expr_stmt:
```

```
  expr EQ    expr                                          { Binop($1, Equal,
     $3) }
| expr NEQ   expr                                          { Binop($1, Neq,
    $3) }
| expr LT    expr                                          { Binop($1, Less,
    $3) }
| expr LEQ   expr                                          { Binop($1, Leq,
    $3) }
| expr GT    expr                                          { Binop($1,
    Greater, $3) }
| expr GEQ   expr                                          { Binop($1, Geq,
    $3) }
| expr AND expr                                            { Binop($1, And,
    $3)}
| expr OR expr                                             { Binop($1, Or, $3)}


data_type:
STRINGD                                                     { String }
| INTD                                                      { Int }
| FLOATD                                                    { Float }
| BOOLD                                                     { Bool }
| PDFD                                                      { Pdf }
| PAGED                                                     { Page }

sp_data_type:
LINED { Line }
| TUPLED { Tuple }
| IMAGED { Image }

expr:
STRING             { LitString($1) }
| INT              { LitInt($1) }
| FLOAT            { LitFloat($1)}
| BOOL             { LitBool($1) }
| ID                { Iden(Ast.IdTest($1)) }
| list_access      { ListAccess(fst $1,snd $1) }
```

```
| ID TYPEASSIGNMENT ASSIGN expr { MapAccess(Ast.IdTest($1), $4) }
| expr ADDOP expr  { Binop($1, Add, $3) }
| expr SUBOP expr  { Binop($1, Sub, $3) }
| expr MULOP expr  { Binop($1, Mul, $3) }
| expr DIVOP expr  { Binop($1, Div, $3) }
| expr CONCAT expr { Binop($1, Concat, $3) }
| expr MODOP expr  { Binop($1, Mod,   $3) }
| LEFTPAREN expr RIGHTPAREN { $2 }
| expr_stmt        { $1 }
| NOT expr         { Uop(Not,$2) }
| SUBOP expr       { Uop(Neg,$2) }
| expr LINEBUFFER  { Uop(LineBuffer,$1) }
| function_call   {CallExpr(fst $1,snd $1)}

list_access:
ID LEFTBRAC expr RIGHTBRAC { (Ast.IdTest($1), $3) }
```

AST

```
type binop = Add | Sub | Mul | Div | Mod | Equal | Neq | Less | Leq | Greater |
    Geq |
          And | Or | Swap | Append | Concat

type uop = Neg | Not | LineBuffer

type list_data_type = List

type id = IdTest of string

type recr_t =
  | TType of t
  | RType of recr_t

and t = Int | Bool | Float | String | Pdf | Page | Line | Tuple | Image |
    ListType of string | MapType of t * t
```

```
type var_decl = id * t


type map_decl = id * t * recr_t

type list_var_decl = id * recr_t

type expression =
  LitInt of int
  | LitString of string
  | Iden of id
  | LitFloat of float
  | LitBool of bool
  | Uop of uop * expression
  | Binop of expression * binop * expression
  | CallExpr of string * expression list
  | ListAccess of id * expression
  | MapAccess of id * expression

type statement =
  | ControlStmt of string
  | Ret of expression
  | While of expression * statement list
  | If of conditional list * statement list option
  | Assign of id * expression
  | ListAssign of expression * expression
  | Vdecl of var_decl
  | ListDecl of list_var_decl
  | MapDecl of map_decl
  | InitAssign of id * t * expression
  | ObjectCreate of id * t * expression list
  | ListInit of id * t * expression list
  | For of statement * expression * statement * statement list
  | CallStmt of string * expression list
  | MapAdd of id * expression * expression
  | MapRemove of id * expression
  | ListAdd of id * expression
```

```
  | ListRemove of id * expression

  and conditional = {
    condition : expression;
    body : statement list;
  }


type import_stmt =
  | Import of string

type func_decl = {
  rtype : recr_t;
  name : string;
  formals : statement list;
  body : statement list;
}

type main_func_decl = {
  body : statement list;
}

type program = {
  ilist : import_stmt list ;
  mainf : main_func_decl ;
  declf : func_decl list ;
}
```

Analyzer

```
open Sast
module StringMap = Map.Make(String);;

type symbol_table = {
  parent : symbol_table option;
    mutable variables : (string * Ast.t) list;
```

```
    mutable functions : (string * Ast.t) list;
}

type environment = {
  scope : symbol_table;      (* symbol table for vars *)
}

type type_map = {
  mutable map : string StringMap.t;
}

let str_eq a b = ((Pervasives.compare a b) = 0)

let rec find_variable (scope : symbol_table) (name : string) : Ast.t option =
  try
    let (_, typ) = List.find (fun (s, _) -> s = name) scope.variables in
    Some(typ)
  with Not_found ->
    match scope.parent with
    | Some(p) -> find_variable p name
    | _ -> None

let rec find_function (scope : symbol_table) (name : string) : Ast.t option =
  try
    let (_, typ) = List.find (fun (s, _) -> s = name) scope.functions in
    Some(typ)
  with Not_found ->
    match scope.parent with
    | Some(p) -> find_function p name
    | _ -> None

let is_keyword (name : string) : bool =
  let rec helper (name : string) (words : string list) : bool =
    match words with
    | [] -> false
    | h::t -> name = h || helper name t
  in
```

```
  helper name ["import";"main";"pdf";"page";"line";"renderpdf";"tuple";"list"]

let alphaCode = ref (Char.code 'A')
let betaCode = ref (Char.code 'A')

let next_type_var() : string =
  let c1 = !alphaCode in
  let c2 = !betaCode in
    if c2 = Char.code 'Z'
    then betaCode := Char.code 'a'
    else incr betaCode;
    if c2 = Char.code 'z'
    then (incr alphaCode; betaCode := Char.code 'A')
    else ();
    if c1 = Char.code 'Z'
    then alphaCode := Char.code 'a'
    else ();
    let name = (Char.escaped (Char.chr c1)) ^ (Char.escaped (Char.chr c2)) in
    name

let initialize_types(tmap : type_map) =
  let typeMap = StringMap.empty in
  let inttype = next_type_var() in
  let typeMap = StringMap.add "int" inttype typeMap in
  let booltype = next_type_var() in
  let typeMap = StringMap.add "bool" booltype typeMap in
  let floattype = next_type_var() in
  let typeMap = StringMap.add "float" floattype typeMap in
  let stringtype = next_type_var() in
  let typeMap = StringMap.add "string" stringtype typeMap in
  let pdftype = next_type_var() in
  let typeMap = StringMap.add "pdf" pdftype typeMap in
  let pagetype = next_type_var() in
  let typeMap = StringMap.add "page" pagetype typeMap in
  let linetype = next_type_var() in
  let typeMap = StringMap.add "line" linetype typeMap in
  let tupletype = next_type_var() in
```

```
  let typeMap = StringMap.add "tuple" tupletype typeMap in
  let lstype = next_type_var() in
  let typeMap = StringMap.add "AD" lstype typeMap in
  let lpagetype = next_type_var() in
  let typeMap = StringMap.add "AF" lpagetype typeMap in
  let lpdftype = next_type_var() in
  let typeMap = StringMap.add "AE" lpdftype typeMap in

  tmap.map <- typeMap


let initialize_predefined_functions (env : environment) =
    let lengthfn = ("length", Ast.Int) in
    env.scope.functions <- lengthfn :: env.scope.functions;
    let getpagesfn = ("getpages", Ast.ListType("AF")) in
    env.scope.functions <- getpagesfn :: env.scope.functions;
    let splitfn = ("split", Ast.ListType("AE")) in
    env.scope.functions <- splitfn :: env.scope.functions;
    let readtable = ("readtable", Ast.ListType("AI")) in
    env.scope.functions <- readtable :: env.scope.functions;
    let readtextfrompdf = ("readtextfrompdf", Ast.String) in
    env.scope.functions <- readtextfrompdf :: env.scope.functions;
    let drawpiechart = ("drawpiechart", Ast.Image) in
    env.scope.functions <- drawpiechart :: env.scope.functions;
    let drawbarchart = ("drawbarchart", Ast.Image) in
    env.scope.functions <- drawbarchart :: env.scope.functions;
    let loadpdf = ("loadpdf", Ast.Pdf) in
    env.scope.functions <- loadpdf :: env.scope.functions;
    let readfn = ("readtextfile", Ast.String) in
    env.scope.functions <- readfn :: env.scope.functions;
    let renderpdf = ("renderpdf", Ast.Int) in
    env.scope.functions <- renderpdf :: env.scope.functions;
    let print = ("print", Ast.Int) in
    env.scope.functions <- print :: env.scope.functions;
    let substr = ("substr", Ast.String) in
    env.scope.functions <- substr :: env.scope.functions;;
```

```
let nest_scope (env : environment) : environment =
  let s = {variables = []; functions = []; parent = Some(env.scope)} in
  {scope = s}

let new_env() : environment =
  let s = { variables = []; functions = []; parent = None } in
  {scope = s}

let new_map() : type_map =
  let m = StringMap.empty in
  {map = m}

let type_of (ae : Sast.texpression) : Ast.t =
  match ae with
  | TLitInt(_, t) -> t
  | TLitString(_, t) -> t
  | TLitFloat(_, t) -> t
  | TLitBool(_, t) -> t
  | TIden(_, t) -> t
  | TBinop(_, _, _, t) -> t
  | TListAccess(_, _, t) -> t
  | TMapAccess(_, _, t) -> t
  | TCallExpr(_, _, t) -> t
  | TUop(_, _, t) -> t

let find_type (t : string) (tmap : type_map) : string =
   let found = StringMap.mem t tmap.map in
   if found
   then
       StringMap.find t tmap.map
   else
       ""

let find_primitive_type (t : Ast.t) (tmap : type_map) : string =
  match t with
  | Ast.Int -> find_type "int" tmap
```

```
  | Ast.Bool -> find_type "bool" tmap
  | Ast.Float -> find_type "float" tmap
  | Ast.String -> find_type "string" tmap
  | Ast.Pdf -> find_type "pdf" tmap
  | Ast.Page -> find_type "page" tmap
  | Ast.Line -> find_type "line" tmap
  | Ast.Tuple -> find_type "tuple" tmap
  | Ast.Image -> find_type "image" tmap
  | _ -> failwith "You're doing something wrong! This shouldn't have been
      called."

let find_primitive_string (t : Ast.t) : string =
  match t with
  | Ast.Int -> "int"
  | Ast.Bool -> "bool"
  | Ast.Float -> "float"
  | Ast.String -> "string"
  | Ast.Pdf -> "pdf"
  | Ast.Page -> "page"
  | Ast.Line -> "line"
  | Ast.Tuple -> "tuple"
  | Ast.Image -> "image"
  | _ -> failwith "Data Type Not Primitive."

let find_primitive (s : string) : Ast.t =
  match s with
  | "int" -> Ast.Int
  | "bool" -> Ast.Bool
  | "float" -> Ast.Float
  | "string" -> Ast.String
  | "pdf" -> Ast.Pdf
  | "page" -> Ast.Page
  | "line" -> Ast.Line
  | "tuple" -> Ast.Tuple
  | "image" -> Ast.Image
  | _ -> failwith "Data Type Not Primitive"
```

```
let rec find_list_element_type (t : string) (tmap : type_map) : Ast.t =
  let rtmap = StringMap.fold (fun key value nmap -> StringMap.add value key
      nmap) tmap.map StringMap.empty in
  let found = StringMap.mem t rtmap in
  if found
  then
    let ftype = StringMap.find t rtmap in
    let f = StringMap.mem ftype rtmap in
    if f
    then
      Ast.ListType(ftype)
    else
      let ptype = find_primitive ftype in
      ptype
  else Ast.ListType("")

let rec annotate_expr (e : Ast.expression) (env : environment) (tmap :
    type_map) : Sast.texpression =
  match e with
  | Ast.LitInt(n) -> TLitInt(n, Ast.Int)
  | Ast.LitBool(n) -> TLitBool(n, Ast.Bool)
  | Ast.LitFloat(n) -> TLitFloat(n, Ast.Float)
  | Ast.LitString(n) -> TLitString(n, Ast.String)
  | Ast.Iden(s) ->
    (match s with
    | Ast.IdTest(w) ->
      let typ = find_variable env.scope w in
      (match typ with
      | Some(x) -> TIden(s,x)
      | None -> failwith ("Unrecognized identifier " ^ w ^ ".")))

  | Ast.Binop(e1,o,e2) ->
    let ae1 = annotate_expr e1 env tmap in
    let ae2 = annotate_expr e2 env tmap in
    let t1 = type_of ae1 in
    let t2 = type_of ae2 in
    if t1 = t2
```

```
    then
      (match o with
        | Ast.Add
        | Ast.Sub
        | Ast.Div
        | Ast.Swap
        | Ast.Append
        | Ast.Mod
        | Ast.Mul -> TBinop(ae1,o,ae2,t1)
        | Ast.Equal
        | Ast.Neq
        | Ast.Less
        | Ast.Leq
        | Ast.Greater
        | Ast.And
        | Ast.Or
        | Ast.Geq -> TBinop(ae1,o,ae2,Ast.Bool)
        | _ -> failwith "How you concat two same things?")
    else
     (match o with
        | Ast.Concat ->
          (match t1, t2 with
            | (Ast.Pdf, Ast.Page) -> TBinop(ae1,o,ae2,t1)
            | (Ast.Tuple, Ast.Line) -> TBinop(ae1,o,ae2,t1)
            | (Ast.Tuple,Ast.Image) -> TBinop(ae1,o,ae2,t1)
            | _ -> failwith "Oops")
        | Ast.Add ->
          (match t1,t2 with
            | (Ast.String, Ast.Int) -> TBinop(ae1,o,ae2,t1)
            | (Ast.Int, Ast.String) -> TBinop(ae1,o,ae2,t2)
            | _ -> failwith "Invalid Concatenation")
        | _ -> failwith "Incompatible types")

  | Ast.ListAccess(i,e) ->
      let ae = annotate_expr e env tmap in
      let t = type_of ae in
      (match t with
```

```
    | Ast.Int ->
        (match i with
        | Ast.IdTest(w) ->
            let typ = find_variable env.scope w in
            (match typ with
            | Some(x) ->
                (match x with
                | Ast.ListType(s) ->
                    let etype = find_list_element_type s tmap in
                    TListAccess(i,ae,etype)
                | _ -> failwith "Variable not List")
            | None -> failwith ("Unrecognized identifier " ^ w ^ ".")))
    | _ -> failwith "Invalid List Access Expression")
  | Ast.MapAccess(i, e) ->
      let ae = annotate_expr e env tmap in
      let t = type_of ae in
      (match i with
        | Ast.IdTest(w) ->
                let typ = find_variable env.scope w in
                (match typ with
                  | Some(x) ->
                        (match x with
                        | Ast.MapType(kd,vd) ->
                              if kd = t
                              then TMapAccess(i, ae, x)
                              else failwith "Incorrect type for access"
                        | _ -> failwith "Variable not Map" )
                  | None -> failwith ("Unrecognized identifier " ^ w ^ ".") )
                    )
  | Ast.CallExpr(e, elist) ->
    let et = find_function env.scope e in
    let aelist = List.map (fun x -> annotate_expr x env tmap) elist in
    (match et with
      | Some(x) -> TCallExpr(e, aelist, x)
      | None -> failwith "Did not find the type for this function" )
  | Ast.Uop(u,e) ->
       let ae = annotate_expr e env tmap in
```

```
          let t = type_of ae in
          match u with
          | Ast.LineBuffer -> TUop(u, ae, Ast.String)
          | _ -> TUop(u, ae, t)

and annotate_recr_type (rd : Ast.recr_t) (tmap : type_map) : string =
  (match rd with
    | Ast.TType(t) ->
      find_primitive_type t tmap
    | Ast.RType(r) ->
        let d = annotate_recr_type r tmap in
        let rt = find_type d tmap in
        (match rt with
          | "" ->
              let ard = next_type_var() in
              tmap.map <- StringMap.add d ard tmap.map;
              ard
          | _ ->
              rt))

and annotate_assign (i : Ast.id) (e : Ast.expression) (env : environment) (tmap
    : type_map) : Ast.id * Sast.texpression =
  let ae = annotate_expr e env tmap in
  let te = type_of ae in
  let id = match i with | Ast.IdTest (s) -> s in
  let tid = find_variable env.scope id in
  (match tid with
  | Some(idt) ->
      (match te with
      | Ast.MapType(kdt, vdt) ->
          if vdt = idt
          then i,ae
          else failwith "Invalid assignment."
      | _ ->
          if idt = te
          then i,ae
          else failwith "Invalid assignment.")
```

```
  | None -> failwith "Invalid assignment | Variable Not Found.")

and annotate_map_add (i : Ast.id) (e1 : Ast.expression) (e2 : Ast.expression)
    (env : environment) (tmap : type_map) : Ast.id * Sast.texpression *
    Sast.texpression =
  let ae1 = annotate_expr e1 env tmap in
  let ae2 = annotate_expr e2 env tmap in
  let te1 = type_of ae1 in
  let te2 = type_of ae2 in
  let id = match i with | Ast.IdTest (s) -> s in
  let tid = find_variable env.scope id in
  (match tid with
  | Some(idt) ->
      (match idt with
      | Ast.MapType(kidt,vidt) ->
          if kidt = te1
          then if vidt = te2
               then i,ae1,ae2
               else failwith "Invalid assignment | Value not Valid"
          else failwith "Invalid assignment | Key not Valid"
      | _ -> failwith "Invalid assignment | Variable not Map")
  | None -> failwith "Invalid assignment | Variable Not Found.")

and annotate_map_remove (i : Ast.id) (e : Ast.expression) (env : environment)
    (tmap : type_map) : Ast.id * Sast.texpression =
  let ae = annotate_expr e env tmap in
  let te = type_of ae in
  let id = match i with | Ast.IdTest (s) -> s in
  let tid = find_variable env.scope id in
  (match tid with
  | Some(idt) ->
      (match idt with
      | Ast.MapType(kidt,vidt) ->
          if kidt = te then i,ae
          else failwith "Invalid assignment | Key not Valid"
      | _ -> failwith "Invalid assignment | Variable not Map")
  | None -> failwith "Invalid assignment | Variable Not Found.")
```

```
and annotate_list_assign (e1 : Ast.expression) (e2 : Ast.expression) (env :
    environment) (tmap : type_map) : Sast.texpression * Sast.texpression =
  let ae1 = annotate_expr e1 env tmap in
  let ae2 = annotate_expr e2 env tmap in
  let et2 = type_of ae2 in
  (match ae1 with
  | TIden(lid,lt) ->
      let id = (match lid with Ast.IdTest(s) -> s) in
      let ltype = find_variable env.scope id in
      (match ltype with
      | Some(l) ->
          let ls = (match l with Ast.ListType(s) -> s | _ -> failwith "Should
              have been a list type") in
          let etype = find_list_element_type ls tmap in
          if etype = et2
          then ae1,ae2
          else failwith "Invalid Assignment | Type Mismatch"
      | None -> failwith "List Variable Not Found")
  | TListAccess(lid,lexpr,lt) ->
      if lt = et2
      then ae1,ae2
      else failwith "Invalid Assignment | Type Mismatch"
  | _ -> failwith "Invalid Assignment | Neither List nor ListAccess")

and add_scope_variable (i : Ast.id) (d : Ast.t) (env : environment) : unit =
  match i with
    | Ast.IdTest(s) ->
    if is_keyword s
      then failwith "Cannot assign keyword."
      else
      let typ = find_variable env.scope s in
      (match typ with
      | Some(t) ->
        failwith "Invalid assignment, already exists."
    | None ->
      env.scope.variables <- (s,d) :: env.scope.variables);
```

```
and annotate_stmt (s : Ast.statement) (env : environment) (tmap : type_map) :
    Sast.tstatement =
  match s with
  | Ast.ListInit(e,d,el) -> (match d with
                             | Ast.String
                             | Ast.Int
                             | Ast.Bool
                             | Ast.Pdf
                             | Ast.Page
                             | Ast.Float ->
                                add_scope_variable e d env;
                               let ad = d in
                               let ael = annotate_exprs el env tmap in
                               let ttt = TListInit(e,ad,ael) in
                               ttt
                             | _ -> failwith "Invalid Object Type.")
  | Ast.Ret(e) ->
      let ae = annotate_expr e env tmap in
      let typ = type_of ae in
      TRet(ae, typ)
  | Ast.ControlStmt(s) -> TControlStmt(s)
  | Ast.Assign(i, e) ->
      let (ae1, ae2) = annotate_assign i e env tmap in
      TAssign(ae1, ae2)
  | Ast.InitAssign(i,t,e) ->
      (match t with
      | Ast.Int
      | Ast.Bool
      | Ast.Float
      | Ast.String
      | Ast.Pdf
      | Ast.Page ->
          add_scope_variable i t env;
          let ae = annotate_expr e env tmap in
          TInitAssign(i,t,ae)
      | _ -> failwith "Invalid Assignment Type.")
```

```
| Ast.ListAssign(e1,e2) ->
    let (ae1, ae2) = annotate_list_assign e1 e2 env tmap in
    TListAssign(ae1,ae2)
| Ast.CallStmt(e, elist) ->
    let ae = e in
    let aet = find_function env.scope ae in
    (match aet with
    | Some(t) ->
        let aelist = List.map (fun x -> annotate_expr x env tmap) elist in
        TCallStmt(ae, aelist)
    | None -> failwith "Function Not in Scope")
| Ast.ListDecl(e,rd) ->
    let ard = annotate_recr_type rd tmap in
    let ld = Ast.ListType(ard) in
    add_scope_variable e ld env;
    TListDecl(e, ld)
| Ast.ListAdd(i,e) ->
    let ie = Ast.Iden(i) in
    let (t,ae) = annotate_list_assign ie e env tmap in
    (match t with
    | TIden(ti,tt) -> TListAdd(ti,ae)
    | _ -> failwith "Invalid Identifier Expression")
| Ast.ListRemove(i,e) ->
    let ae = annotate_expr e env tmap in
    let te = type_of ae in
    let id = match i with | Ast.IdTest(s) -> s in
    let tid = find_variable env.scope id in
    (match tid with
    | Some(idt) ->
        (match idt with
        | Ast.ListType(lt) ->
            (match te with
            | Ast.Int -> TListRemove(i,ae)
            | _ -> failwith "Invalid List Access")
        | _ -> failwith "Invalid assignment | Variable not List")
    | None -> failwith "Invalid assignment | Variable Not Found.")
| Ast.MapDecl(e, kd, vd) ->
```

```
    (match vd with
    | Ast.TType(x) ->
           let md = Ast.MapType(kd,x) in
           add_scope_variable e md env;
           TMapDecl(e, md)
    | Ast.RType(x) ->
           let rd = annotate_recr_type x tmap in
           let mrd = Ast.ListType(rd) in
           let md = Ast.MapType(kd,mrd) in
           add_scope_variable e md env;
           TMapDecl(e, md))
| Ast.MapAdd(i,e1,e2) ->
    let (t,ae1,ae2) = annotate_map_add i e1 e2 env tmap in
    TMapAdd(t,ae1,ae2)
| Ast.MapRemove(i,e) ->
    let (t,ae) = annotate_map_remove i e env tmap in
    TMapRemove(t,ae)
| Ast.Vdecl(e,d) ->
  add_scope_variable e d env;
    TVdecl(e, d)
| Ast.ObjectCreate(e,sd,el) ->
    (match sd with
    | Ast.Line
    | Ast.Image
    | Ast.Tuple ->
       add_scope_variable e sd env;
      let ad = sd in
      let ael = annotate_exprs el env tmap in
      let ttt = TObjectCreate(e,ad,ael) in
      ttt
    | _ -> failwith "Invalid Object Type.")
| Ast.While(e,sl) ->
    let nenv = nest_scope env in
    (match e with
    | Ast.Binop(e1,o,e2) ->
        (match o with
        | Ast.Equal
```

```
    | Ast.Neq
    | Ast.Less
    | Ast.Leq
    | Ast.Greater
    | Ast.Geq ->
        let ae1 = annotate_expr e1 nenv tmap in
        let ae2 = annotate_expr e2 nenv tmap in
        let te = TBinop(ae1,o,ae2,Ast.Bool) in
        let tsl = annotate_stmts sl nenv tmap in
        TWhile(te,tsl)
    | _ -> failwith "Invalid While Expression Type.")
    | _ -> failwith "Invalid While Expression Type.")
| Ast.If(cl,sl) ->
    let tcl = annotate_conds cl env tmap in
    (match sl with
    | Some(xsl) ->
        let nenv = nest_scope env in
        let tsl = annotate_stmts xsl nenv tmap in
        TIf(tcl,Some(tsl))
    | None -> TIf(tcl,None))
| Ast.For(s1,e,s2,sl) ->
    let nenv = nest_scope env in
    (match s1 with
    | Ast.Assign(i1,ie1) ->
        let aes1 = annotate_expr ie1 nenv tmap in
        let ets1 = type_of aes1 in
        (match ets1 with
        | Ast.Int ->
            let ts1 = annotate_stmt s1 nenv tmap in
            (match e with
            | Ast.Binop(e1,o,e2) ->
                (match o with
                | Ast.Equal
                | Ast.Neq
                | Ast.Less
                | Ast.Leq
                | Ast.Greater
```

```
                | Ast.Geq ->
                    let ae1 = annotate_expr e1 nenv tmap in
                          let ae2 = annotate_expr e2 nenv tmap in
                          let te = TBinop(ae1,o,ae2,Ast.Bool) in
                          (match s2 with
                          | Ast.Assign(i2,ie2) ->
                              let aes2 = annotate_expr ie2 nenv tmap in
                              let ets2 = type_of aes2 in
                              (match ets2 with
                              | Ast.Int ->
                                  let ts2 = annotate_stmt s2 nenv tmap in
                                  (*let (ae21,ae22) = annotate_assign i2 ie2
                                      nenv in
                                  let ts2 = TAssign(ae11,ae12) in*)
                                  let tsl = annotate_stmts sl nenv tmap in
                                  TFor(ts1,te,ts2,tsl)
                              | _ -> failwith "Invalid Assignment Expression
                                  Type.")
                          | _ -> failwith "Invalid For Statement.")
                | _ -> failwith "Invalid For Expression Type.")
            | _ -> failwith "Invalid For Expression Type.")
        | _ -> failwith "Invalid Assignment Expression Type.")
    | _ -> failwith "Invalid For Statement.")

and annotate_func_decl (fdecl : Ast.func_decl) (env : environment) (tmap :
   type_map) : Sast.tfunc_decl =
  let retType =
  (match fdecl.Ast.rtype with
  | Ast.TType(t) -> t
  | Ast.RType(r) -> let art = annotate_recr_type r tmap in Ast.ListType(art)) in
  env.scope.functions <- (fdecl.Ast.name , retType) :: env.scope.functions;
  let s = {variables = []; functions = []; parent = Some(env.scope)} in
  let fenv = {scope = s} in
  let aes = annotate_stmts fdecl.Ast.formals fenv tmap in
  let asts = annotate_stmts fdecl.Ast.body fenv tmap in
  {rtype = retType; name = fdecl.Ast.name; tformals = aes; tbody = asts}
```

```
and annotate_main_func_decl (mdecl : Ast.main_func_decl) (env : environment)
    (tmap : type_map) : Sast.tmain_func_decl =
  let asts = annotate_stmts mdecl.Ast.body env tmap in
  {tbody = asts}

and annotate_import_statement (istmt : Ast.import_stmt) (env : environment)
    (tmap : type_map) : Sast.tprogram =
  (match istmt with
  | Ast.Import(s) ->
      let l = String.length s in
      let sl = 1 in
      let el = l-2 in
      let is = String.sub s sl el in
      let aip = parse_file is in
      aip)

and annotate_cond (cond: Ast.conditional) (env : environment) (tmap : type_map)
    : Sast.tconditional =
  let ae = annotate_expr cond.Ast.condition env tmap in
  let t = type_of ae in
  (match t with
  | Ast.Bool ->
      let nenv = nest_scope env in
      let tsl = annotate_stmts cond.Ast.body nenv tmap in
      {tcondition = ae; tbody = tsl}
  | _ -> failwith "Invalid For Statement.")

and annotate_conds (conds : Ast.conditional list) (env : environment) (tmap :
    type_map) : Sast.tconditional list =
  List.map (fun i -> annotate_cond i env tmap) conds

and annotate_import_statements (istmts : Ast.import_stmt list) (env :
    environment) (tmap : type_map) : Sast.tprogram list =
  List.map (fun i -> annotate_import_statement i env tmap) istmts

and annotate_exprs (exprs : Ast.expression list) (env : environment) (tmap :
    type_map) : Sast.texpression list =
```

```
  List.map (fun s -> annotate_expr s env tmap) exprs

and annotate_stmts (stmts : Ast.statement list) (env : environment) (tmap :
    type_map) : Sast.tstatement list =
  List.map (fun x -> annotate_stmt x env tmap) stmts

and annotate_func_decls (fdecls : Ast.func_decl list) (env : environment) (tmap
    : type_map) : Sast.tfunc_decl list =
  List.map (fun f -> annotate_func_decl f env tmap) fdecls

and parse_file (fname : string) : Sast.tprogram =
  let file = open_in fname in
  let lexbuf = Lexing.from_channel file in
  let program = Parser.program Lexer.token lexbuf in
  let annotatedProgram = annotate_prog program in
  annotatedProgram

and extract_function (itp : Sast.tprogram) (env : environment) (tmap :
    type_map) : Sast.tfunc_decl list =
    let m = StringMap.fold (fun key value newMap -> StringMap.add value key
        newMap) itp.tmap StringMap.empty in
    let mergedMap = StringMap.merge (fun k v1 v2 ->
                        match v1,v2 with
                        | Some(v1), Some(v2) -> Some(v2)
                        | Some(v1), None -> Some(v1)
                        | None, Some(v2) -> Some(v2)
                        | _ -> None) m tmap.map in
    tmap.map <- mergedMap;
   let fdecls = itp.tdeclf in
   let _ = List.map (fun f -> env.scope.functions <- (f.name , f.rtype) ::
       env.scope.functions) fdecls in
   fdecls

and extract_functions (itps : Sast.tprogram list) (env : environment) (tmap :
    type_map) : Sast.tfunc_decl list =
   let l = List.map (fun f -> extract_function f env tmap) itps in
   let atf = List.fold_left (fun acc x -> List.append acc x) [] l in
```

```
   atf

and annotate_prog (p : Ast.program) : Sast.tprogram =
  let env = new_env() in
  initialize_predefined_functions(env);
  let tmap = new_map() in
  initialize_types tmap;
  let ai = annotate_import_statements p.Ast.ilist env tmap in
  let ef = extract_functions ai env tmap in
  let f = annotate_func_decls p.Ast.declf env tmap in
  let af = List.append ef f in
  let am = annotate_main_func_decl p.Ast.mainf env tmap in
  let revMap = StringMap.fold (fun key value newMap -> StringMap.add value key
     newMap) tmap.map StringMap.empty in
  Printf.printf "There there\n";
  {tmap = revMap; tmainf = am; tdeclf = af}
```

SAST

```
open Ast
module StringMap = Map.Make(String);;

type texpression =
  TLitInt of int * t
  | TLitString of string * t
  | TIden of Ast.id * t
  | TLitFloat of float * t
  | TLitBool of bool * t
  | TUop of Ast.uop * texpression * t
  | TBinop of texpression * Ast.binop * texpression * t
  | TCallExpr of string * texpression list * t
  | TListAccess of Ast.id * texpression * t
  | TMapAccess of Ast.id * texpression * t
```

```
type tstatement =
  | TRet of texpression * t
  | TControlStmt of string
  | TWhile of texpression * tstatement list
  | TIf of tconditional list * tstatement list option
  | TAssign of Ast.id * texpression
  | TListAssign of texpression * texpression
  | TVdecl of Ast.var_decl
  | TListDecl of Ast.id * Ast.t
  | TMapDecl of Ast.id * Ast.t
  | TInitAssign of Ast.id * Ast.t * texpression
  | TObjectCreate of Ast.id * Ast.t * texpression list
  | TFor of tstatement * texpression * tstatement * tstatement list
  | TCallStmt of string * texpression list
  | TMapAdd of Ast.id * texpression * texpression
  | TMapRemove of Ast.id * texpression
  | TListAdd of Ast.id * texpression
  | TListRemove of Ast.id * texpression
  | TListInit of Ast.id * Ast.t * texpression list

  and tconditional = {
    tcondition : texpression;
    tbody : tstatement list;
  }

type tfunc_decl = {
  rtype : Ast.t;
  name : string;
  tformals : tstatement list;
  tbody : tstatement list;
}

type tmain_func_decl = {
  tbody : tstatement list;
}

type tprogram = {
```

```
  tmap : string StringMap.t;
  tmainf : tmain_func_decl;
  tdeclf : tfunc_decl list;
}
```

Codegen

```
open Sast
open Ast
open Printf
open Random
module StringMap = Map.Make(String);;

(************
  HELPERS
************)


let rec getJavaType typ typemap =
match typ with
  | Int -> "Integer"
  | Bool -> "Boolean"
  | Float -> "Float"
  | String -> "String"
  | Pdf -> "PDDocument"
  | Page -> "PDPage"
  | Line -> "Line"
  | Tuple -> "Tuple"
  | Image -> "Image"
  | ListType(l) -> makeLists l typemap
  | MapType(k,v) ->  let keytype =
                                    (match k with
                                          | Int -> "Integer"
                                          | Bool -> "Boolean"
                                          | Float -> "Float"
                                          | String -> "String"
```

```
                                        | Pdf -> "PDDocument"
                                        | Page -> "PDPage"
                                        | Line -> "Line"
                                        | Tuple -> "Tuple"
                                        | Image -> "Image"
                                        | _ -> "Key type can't be list or map")
                                           in
                                    let valuetype =
                                        ( match v with
                                          | ListType(x) ->
                                                   let acc = makeLists x typemap
                                                       in
                                                   acc
                                          | Int -> "Integer"
                                          | Bool -> "Boolean"
                                          | Float -> "Float"
                                          | String -> "String"
                                          | Pdf -> "PDDocument"
                                          | Page -> "PDPage"
                                          | Line -> "Line"
                                          | Tuple -> "Tuple"
                                          | Image -> "Image"
                                          | _  -> "value type can't be map"   )
                                           in "Map<" ^keytype
                                           ^","^valuetype^">"

and makeLists (typeid : string) (typemap) : string =
      let found = StringMap.mem typeid typemap in
      if found
      then let foundType = StringMap.find typeid typemap in
      let recurseType = makeLists foundType typemap in
      let liststring = "List<" ^ recurseType ^ ">" in
      liststring
      else
          (match typeid with
          | "int" -> "Integer"
          | "bool" -> "Boolean"
```

```
          | "float" -> "Float"
          | "string" -> "String"
          | "pdf" -> "PDDocument"
          | "page" -> "PDPage"
          | "line" -> "Line"
          | "tuple" -> "Tuple"
          | "image" -> "Image"
          | _ -> failwith "Type not found" )

let type_of (ae : Sast.texpression) : Ast.t =
  match ae with
  | TLitInt(_, t) -> t
  | TLitFloat(_, t) -> t
  | TLitString(_, t) -> t
   | TLitBool(_, t) -> t
  | TUop(_, _, t) -> t
  | TCallExpr(_, _, t) -> t
  | TBinop(_, _, _, t) -> t
   | TIden(_,t) -> t
  | TListAccess(_, _, t) -> t
  | TMapAccess (_, _, t) -> t

let java_from_type (ty: Ast.t) : string =
    match ty with
      | _ -> "PrimitiveObject"


let writeId iden =
   sprintf "%s" iden

let writeIntLit intLit =
  sprintf "%d" intLit

let writeFloatLit floatLit =
  sprintf "new Float(%f)" floatLit

let writeBoolLit boolLit =
```

```
  sprintf "new Boolean(%b)" boolLit

let writeStringLit stringLit =
  sprintf "%s" stringLit

let rec writeJavaProgramToFile fileName programString =
   let file = open_out ("javagen/" ^ fileName ^ ".java") in
      fprintf file "%s" programString


  (*and generateFunctionList prog =
  let concatenatedFunctions = List.fold_left (fun a b -> a ^
     (generateFunctionDefinitions b)) "" prog in
   sprintf "%s" concatenatedFunctions
  and generateFunctionDefinitions = function
       tmainf(stmtList) -> writeMainFunction stmtList
     | failwith "Not handled"*)



let rec writeBinop expr1 op expr2 =
  let e1 = generateExpression expr1 and e2 = generateExpression expr2 in
    let type1 = type_of expr1 in
     let type2 = type_of expr2 in
     let writeBinopHelper e1 op e2 =
     (match op with
        Add ->
            if type1 = type2
              then sprintf "%s + %s" e1 e2
            else
              (match type1, type2 with
              | (Ast.String, Ast.Int) -> sprintf "%s + Integer.toString(%s)" e1
                  e2
              | (Ast.Int, Ast.String) -> sprintf "Integer.toString(%s) + %s" e1
                  e2
              | _ -> failwith "Invalid Concatenation")
      | Sub -> sprintf "%s - %s" e1 e2
```

```
      | Mul -> sprintf "%s * %s" e1 e2
      | Div -> sprintf "%s / %s" e1 e2
      | Equal -> sprintf "%s == %s" e1 e2
      | Neq -> sprintf "%s != %s" e1 e2
      | Less -> sprintf "%s < %s" e1 e2
      | Leq -> sprintf "%s <= %s" e1 e2
      | Greater -> sprintf "%s > %s" e1 e2
      | Geq -> sprintf "%s >= %s" e1 e2
      | Mod -> sprintf "%s || %s" e1 e2
      | And -> sprintf "%s && %s" e1 e2
      | Or -> sprintf "%s || %s" e1 e2
      | Swap -> sprintf "%s || %s" e1 e2
      | Append -> sprintf "%s || %s" e1 e2
      | Concat ->
              match type1 with
              | Pdf ->
                      (match type2 with
                        | Page -> sprintf "Util.addPageToPDF(%s,%s);\n" e1 e2
                        | _ -> failwith "Not handled")
      | Tuple ->
      (match type2 with
          | Line ->
               sprintf "Util.addLineToTuple(%s,%s)" e1 e2
          | Image ->
               sprintf "Util.addImageToTuple(%s, %s)" e1 e2
          | _ -> failwith "Not handled" )
      | _ -> failwith "Something went wrong!")
    in writeBinopHelper e1 op e2

and writeUop expr1 op =
  let e1 = generateExpression expr1 in
  let writeUopHelper e1 op = match op with
    | LineBuffer -> sprintf "%s.getRemainingText();" e1
    | Neg -> sprintf "-%s" e1
    | Not -> sprintf "!%s" e1
  in writeUopHelper e1 op
```

```
and writeObjectStmt tid tspDataType tExprList =
let idstring =
  (match tid with
   | IdTest(s) -> s ) in
 (match tspDataType with
 | Line -> (match tExprList with
         [] -> sprintf "Line %s = new Line();\n %s.setFont(\"TIMES_ROMAN\");\n
            %s.setText(\"Hello World\");\n %s.setXcod(100);\n
            %s.setYcod(700);\n %s.setFontSize(12);\n %s.setWidth(500);\n"
            idstring idstring idstring idstring idstring idstring idstring
         | _ ->
         let exprMapForLine = getExpressionMap tExprList in
         let drawString = StringMap.find "1" exprMapForLine in
         let font = StringMap.find "2" exprMapForLine in
         let fontSize = StringMap.find "3" exprMapForLine in
         let xcod = StringMap.find "4" exprMapForLine in
         let ycod = StringMap.find "5" exprMapForLine in
         let width = StringMap.find "6" exprMapForLine in
         sprintf "Line %s = new Line();\n %s.setFont(%s);\n %s.setText(%s);\n
            %s.setXcod(%s);\n %s.setYcod(%s);\n %s.setFontSize(%s);\n
            %s.setWidth(%s);\n" idstring idstring font idstring drawString
            idstring xcod idstring ycod idstring fontSize idstring width)
 | Tuple ->
 let exprMapForTuple = getExpressionMap tExprList in
 let pdfIden = StringMap.find "1" exprMapForTuple in
 let pageIden = StringMap.find "2" exprMapForTuple in
  sprintf "Tuple %s = new Tuple(%s,%s);\n" idstring pdfIden pageIden
 | Image -> (match tExprList with
         [] -> sprintf "\nFile file = new File(\"\"); \nImage %s = new
            Image(file,%s,%s,%s,%s);\n" idstring "800" "600" "100" "600"
         | _ ->
          let exprMapForImage = getExpressionMap tExprList in
          let fileLoc = StringMap.find "5" exprMapForImage in
          let xcood = StringMap.find "4" exprMapForImage in
          let ycood = StringMap.find "3" exprMapForImage in
          let height = StringMap.find "2" exprMapForImage in
          let width = StringMap.find "1" exprMapForImage in
```

```
          let fileVar = idstring^"file" in
          sprintf "\nFile %s = new File(\"%s\"); \nImage %s = new
              Image(%s,%s,%s,%s,%s);\n" fileVar fileLoc idstring fileVar height
              width xcood ycood)
 | _ -> failwith "Something went wrong")

and writeListInit tid tdataype tExprList typemap =
let idstring =
  (match tid with
   | IdTest(s) -> s ) in
    let ttype = getJavaType tdataype typemap in
    let expressionListString = List.fold_left (fun a b -> a ^
        (generateExpression b)^ ",") "" tExprList in
    let argList = String.sub expressionListString 0 ((String.length
        expressionListString) - 1) in
    sprintf "List<%s> %s = new ArrayList<%s>(Arrays.asList(%s));\n" ttype
        idstring ttype argList



 (*and writeObjectStmt tid tspDataType tExprList =
 let idstring =
   (match tid with
    | IdTest(s) -> s ) in
  match tspDataType with
  | Line ->
  let drawString = "Hello World" in
  let font = "TIMES_ROMAN" in
  let fontSize = 12 in
  let xcod = 100 in
  let ycod = 600 in
  sprintf "Line %s = new Line();\n %s.setFont(\"%s\");\n %s.setText(\"%s\");\n
      %s.setXcod(%d);\n %s.setYcod(%d);\n %s.setFontSize(%d);\n" idstring
      idstring font idstring drawString idstring xcod idstring ycod idstring
      fontSize
  | Tuple ->
  sprintf "Tuple %s = new Tuple(%s,%s);\n" idstring "pdfVar" "pageVar"
```

```
  | _ -> failwith "Something went wrong"
*)


(*and getExpressionMap exprList =
let exprMap = StringMap.empty in
StringMap.add "1" "Test" exprMap;
StringMap.add "2" "Test" exprMap;
StringMap.add "3" "Test" exprMap;
StringMap.add "4" "Test" exprMap;
StringMap.add "5" "Test" exprMap;
exprMap*)

and getExpressionMap exprList =
let rec access_list exprMap exprList index =
match exprList with
| [] -> exprMap
| head::body ->
(
let indexString = string_of_int index in
let value = generateExpression head in
let exprMap = StringMap.add indexString value exprMap in
let nextIndex = (index + 1) in
access_list exprMap body nextIndex
)
in access_list StringMap.empty exprList 1;


and getFuncExpressionMap exprList =
let rec access_list funcExprMap exprList index =
match exprList with
| [] -> funcExprMap
| head::body ->
(
let indexString = string_of_int index in
let value = generateExpression head in
let funcExprMap = StringMap.add indexString value funcExprMap in
```

```
let nextIndex = (index + 1) in
access_list funcExprMap body nextIndex
)
in access_list StringMap.empty exprList 1;


and writeFunctionCallExpr name exprList =
match name with
| "length" -> let identifier = List.hd exprList in
 ( match identifier with
  |  TIden(n, t) -> (
      let name =
      ( match n with
      |IdTest(n) -> n) in
    match t with
    | String -> sprintf "%s.length()" name
    | ListType(x) -> sprintf "%s.size()" name
    | MapType(t,x) -> sprintf "%s.size()" name
    | _ -> failwith "How dare you ask for length of this type?"
  )
  | _ -> failwith "expecting an identifier"
 )
| "readtextfile" -> let funcExprMap = getFuncExpressionMap exprList in
let location = StringMap.find "1" funcExprMap in
sprintf "\n Util.readFile(%s)\n" location
| "drawpiechart" -> let funcExprMapForPieChart = getFuncExpressionMap exprList
    in
let dataList = StringMap.find "1" funcExprMapForPieChart in
let attributeMap = StringMap.find "2" funcExprMapForPieChart in
sprintf "\n Util.drawPieChart(%s, %s)\n" dataList attributeMap
| "drawbarchart" -> let funcExprMapForPieChart = getFuncExpressionMap exprList
    in
let dataList = StringMap.find "1" funcExprMapForPieChart in
let attributeMap = StringMap.find "2" funcExprMapForPieChart in
sprintf "\n Util.drawBarChart(%s, %s)\n" dataList attributeMap
| "readtable" -> let funcExprMapForTable = getFuncExpressionMap exprList in
let location = StringMap.find "1" funcExprMapForTable in
```

```
let pagenumberList = StringMap.find "2" funcExprMapForTable in
sprintf "\n Util.readTable(%s, %s)\n" location pagenumberList
| "readtextfrompdf" -> let funcExprMapForTable = getFuncExpressionMap exprList
    in
let location = StringMap.find "1" funcExprMapForTable in
let pagenumberList = StringMap.find "2" funcExprMapForTable in
sprintf "\n Util.readTextFromPdf(%s, %s)\n" location pagenumberList
| "getpages" -> let funcExprMap = getFuncExpressionMap exprList in
let pdffile = StringMap.find "1" funcExprMap in
sprintf "\n Util.getPages(%s)\n" pdffile
| "loadpdf" -> let funcExprMap = getFuncExpressionMap exprList in
let pdffile = StringMap.find "1" funcExprMap in
sprintf "\n Util.loadPdf(%s)\n" pdffile
| "split" -> let funcExprMap = getFuncExpressionMap exprList in
let pdffile = StringMap.find "2" funcExprMap in
let varList = StringMap.find "1" funcExprMap in
sprintf "\n Util.splitPdf(%s,%s)" pdffile varList
| "substr" -> let funcExprMap = getFuncExpressionMap exprList in
let stringVar = StringMap.find "1" funcExprMap in
let startIndex = StringMap.find "2" funcExprMap in
let endIndex = StringMap.find "3" funcExprMap in
sprintf "\n Util.substr(%s,%s,%s)" stringVar startIndex endIndex
| _ ->
let expressionListString = List.fold_left (fun a b -> a ^ (generateExpression
    b)^ ",") "" exprList in
let argList = String.sub expressionListString 0 ((String.length
    expressionListString) - 1) in
sprintf "\n%s(%s);" name argList

and gPE s t =
   match t with
   Ast.Int -> let e = "Integer.toString(" ^ s ^ ")" in e
   | Ast.Float -> let e = "Float.toString(" ^ s ^ ")" in e
   | Ast.Bool -> let e = "Boolean.toString(" ^ s ^ ")" in e
   | Ast.String -> s
   | _ -> failwith "Invalid Print Type"
```

```
and generatePrintExpression = function
  TBinop(ope1, op, ope2, t) -> let w = writeBinop ope1 op ope2 in let g = gPE w
      t in g
   | TUop(op,ope1, t) -> let w = writeUop ope1 op in let g = gPE w t in g
   | TLitString(stringLit, t) -> let w = writeStringLit stringLit in let g =
       gPE w t in g
   | TLitInt(intLit, t) -> let w = writeIntLit intLit in let g = gPE w t in g
   | TLitFloat (floatLit, t) -> let w = writeFloatLit floatLit in let g = gPE w
       t in g
   | TLitBool(boolLit, t) -> let w = writeBoolLit boolLit in let g = gPE w t in
       g
   | TCallExpr(name, exprList, t) -> let w = writeFunctionCallExpr name
       exprList in let g = gPE w t in g
   | TIden(name, t) ->
   (match name with
   |IdTest(n) -> let w = writeId n in let g = gPE w t in g
    )
   | TListAccess(tid, tex, t) -> let w = writeListAccess tid tex in let g = gPE
       w t in g
   | TMapAccess (tid, tex, t) -> let w = writeMapAccess tid tex in let g = gPE
       w t in g

and writeFunctionCallStmt name exprList =
match name with
| "renderpdf" -> let funcExprMap = getFuncExpressionMap exprList in
let pdfIden = StringMap.find "1" funcExprMap in
let location = StringMap.find "2" funcExprMap in
sprintf "\n%s.save(%s);\n %s.close();" pdfIden location pdfIden
| "print" ->
    let expressionListString = List.fold_left (fun a b -> a ^
        (generatePrintExpression b)^ ",") "" exprList in
    let argList = String.sub expressionListString 0 ((String.length
        expressionListString) - 1) in
    sprintf "System.out.printf(%s);\n" argList
| _ ->
let expressionListString = List.fold_left (fun a b -> a ^ (generateExpression
    b)^ ",") "" exprList in
```

```
let argList = String.sub expressionListString 0 ((String.length
    expressionListString) - 1) in
sprintf "\n%s(%s);" name argList


and writeInitAssignStmt iden t expression =
let expressionString = generateExpression expression in
let name =
( match iden with
| IdTest(n ) -> n ) in
match t with
| Int -> sprintf "\nInteger %s = %s;" name expressionString
| String -> sprintf "\nString %s = %s;" name expressionString
| Bool -> sprintf "\nBoolean %s = %s;" name expressionString
| Float -> sprintf "\nFloat %s = %s;" name expressionString
| Pdf -> sprintf "\nPDDocument %s = %s" name expressionString
| _ -> failwith "initialization not possible for this type"



and writeControlStmt name =
match name with
| "Continue" -> sprintf "\ncontinue;"
| "Break" -> sprintf "\nbreak;"
| _ -> failwith "undefined control statement"


(*and writeFunctionCallStmt name exprList =
match name with
| "renderpdf" ->
let pdfIden = "pdfVar" in
let location = "helloworld.pdf" in
sprintf "\n%s.save(\"%s\");\n %s.close();" pdfIden location pdfIden
| _ -> failwith "undefined function"*)

 and writeListAccess tid texpression =
   let gexpr = generateExpression texpression in
```

```
   match tid with
   | IdTest(x) -> sprintf "%s.get(%s)" x gexpr

 and writeMapAccess tid texpression =
   let gexpr = generateExpression texpression in
   match tid with
   | IdTest(x) -> sprintf "%s.get(%s)" x gexpr

 and generateExpression = function
     TBinop(ope1, op, ope2, _) -> writeBinop ope1 op ope2
   | TUop(op,ope1, _) -> writeUop ope1 op
   | TLitString(stringLit, _) -> writeStringLit stringLit
   | TLitInt(intLit, _) -> writeIntLit intLit
   | TLitFloat (floatLit, _) -> writeFloatLit floatLit
   | TLitBool(boolLit, _) -> writeBoolLit boolLit
   | TCallExpr(name, exprList, _) -> writeFunctionCallExpr name exprList
   | TIden(name, _) ->
   (match name with
   |IdTest(n) -> writeId n
    )
   | TListAccess(tid, tex, _) -> writeListAccess tid tex
   | TMapAccess (tid, tex, _) -> writeMapAccess tid tex


let rec writeAssignmentStmt id expr2 =
        let e2string = generateExpression expr2 in
        match id with
             IdTest(n) -> sprintf "%s = %s;\n" n e2string




let rec writeDeclarationStmt tid tdataType typemap =
  match tid with
      | IdTest(name) ->
                      (match tdataType with
```

```
| Pdf -> sprintf "PDDocument %s = new
    PDDocument();\n" name
| Page -> sprintf "PDPage %s = new PDPage();\n"
    name
| Int -> sprintf "Integer %s = new
    Integer(0);\n" name
| Float -> sprintf "Float %s = new Float(0.0);
    \n" name
| Bool -> sprintf "Boolean %s = new
    Boolean(true);\n" name
| String -> sprintf "String %s = new
    String();\n" name
| ListType(x) ->
  let acc = makeLists x typemap in
  sprintf "%s %s = new Array%s(); \n" acc name
     acc
| MapType(k,v) ->
    let keytype =
    (   match k with
          | Int -> "Integer"
          | Bool -> "Boolean"
          | Float -> "Float"
          | String -> "String"
          | Pdf -> "PDDocument"
          | Page -> "PDPage"
          | Line -> "Line"
          | Tuple -> "Tuple"
          | Image -> "Image"
          | _ -> failwith "Can't use Lists or
              Maps as keys") in
    let valuetype =
        ( match v with
          | ListType(x) ->
                    let acc = makeLists x typemap
                        in
                    acc
          | Int -> "Integer"
```

```
                                        | Bool -> "Boolean"
                                        | Float -> "Float"
                                        | String -> "String"
                                        | Pdf -> "PDDocument"
                                        | Page -> "PDPage"
                                        | Line -> "Line"
                                        | Tuple -> "Tuple"
                                        | Image -> "Image"
                                        | _ -> failwith "Can't put map type as
                                            value type") in
                                    sprintf "Map<%s,%s> %s = new
                                        HashMap<%s,%s>(); \n" keytype
                                        valuetype name keytype valuetype


                                | _ -> failwith "Can't declare list, tuple or
                                    image" )


and writeListAssign lexpr texpression =
     let genexpr = generateExpression texpression in
     (match lexpr with
     | TListAccess(tid, texpr, t) -> (match tid with
                                  | IdTest(name) -> let texprs =
                                      generateExpression texpr in sprintf
                                      "%s.add(%s,%s);\n" name texprs genexpr)
     | _ -> "Not a list access" )
and writeListAdd tid texpression typemap =
     let genexpr = generateExpression texpression in
     match tid with
     | IdTest(name) -> sprintf "%s.add(%s); \n" name genexpr

and writeListRemove tid texpression typemap =
     let genexpr = generateExpression texpression in
     match tid with
     | IdTest(name) -> sprintf "%s.remove(%s); \n" name genexpr
```

```
and writeMapAdd tid texpression1 texpression2 =
      let genexpr1 = generateExpression texpression1 in
      let genexpr2 = generateExpression texpression2 in
      match tid with
      | IdTest(name) -> sprintf "%s.put(%s,%s); \n" name genexpr1 genexpr2

and writeMapRemove tid texpression =
      let genexpr = generateExpression texpression in
      match tid with
      | IdTest(name) -> sprintf "%s.remove(%s); \n" name genexpr


 and generateStatement tstatement typemap =
 match tstatement with
     | TVdecl(tid, tdataType) -> writeDeclarationStmt tid tdataType typemap
     | TAssign(tid, tExpression ) -> writeAssignmentStmt tid tExpression
     | TObjectCreate(tid, tspDataType, tExprList ) -> writeObjectStmt tid
        tspDataType tExprList
     | TCallStmt(name, exprList ) -> writeFunctionCallStmt name exprList
     | TInitAssign(iden, t, expression) -> writeInitAssignStmt iden t expression
     | TFor(initStmt, condition, incrStmt, body) -> writeForLoopStatement
        initStmt condition incrStmt body typemap
     | TWhile(condition, body) -> writeWhileStatement condition body typemap
     | TListAssign (lac, lvexpr) -> writeListAssign lac lvexpr
     | TIf(conditionStmtList, elsestmtList) -> writeIfBlock conditionStmtList
        elsestmtList typemap
     | TControlStmt(name) -> writeControlStmt name
     | TListDecl(tid, tdataype) -> writeDeclarationStmt tid tdataype typemap
     | TListAdd(tid, texpr) -> writeListAdd tid texpr typemap
     | TListRemove(tid, texpr) -> writeListRemove tid texpr typemap
     | TMapDecl(tid, tdataype) -> writeDeclarationStmt tid tdataype typemap
     | TMapAdd(tid, texpr1, texpr2) -> writeMapAdd tid texpr1 texpr2
     | TMapRemove(tid, texpr) -> writeMapRemove tid texpr
     | TRet(texpr,t) -> writeReturnStatement texpr
     | TListInit(tid, tdatatype, tExprList ) -> writeListInit tid tdatatype
        tExprList typemap
```

```
and writeReturnStatement tExpression =
let exprString = generateExpression tExpression in
sprintf "\nreturn %s;" exprString


and writeStmtList stmtList typemap =
let outStr = List.fold_left (fun a b -> a ^ (generateStatement b typemap)) ""
    stmtList in
sprintf "%s" outStr


and generateConditionStmt conditionalList index typemap =
  match conditionalList with
   [] -> []
   | a::l -> let ifExpression = generateExpression a.tcondition in
   let body = writeStmtList a.tbody typemap in
    match index with
   | 1  -> sprintf "\n if (%s) \n{ \n %s \n}" ifExpression body ::
       generateConditionStmt l (index+1) typemap
   | _ -> sprintf "\n else if (%s) \n{ \n %s \n}" ifExpression body ::
       generateConditionStmt l (index+1) typemap

and generateConditionalList conditionList typemap=
  let concatenatedConditionalsList = generateConditionStmt conditionList 1
      typemap in
   let concatenatedConditionals = List.fold_left (fun a b -> a ^ b ) ""
       concatenatedConditionalsList in
    sprintf "%s" concatenatedConditionals

and writeElseStmt body typemap =
let bodyString = writeStmtList body typemap in
sprintf " else \n{ %s \n}" bodyString


and writeIfBlock conditionList elseBody typemap =
let conditionListString = generateConditionalList conditionList typemap in
```

```
match elseBody with
Some(x) -> let elseBodyString = writeElseStmt x typemap in
sprintf " %s \n %s " conditionListString elseBodyString
| None -> sprintf " %s " conditionListString


and writeForLoopStatement initStmt condition incrStmt body typemap =
let exprString = generateExpression condition in
let initStmtString = generateStatement initStmt typemap in
let incrStmtString = generateStatement incrStmt typemap in
let incrStmtSubString = String.sub incrStmtString 0 ((String.length
    incrStmtString) - 2) in
let bodyString = writeStmtList body typemap in
sprintf "\nfor(%s %s ; %s) \n { %s \n }" initStmtString exprString
    incrStmtSubString bodyString

and writeWhileStatement condition body typemap =
let exprString = generateExpression condition in
let bodyString = writeStmtList body typemap in
sprintf "\nwhile(%s ) \n { %s \n }" exprString bodyString


and generateJavaProgram fileName prog =
  let statementString = generateMainFunction prog.tmainf prog.tmap in
  let decllist = prog.tdeclf in
  let funcDeclString = generateOtherFunctions decllist prog.tmap in
  let classBody = statementString^funcDeclString in
  let progString = sprintf "
  import java.io.File;
  import org.apache.pdfbox.pdmodel.PDDocument;
  import org.apache.pdfbox.pdmodel.PDPage;
  import org.apache.pdfbox.pdmodel.PageLayout;
  import org.apache.pdfbox.pdmodel.font.PDFont;
  import org.apache.pdfbox.pdmodel.PDPageContentStream;
  import org.apache.pdfbox.pdmodel.font.PDType1Font;
  import java.util.ArrayList;
  import java.util.Arrays;
```

```
  import java.util.List;
  import java.util.HashMap;
  import java.util.Map;
  public class %s
  {
    %s
  }
" fileName classBody in
  writeJavaProgramToFile fileName progString;
  progString
```

```
and generateFunction (b : Sast.tfunc_decl) typemap =
let name = b.name in
let returnType = b.rtype in
let returnTypeString = getJavaType returnType typemap in
let formalStatementList = b.tformals in
let functionBody = b.tbody in
let formalsListString = generateFormalsList formalStatementList typemap in
let functionBodyString = writeStmtList functionBody typemap in
sprintf"\npublic static %s %s ( %s ) throws Exception \n{ \n%s \n}"
    returnTypeString name formalsListString functionBodyString


and generateFormal formal typemap =
match formal with
    | TVdecl(id,t) ->
              (let name = match id with
                          | IdTest(n) -> n in
                          let jtype = getJavaType t typemap in sprintf "%s %s"
                              jtype name )
    | TListDecl(id,t) ->
               (let name = match id with
```

```
                        | IdTest(n) -> n in
                        let jtype = getJavaType t typemap in sprintf "%s %s"
                            jtype name )
    | TMapDecl(id,t) ->
                (let name = match id with
                        | IdTest(n) -> n in
                        let jtype = getJavaType t typemap in sprintf "%s %s"
                            jtype name )
    | TObjectCreate(id,t,exprList) ->
                (let name = match id with
                        | IdTest(n) -> n in
                        let jtype = getJavaType t typemap in sprintf "%s %s"
                            jtype name )
    | _ -> failwith "What formals do you want bruh?"



and generateFormalsList formalStatementList typemap =
let outStr = List.fold_left (fun a b -> a ^ (generateFormal b typemap)^ ",") ""
    formalStatementList in
let len = String.length outStr in
if len > 0
then
  let arg = String.sub outStr 0 ((String.length outStr) - 1) in
  sprintf "%s" arg
else sprintf ""


and generateOtherFunctions functionList typemap =
let outStr = List.fold_left (fun a b -> a ^ (generateFunction b typemap)) ""
    functionList in
sprintf "%s" outStr


and writeMainFunction stmtList typemap =
let mainBody = writeStmtList stmtList typemap in
sprintf " public static void main(String[] args) throws Exception
```

```
    {
      %s
    }  " mainBody



  and generateMainFunction prog typemap =
  let mainFunctionBody = writeMainFunction prog.tbody typemap in
  sprintf "%s" mainFunctionBody
```

JAVA CODE

Util.java

```
import java.io.File;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.PDPage;
import org.apache.pdfbox.pdmodel.font.PDFont;
import org.apache.pdfbox.pdmodel.PDPageContentStream;
import org.apache.pdfbox.pdmodel.PDPageTree;
import org.apache.pdfbox.pdmodel.font.PDType1Font;
import org.apache.pdfbox.pdmodel.graphics.image.PDImageXObject;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import com.giaybac.traprange.PDFTableExtractor;
import com.giaybac.traprange.entity.Table;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.*;



import org.jfree.chart.ChartFactory;
```

```
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;

import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.data.general.DefaultPieDataset;
import org.apache.pdfbox.text.PDFTextStripper;

public class Util{

  public static PDDocument addPageToPDF(PDDocument doc,PDPage page) throws
      Exception{

    doc.addPage(page);

    return doc;
  }

  public static Tuple addLineToTuple(Tuple tuple,Line line)throws Exception{
    // Start a new content stream which will "hold" the to be created content

    int pwidth = line.getWidth();
    int start = 0;
    int end = 0;
    PDPageContentStream contentStream = new
        PDPageContentStream(tuple.getDocument(), tuple.getPage(), true, true);

    PDFont font = getFontFromString(line.getFont());
    int[] array = possibleWrapPoints(line.getText());


    line.setText(line.getText().replaceAll("\n", " " ));
    line.setText(line.getText().replaceAll("\t", " "));
    line.setText(line.getText().replaceAll("\r", " "));
    // Define a text content stream using the selected font, moving the cursor
        and drawing the text "Hello World"
```

```
  for (int i =0; i < array.length; i++ ) {
     float width =
         font.getStringWidth(line.getText().substring(start,array[i])) / 1000 *
         line.getFontSize();
       if ( start <= end && width > pwidth ) {

          contentStream.beginText();
          contentStream.setFont(font, line.getFontSize());
          contentStream.moveTextPositionByAmount( line.getXcod(),
              line.getYcod() );
          end = array[i];
          contentStream.drawString(line.getText().substring(start,end));
          contentStream.endText();
          line.setRemainingText(line.getText().substring(end,line.getText().length()));

           // Make sure that the content stream is closed:
           contentStream.close();
   break;
       }
       else if(i == array.length - 1)
       {
          contentStream.beginText();
          contentStream.setFont(font, line.getFontSize());
          contentStream.moveTextPositionByAmount( line.getXcod(),
              line.getYcod() );
          end = array[i];
          contentStream.drawString(line.getText().substring(start,end));
          contentStream.endText();
          line.setRemainingText("");
           // Make sure that the content stream is closed:
           contentStream.close();

       }
   }
   return tuple;
}
```

```
public static Tuple addImageToTuple(Tuple tuple, Image image) throws Exception
{
 PDImageXObject pdIMage =
     PDImageXObject.createFromFileByContent(image.getFile(),
     tuple.getDocument());
 PDPageContentStream contentStream = new
     PDPageContentStream(tuple.getDocument(), tuple.getPage(), true, true);
 contentStream.drawImage(pdIMage, image.getXCood(), image.getYCood(),
     image.getWidth(), image.getHeight());
 contentStream.close();

 return tuple;

}


 public static List<List<String>> readTable(String location, List<Integer>
     pageNumbers){

        PDFTableExtractor extractor = (new
            PDFTableExtractor()).setSource(location);

        List<List<String>> lists = new ArrayList<List<String>>();
        for(Integer j : pageNumbers)
        {
          extractor.addPage(j-1);

          List<Table> extract = extractor.extract();
          String csv = extract.get(0).toString();

          String[] line = csv.trim().split("\n");
          for(int i = 0; i < line.length; i++)
          {
            String[] splits = line[i].split(";");
            List<String> asList = Arrays.asList(splits);
```

```
            lists.add(asList);


          }

        }
        boolean flag = false;
        List<Integer> listsToBeRemoved = new ArrayList<Integer>();


        int i = 0;
        for(List<String> list: lists)
        {
          for(String str : list)
          {
            if(str.trim().length() == 0)
              flag = true;
              break;
          }

          if(flag)
            listsToBeRemoved.add(i);
          flag = false;
          i++;
        }

        for(i = listsToBeRemoved.size()-1; i >=0; i--)
        {
          lists.remove(i);
        }
        return lists;
    }


    public static String readFile(String location) throws Exception{
```

```
  BufferedReader br = new BufferedReader(new FileReader(location));
  try {
      StringBuilder sb = new StringBuilder();
      String line = br.readLine();

      while (line != null) {
          sb.append(line);
          sb.append(" ");
          line = br.readLine();
      }
      return sb.toString();
  } finally {
      br.close();
  }

}


  static int[] possibleWrapPoints(String text) {
    String[] split = text.split("(.*?)");
    int[] ret = new int[split.length];
    ret[0] = split[0].length();
    for ( int i = 1 ; i < split.length ; i++ )
        ret[i] = ret[i-1] + split[i].length();
    return ret;
}


public static Image drawPieChart(List<List<String>> data, Map<String, String>
    attributes) {
    try {

       DefaultPieDataset pieDataset = new DefaultPieDataset();
       int width = 400; /* Width of the image */
       int height = 300; /* Height of the image */
       String chartTitle = "Chart Title";
       String imageName = "imageName.png";
```

```
int xcod = 100;
int ycod = 700;

if (attributes.containsKey("ChartTitle")) {
   chartTitle = attributes.get("ChartTitle");
}

if (attributes.containsKey("Height")) {
   height = Integer.parseInt(attributes.get("Height"));
}

if (attributes.containsKey("Width")) {
   width = Integer.parseInt(attributes.get("Width"));
}

if (attributes.containsKey("ImageName")) {
   imageName = attributes.get("ImageName") + ".png";
}

if (attributes.containsKey("X")) {
   xcod = Integer.parseInt(attributes.get("X"));
}

if (attributes.containsKey("Y")) {
   ycod = Integer.parseInt(attributes.get("Y"));
}

List<String> subList;
for (int i = 0; i < data.size(); i++) {
   subList = data.get(i);
   pieDataset.setValue(subList.get(0),
       Integer.parseInt(subList.get(1).trim()));
}
// Create the chart
JFreeChart chart = ChartFactory.createPieChart3D(chartTitle,
    pieDataset, true, true, true);
```

```
        ChartUtilities.saveChartAsPNG(new File(imageName), chart, width,
            height);

        Image image = new Image(new File(imageName), width, height, xcod,
            ycod);

        return image;

      }catch(Exception e)
    {
      e.printStackTrace();
      return null;
    }
}



public static String readTextFromPdf(String location, List<Integer>
    pageNumbers) throws Exception{

    File file = new File(location);
    StringBuffer buff = new StringBuffer();

    for(Integer i : pageNumbers)
    {
        PDDocument document = PDDocument.load(file);
        PDFTextStripper pdfStripper = new PDFTextStripper();
        pdfStripper.setStartPage(i);
        pdfStripper.setEndPage(i);
        buff.append(pdfStripper.getText(document));
     }

     return buff.toString();
}

  public static Image drawBarChart(List<List<String>> data, Map<String,
      String> attributes) {
```

```
try {

   DefaultCategoryDataset dataset = new DefaultCategoryDataset();
   int width = 400; /* Width of the image */
   int height = 300; /* Height of the image */
   String chartTitle = "Chart Title";
   String imageName = "imageName.png";
   String xaxis = "X-Axis";
   String yaxis = "Y-Axis";
   int xcod = 100;
   int ycod = 700;

   if (attributes.containsKey("ChartTitle")) {
      chartTitle = attributes.get("ChartTitle");
   }

   if (attributes.containsKey("Height")) {
      height = Integer.parseInt(attributes.get("Height"));
   }

   if (attributes.containsKey("Width")) {
      width = Integer.parseInt(attributes.get("Width"));
   }

   if (attributes.containsKey("ImageName")) {
      imageName = attributes.get("ImageName") + ".png";
   }

   if (attributes.containsKey("X")) {
      xcod = Integer.parseInt(attributes.get("X"));
   }

   if (attributes.containsKey("Y")) {
      ycod = Integer.parseInt(attributes.get("Y"));
   }
```

```
        for (int i = 0; i < data.size(); i++) {
           List<String> subList = data.get(i);
           dataset.addValue(Double.parseDouble(subList.get(1)), xaxis,
               subList.get(0));

        }

        JFreeChart barChart = ChartFactory.createBarChart(chartTitle, xaxis,
            yaxis, dataset,
              PlotOrientation.VERTICAL, true, true, false);

        ChartUtilities.saveChartAsPNG(new File(imageName), barChart, width,
            height);

        Image image = new Image(new File(imageName), height, width, xcod, ycod
            );

        return image;
    }catch (Exception e) {
    e.printStackTrace();
        return null;
    }
 }


public static PDFont getFontFromString(String font){

  switch(font){

    case "TIME_ROMAN" : return PDType1Font.TIMES_ROMAN;

    case "COURIER": return PDType1Font.COURIER;

    case "COURIER_BOLD" : return PDType1Font.COURIER_BOLD;

    case "COURIER_BOLD_OBLIQUE" : return PDType1Font.COURIER_BOLD_OBLIQUE;
```

```
    case "COURIER_OBLIQUE" : return PDType1Font.COURIER_OBLIQUE;

    case "HELVETICA" : return PDType1Font.HELVETICA;

    case "HELVETICA_BOLD" : return PDType1Font.HELVETICA_BOLD;

    case "HELVETICA_BOLD_OBLIQUE" : return PDType1Font.HELVETICA_BOLD_OBLIQUE;

    case "HELVETICA_OBLIQUE" : return PDType1Font.HELVETICA_OBLIQUE;

    case "SYMBOL": return PDType1Font.SYMBOL;

    case "TIMES_BOLD" : return PDType1Font.TIMES_BOLD;

    case "TIMES_BOLD_ITALIC" : return PDType1Font.TIMES_BOLD_ITALIC;

    case "TIMES_ITALIC" : return PDType1Font.TIMES_ITALIC;

    case "ZAPF_DINGBATS" : return PDType1Font.ZAPF_DINGBATS;

    default : return PDType1Font.TIMES_ROMAN;

  }
 }


public static List<PDPage> getPages(PDDocument document) throws Exception{

    PDPageTree pages = document.getPages();
    List<PDPage> listOfPages = new ArrayList<PDPage>();
    Iterator<PDPage> iterator = pages.iterator();
    while(iterator.hasNext())
    {
      PDPage next = iterator.next();
      listOfPages.add(next);

    }
```

```
    return listOfPages;
}

public static List<PDDocument> splitPdf(List<Integer> splits,PDDocument
   document) throws Exception{


  List<PDPage> listOfPages = Util.getPages(document);
  List<PDDocument> listOfDocuments = new ArrayList<PDDocument>();
  PDDocument newDocument = null;
    int lastSplit = 0;
    int j = 0;
    for(Integer i : splits)
    {
       newDocument = new PDDocument();
      for(j =lastSplit; j < i; j++ )
      {
        PDPage page = listOfPages.get(j);
        newDocument.addPage(page);
      }
      lastSplit = j;
      listOfDocuments.add(newDocument);
    }

     newDocument = new PDDocument();
    for(int k = j ; k < listOfPages.size(); k++)
    {
      newDocument.addPage(listOfPages.get(k));
    }

    listOfDocuments.add(newDocument);

    return listOfDocuments;
  }


public static PDDocument loadPdf(String filename) throws Exception{
```

```
    File file = new File(filename);
    PDDocument document = PDDocument.load(file);
    return document;
}

public static String substr(String s, int startIndex, int endIndex) throws
    Exception{

    return s.substring(startIndex,endIndex);
}


}
```

Line.java

```
public class Line {

   private String text;
   private String font;
   private int xcod;
   private int ycod;
   private int fontSize;
   private int width;
   private String remainingText;

   public String getText() {
      return text;
   }
   public void setText(String text) {
      this.text = text;
   }
   public String getFont() {
      return font;
   }
```

```
public void setFont(String font) {
   this.font = font;
}
public int getXcod() {
   return xcod;
}
public void setXcod(int xcod) {
   this.xcod = xcod;
}
public int getYcod() {
   return ycod;
}
public void setYcod(int ycod) {
   this.ycod = ycod;
}
public int getFontSize() {
   return fontSize;
}
public void setFontSize(int fontSize) {
   this.fontSize = fontSize;
}

public String getRemainingText() {
   return remainingText;
}

public void setRemainingText(String remainingText) {
   this.remainingText = remainingText;
}
public void setWidth(int width) {
   this.width = width;
}
public int getWidth() {
   return width;
}
```

```
}
```

Tuple.java

```
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.PDPage;

public class Tuple {

   private PDDocument document;
   private PDPage page;

  public Tuple(PDDocument document, PDPage page) {
      super();
      this.document = document;
      this.page = page;
   }

   public PDDocument getDocument() {
      return document;
   }
   public void setDocument(PDDocument document) {
      this.document = document;
   }
   public PDPage getPage() {
      return page;
   }
   public void setPage(PDPage page) {
      this.page = page;
   }



}
```

Image.java

```
import java.io.File;

public class Image {

   private File file;
   private int height;
   private int width;
   private int XCood;
   private int YCood;


   public Image(File file, int height, int width, int xcood, int ycood)
   {
      this.file = file;
      this.height = height;
      this.width = width;
      this.XCood = xcood;
      this.YCood = ycood;
   }

   public File getFile() {
      return file;
   }
   public void setFile(File file) {
      this.file = file;
   }
   public int getHeight() {
      return height;
   }
   public void setHeight(int height) {
      this.height = height;
   }
   public int getWidth() {
      return width;
   }
```

```
   public void setWidth(int width) {
      this.width = width;
   }
   public int getXCood() {
      return XCood;
   }
   public void setXCood(int xCood) {
      XCood = xCood;
   }
   public int getYCood() {
      return YCood;
   }
   public void setYCood(int yCood) {
      YCood = yCood;
   }

}
```

PAL CODE

Standard Library - stdlib.pal

```
main()
{

}

write_paragraph(tupleVar : tuple, stringVar : string, startMargin : int,
    startHeight : int, fontSize : int, fontType : string, endHeight : int,
    width : int) : string {

  lengthOfString : int = length(stringVar);

  while(startHeight > endHeight)
  {
```

```
  lineVar : line(stringVar,fontType,fontSize,startMargin,startHeight,width);

  tupleVar = tupleVar . lineVar;

  stringVar = lineVar|_;

  startHeight = startHeight - 20;

  lengthOfString = length(stringVar);

  if(lengthOfString == 0)
  {
    break;
  }

  }
  return stringVar;
  }



write_two_column_layout(tupleVar:tuple, stringVar:string, fontType:string,
    fontSize:int):string{

startX:int = 20;

width:int = 230;

startY: int = 700;

endHeight:int = 50;

  stringVar = write_paragraph(tupleVar, stringVar, startX, startY, fontSize,
      fontType, endHeight,width);

  if(length(stringVar) > 0){
    startX = 315;
```

```
    #width = width + width;
    stringVar = write_paragraph(tupleVar, stringVar, startX, startY, fontSize,
        fontType, endHeight,width);
  }

  return stringVar;
}


write_three_column_layout(tupleVar:tuple, stringVar:string, fontType:string,
    fontSize:int):string{

startX:int = 20;

width:int = 150;

startY: int = 700;

endHeight:int = 50;

  stringVar = write_paragraph(tupleVar, stringVar, startX, startY, fontSize,
      fontType, endHeight,width);

  if(length(stringVar) > 0){
    startX = 200;
    #width = width + width;
    stringVar = write_paragraph(tupleVar, stringVar, startX, startY, fontSize,
        fontType, endHeight,width);
  }

  if(length(stringVar) > 0){
    startX = 380;
    #width = width + width;
    stringVar = write_paragraph(tupleVar, stringVar, startX, startY, fontSize,
        fontType, endHeight,width);
  }
```

```
  return stringVar;
}


write_4grid_layout(tupleVar:tuple, stringVar:string, fontType:string,
    fontSize:int) : string
{

    startX:int = 20;

    width:int = 240;

    startY: int = 600;

    endHeight:int = 440;

   stringVar = write_paragraph(tupleVar, stringVar, startX, startY, fontSize,
       fontType, endHeight,width);

   if(length(stringVar) > 0){
    startX = 320;
    stringVar = write_paragraph(tupleVar, stringVar, startX, startY, fontSize,
        fontType, endHeight,width);
  }


  endHeight = 60;

  startY = 290;

   if(length(stringVar) > 0){
    startX = 20;
    stringVar = write_paragraph(tupleVar, stringVar, startX, startY, fontSize,
        fontType, endHeight,width);
  }

   if(length(stringVar) > 0){
```

```
    startX = 320;
    stringVar = write_paragraph(tupleVar, stringVar, startX, startY, fontSize,
        fontType, endHeight,width);
  }

  return stringVar;

}

write_pages(stringVar : string, fontSize : int, fontType : string,
    layoutType:string) : pdf{
  pdfVar:pdf;
  while(length(stringVar) > 0){
    pageVar:page;
    pdfVar = pdfVar . pageVar;
    tupleVar:tuple(pdfVar,pageVar);

    if(layoutType == "TWO_COLUMN"){
      stringVar = write_two_column_layout(tupleVar,stringVar,fontType,fontSize);
    } elif(layoutType == "THREE_COLUMN"){
      stringVar =
          write_three_column_layout(tupleVar,stringVar,fontType,fontSize);
    } else{
      stringVar = write_4grid_layout(tupleVar,stringVar,fontType,fontSize);
    }

  }
  return pdfVar;
}
```

Demo Program - finaldemo.pal

```
import ("stdlib.pal");

main()
{
```

```
pdfVar : pdf;


#read text from first two pages of raw pdf and render in column layout on
    output pdf
textpagenumbers : list int(1,2);

filepath : string = "input.pdf";

content : string = readtextfrompdf(filepath,textpagenumbers);

pdfVar = write_pages(content, 12, "COURIER_OBLIQUE", "TWO_COLUMN");



#read table from raw pdf and convert to bar chart
table : list list string;
tablepagenumbers : list int(3);

table = readtable(filepath,tablepagenumbers);

properties : map string,string;

properties += "ChartTitle","PLT Assignment 3 Statistics";
properties += "Height","250";
properties += "Width","300";
properties += "X","150";
properties += "Y","500";

chartimage : image;

chartimage = drawbarchart(table,properties);
pageVar : page;

#Add page to pdf
```

```
pdfVar = pdfVar . pageVar;

tupleVar : tuple(pdfVar, pageVar);

tupleVar = tupleVar . chartimage;


#read table from raw pdf and convert to pie chart chart

properties += "X","200";
properties += "Y","100";

chartimage = drawpiechart(table,properties);

tupleVar = tupleVar . chartimage;


#read page 3 form raw pdf and write exactly one page in 3 column layout on
    output pdf

pagenumbers : list int(4);
content = readtextfrompdf(filepath,pagenumbers);

pageforthreecolumnlayout : page;
pdfVar = pdfVar . pageforthreecolumnlayout;
texttuple : tuple(pdfVar, pageforthreecolumnlayout);

content = write_three_column_layout(texttuple, content, "HELVETICA", 12);

renderpdf(pdfVar,"finaldemooutput.pdf");


#load rendered pdf and split at page numbers 4 and 5. Expected: All two
    column layout pages
#go in the first split, the image page in the 2nd split and the 3 column
    layout in the
#third split
```

```
  pdfVar = loadpdf("finaldemooutput.pdf");

  pdfs : list pdf;

  splitnumbers : list int(4,5);

  pdfs = split(pdfVar, splitnumbers);

  i: int;

  lengthoflist : int;
  lengthoflist = length(pdfs);

  for(i = 0 ; i < lengthoflist; i = i + 1)
  {
    renderpdf(pdfs[i],"split"+i+".pdf");
  }


}
```

Select Test Cases

test-arith1.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   intVar : int;
   intVar = 39 + 3;
   tupleVar : tuple(pdfVar, pageVar);
   linevar : line("ABCD", "Times_New_Roman" , 12 , 100, 700, 600);
   tupleVar = tupleVar . linevar;
   renderpdf(pdfVar, "test-arith1.pdf");
```

```
}
```

test-arith2.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   intVar : int;
   intVar = 1 + 2 * 3 + 4;
   tupleVar : tuple(pdfVar, pageVar);
   linevar : line("intVar", "Times_New_Roman" , 12 , 100, 700,300);
   tupleVar = tupleVar . linevar;
   renderpdf(pdfVar, "test-arith2.pdf");
}
```

test-bool1.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   stringVar : string;
   stringVar = "Failure";

   tupleVar : tuple (pdfVar, pageVar);
   linevar : line ("intVar", "Times_New_Roman" , 12 , 100, 700, 500);

   if(true && false)
   {
      stringVar = "Success";
   }

   if(true || false)
```

```
   {
      stringVar = "Failure";
   }
   renderpdf(pdfVar, "test-bool1.pdf");
}
```

test-bool2.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   stringVar : string;
   stringVar = "Failure";



   if (true || true)
   {
      stringVar = "Success";
   }

   if (true || false)
   {
      stringVar = "Success";
   }
   if (false || false)
   {
      stringVar = "Failure";
   }

   tupleVar : tuple(pdfVar, pageVar);
   linevar : line(stringVar, "Times_New_Roman" , 12 , 100, 700,500);

   renderpdf(pdfVar, "test-bool2.pdf");
}
```

test-bool3.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   stringVar : string;
   stringVar = "Failure";

   if (!true == true)
   {
      stringVar = "Failure";
   }
   if (!false == true)
   {
      stringVar = "Success";
   }

   tupleVar : tuple(pdfVar, pageVar);
   linevar : line(stringVar, "Times_New_Roman" , 12 , 100, 700, 500);

   renderpdf(pdfVar, "test-bool3.pdf");
}
```

test-for1.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   stringVar : string;
   stringVar = "Failure";
```

```
   intVar : int;

   for(intVar = 0; intVar < 5 ; intVar = intVar + 1)
   {
      stringVar = stringVar + "Success";
   }

   tupleVar : tuple(pdfVar, pageVar);
   linevar : line(stringVar, "Times_New_Roman" , 12 , 100, 700, 500);

   tupleVar = tupleVar . linevar;
   renderpdf(pdfVar, "test-for1.pdf");
}
```

test-func1.pal

```
main()
{
   intVar : int;
   intVar = sum (5, 6);
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   tupleVar : tuple (pdfVar, pageVar);
   linevar : line("intVar", "Times_New_Roman" , 12 , 100, 700, 500);
   renderpdf(pdfVar, "test-func1.pdf");
}

sum(a : int, b : int) : int
{
   return a + b;
}
```

test-if1.pal

```
main()
```

```
{
   pdfVar : pdf;
   pageVar : page;

   pdfVar = pdfVar.pageVar;

   lineVar42 : line("42", "HELVETICA_BOLD_OBLIQUE",12,100,700,500);
   lineVar17 : line("17", "HELVETICA_BOLD_OBLIQUE",12,100,500,500);


   tupleVar : tuple(pdfVar,pageVar);

   /*if(true == true)
   {
   tupleVar = tupleVar.lineVar42;
   }*/

   tupleVar = tupleVar.lineVar17;

   renderpdf(pdfVar,"test-if1.pdf");

}
```

test-if2.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;

   pdfVar = pdfVar.pageVar;

   lineVar42 : line("42", "Times_New_Roman",12,100,700,500);
   lineVar17 : line("17", "Times_New_Roman",12,100,500, 500);
   lineVar8 : line("8", "Times_New_Roman",12,100,300, 500);

   tupleVar : tuple(pdfVar,pageVar);
```

```
   if(true == true )
   {
   tupleVar = tupleVar.lineVar42;
   }
   else
   {
   tupleVar = tupleVar.lineVar17;
   }

   tupleVar = tupleVar.lineVar8;
   renderpdf(pdfVar,"test-if2.pdf");
}
```

test-if3.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;

   pdfVar = pdfVar.pageVar;

   lineVar42 : line("42", "Times_New_Roman",12,100,700, 500);
   lineVar17 : line("17", "Times_New_Roman",12,100,500, 500);

   tupleVar : tuple(pdfVar,pageVar);

   if(false == false)
   {
      tupleVar = tupleVar.lineVar42;
   }

   tupleVar = tupleVar.lineVar17;
    renderpdf(pdfVar,"test-if3.pdf");
}
```

test-if4.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;

   pdfVar = pdfVar.pageVar;

   lineVar42 : line("42", "Times_New_Roman",12,100,700, 500);
   lineVar17 : line("17", "Times_New_Roman",12,100,500, 500);
   lineVar8 : line("8", "Times_New_Roman",12,100,300, 500);

   tupleVar : tuple(pdfVar,pageVar);

   if(false == false)
   {
      tupleVar = tupleVar.lineVar42;
   }
   else
   {
      tupleVar = tupleVar.lineVar17;
   }
   tupleVar = tupleVar.lineVar8;

   renderpdf(pdfVar,"test-if4.pdf");

}
```

test-listadd1.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
    tupleVar : tuple(pdfVar,pageVar);
```

```
    listVar:list string;

    listVar += "Hello";

    lineVar : line(listVar[0], "HELVETICA_BOLD_OBLIQUE",12,100,700,500);
    tupleVar = tupleVar.lineVar;

   renderpdf(pdfVar,"test-listadd1.pdf");

}
```

test-listadd21.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
    tupleVar : tuple(pdfVar,pageVar);
    listVar:list string;

    listVar += "Hello";
    listVar += "This";
    listVar += "Is";
    listVar += "Our";
    listVar += "PLT";
    listVar += "Project";

   lineVar : line(listVar[4], "HELVETICA_BOLD_OBLIQUE",12,100,700,500);
   tupleVar = tupleVar.lineVar;

   renderpdf(pdfVar,"test-listadd2.pdf");

}
```

test-listinit.pal

```
main()
{
   listvar : list int(1,2,3,4);
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   intVar : int;
   intVar = 39 + 3;
   tupleVar : tuple(pdfVar, pageVar);
   linevar : line("ABCD", "Times_New_Roman" , 12 , 100, 700, 600);
   tupleVar = tupleVar . linevar;
   renderpdf(pdfVar, "test-listinit1.pdf");
}
```

test-listremove1.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
    tupleVar : tuple(pdfVar,pageVar);
    listVar:list string;

    listVar += "Hello";
    listVar += "World";
    listVar -= [1];

    lineVar : line(listVar[0], "HELVETICA_BOLD_OBLIQUE",12,100,700,500);
    tupleVar = tupleVar.lineVar;

    renderpdf(pdfVar,"test-listremove1.pdf");
}
```

test-mapadd1.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
    tupleVar : tuple(pdfVar,pageVar);
    mapVar:map string,string;

    mapVar += "Hello","World";

    lineVar : line(mapVar:= "Hello", "HELVETICA_BOLD_OBLIQUE",12,100,700,500);
   tupleVar = tupleVar.lineVar;

   renderpdf(pdfVar,"test-mapadd1.pdf");
}
```

test-mapfind1.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
    tupleVar : tuple(pdfVar,pageVar);
    mapVar:map string,string;

    mapVar += "Hello","World";
    mapVar += "PLT","Project";
    mapVar += "COMS","4115";
    stringVar:string = "COMS";

    lineVar : line(mapVar:= stringVar, "HELVETICA_BOLD_OBLIQUE",12,100,700,500);
   tupleVar = tupleVar.lineVar;

   renderpdf(pdfVar,"test-mapfind1.pdf");
```

```
}
```

test-mapremove1.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
    tupleVar : tuple(pdfVar,pageVar);
    mapVar:map string,string;

    mapVar += "Hello","World";
    mapVar += "1","2";

    mapVar -= "1";

    lineVar : line(mapVar:= "Hello", "HELVETICA_BOLD_OBLIQUE",12,100,700,500);
   tupleVar = tupleVar.lineVar;

   renderpdf(pdfVar,"test-mapremove1.pdf");
}
```

test-substr1.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   stringVar : string;
   stringVar = "Success";
   stringVar = substr(stringVar, 1, 3);

   tupleVar : tuple(pdfVar, pageVar);
   linevar : line(stringVar, "Times_New_Roman" , 12 , 100, 700, 500);
```

```
   tupleVar = tupleVar . linevar;
   renderpdf(pdfVar, "test-stringsubstr1.pdf");

}
```

test-neg1.pal

```
main()
{
   pdfVar : pdf;
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   intVar : int;
   intVar = -9 + -8;
   stringVar : string;
   if(intVar == -17)
   {
      stringVar = "Pass";
   }
   else
   {
      stringVar = "Fail";
   }

   tupleVar : tuple(pdfVar, pageVar);
   linevar : line(stringVar, "Times_New_Roman" , 12 , 100, 700, 600);
   tupleVar = tupleVar . linevar;
   renderpdf(pdfVar, "test-neg1.pdf");

}
```

test-while1.pal

```
main()
{
   pdfVar : pdf;
```

```
   pageVar : page;
   pdfVar = pdfVar.pageVar;
   intVar : int = 1;
    stringVar : string = "Done";
    tupleVar : tuple(pdfVar,pageVar);

    while(intVar < 5){
      intVar = intVar + 1;

          if(intVar == 5){
            stringVar = "Done";
    }
  }

   lineVar : line(stringVar, "HELVETICA_BOLD_OBLIQUE",12,100,700,500);
   tupleVar = tupleVar.lineVar;
   renderpdf(pdfVar,"test-while1.pdf");

}
```

test-functioncall1.pal

```
main() {
    i : int;
    s : string;
    l : list tuple;
    ll : list list tuple;
    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    tuplevar : tuple(pdfvar,pagevar);

    i = 0;
    s = "Function Call Checked";
    l += tuplevar;
    ll += l;
```

```
    func(i,s,ll);

    renderpdf(pdfvar,"test-functioncall1.pdf");
}

func (i : int, s : string, l : list list tuple) : int{
    linevar : line(s,"TIME_NEW_ROMAN",12,10,700,650);
    l_ : list tuple;
    l_ = l[i];
    l_[0] = l_[0].linevar;
    return 1;
}
```

test-functiondeclaration1.pal

```
main() {
    i : int;
    s : string;
    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    t : tuple(pdfvar,pagevar);

    i = 0;
    s = "Function Declaration Checked";

    func(i,s,t);

    renderpdf(pdfvar,"test-functiondeclaration1.pdf");
}

func (i : int, s : string, t : tuple) : int{
    l : line(s,"TIME_NEW_ROMAN",12,10,700,650);
    t = t.l;
    return 1;
}
```

test-functionhierarchy1.pal

```
main() {
    i : int;
    s : string;
    l : list tuple;
    ll : list list tuple;
    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    tuplevar : tuple(pdfvar,pagevar);

    i = 0;
    s = "Function HIerarchy Checked";
    l += tuplevar;
    ll += l;

    func2(i,s,ll);

    renderpdf(pdfvar,"test-functionhierarchy1.pdf");
}

func0 (i : int) : int {
    return i;
}

func1 (i : int, s : string) : string {
    i = func0(i);
    s = s + i;
    return s;
}

func2 (i : int, s : string, l : list list tuple) : int{
    s = func1(i,s);
    linevar : line(s,"TIME_NEW_ROMAN",12,10,700,650);
    l_ : list tuple;
    l_ = l[i];
```

```
    l_[0] = l_[0].linevar;
    return 1;
}
```

test-import1.pal

```
import ("test-functiondeclaration1.pal");

main() {
    i : int;
    s : string;
    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    t : tuple(pdfvar,pagevar);

    i = 0;
    s = "Import Checked";

    func(i,s,t);

    renderpdf(pdfvar,"test-import1.pdf");
}
```

test-importnested1.pal

```
import ("test-import1.pal");

main() {
    i : int;
    s : string;
    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    t : tuple(pdfvar,pagevar);
```

```
    i = 0;
    s = "Import Nested Checked";

    func(i,s,t);

    renderpdf(pdfvar,"test-importnested1.pdf");
}
```

test-listhierarchy1.pal

```
main() {
    l1 : list int;
    l2 : list int;
    l3 : list int;

    ll : list list int;

    iter : int;

    for (iter = 1; iter <= 3; iter = iter + 1) {
        l1 += iter;
        l2 += iter + 3;
        l3 += iter + 6;
    }

    ll += l1;
    ll += l2;
    ll += l3;

    result : string = "Result: ";

    for (iter = 0; iter < 3; iter = iter + 1) {
        it : int;
        l : list int;
        l = ll[iter];
        for (it = 0; it < 3; it = it + 1) {
            result = result + l[it];
```

```
        }
    }

    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    tuplevar : tuple(pdfvar,pagevar);
    linevar : line(result, "TIME_NEW_ROMAN", 12, 10, 600, 650);
    tuplevar = tuplevar.linevar;
    renderpdf(pdfvar,"test-listhierarchy1.pdf");
}
```

test-listhierarchy2.pal

```
main() {
    i : int;
    l : list int;
    ll : list list int;
    lll : list list list int;

    l[0] = 1;
    l += 1;
    l[0] = i;
    l += i;
    i = l[0];
    ll[0] = l;
    ll += l;
    l = ll[0];
    lll[0] = ll;
    lll += ll;
    ll = lll[0];

    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    tuplevar : tuple(pdfvar,pagevar);
```

```
    linevar : line("List Hierarchy Checked", "TIME_NEW_ROMAN", 12, 10, 700,
        650);
    tuplevar = tuplevar.linevar;
    renderpdf(pdfvar,"test-listhierarchy2.pdf");
}
```

test-listmaphierarchy1.pal

```
main() {
    i : int;
    l : list int;
    ll : list list int;
    lll : list list list int;

    l[0] = 1;
    l += 1;
    l[0] = i;
    l += i;
    i = l[0];
    ll[0] = l;
    ll += l;
    l = ll[0];
    lll[0] = ll;
    lll += ll;
    ll = lll[0];

    m : map int,int;
    m += 1,1;
    m += i,i;
    m += l[0],i;
    m += i,l[0];
    m -= l[0];

    mm : map int, list int;
    mm += l[0],l;
    mm += l[0],ll[0];
```

```
    mmmm : map int, list list list int;
    mmmm += l[0],lll;
    lll = mmmm:=l[0];

    pdfvar: pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    tuplevar : tuple(pdfvar,pagevar);
    linevar : line("List Map Hierarchy Checked", "TIME_NEW_ROMAN", 12, 10, 700,
        650);
    tuplevar = tuplevar.linevar;
    renderpdf(pdfvar,"test-listmaphierarchy1.pdf");
}
```

test-mergelist1.pal

```
main() {
    l1 : list int;
    l1 += 1;
    l1 += 3;
    l1 += 5;

    l2 : list int;
    l2 += 2;
    l2 += 4;
    l2 += 6;

    mL : list int;

    iter : int = 0;
    result : string = "Result: ";

    for (iter = 0; iter < 3; iter = iter + 1) {
        mL += l1[iter];
        mL += l2[iter];
    }
```

```
    for (iter = 0; iter < 6; iter = iter + 1) {
        result = result + mL[iter];
    }

    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    tuplevar : tuple(pdfvar,pagevar);
    linevar : line(result, "TIME_NEW_ROMAN", 12, 10, 600, 650);
    tuplevar = tuplevar.linevar;
    renderpdf(pdfvar,"test-mergelist1.pdf");
}
```

test-operators1.pal

```
main() {
    i : int;
    i = 1 * 2 + 3 / 4 * 5 + 6 - 7 * 8 / 9 * 10 + 11 - 12 * 15 / 16 * 17 + 18 *
        19 - 20;

    result : string = "Result: ";
    result = result + i;
    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    tuplevar : tuple(pdfvar,pagevar);
    linevar : line(result, "TIME_NEW_ROMAN", 12, 10, 700, 650);
    tuplevar = tuplevar.linevar;
    renderpdf(pdfvar,"test-operators1.pdf");
}
```

test-operators2.pal

```
main() {
    i : int;
```

```
    i = ((1 * (2 + 3) / 4 * 5 + 6 - 7) * (8 / 9 * 10 + (11 - 12) * 15 / 16 * 17
        + 18) * 19 - 20);

    result : string = "Result: ";
    result = result + i;
    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    tuplevar : tuple(pdfvar,pagevar);
    linevar : line(result, "TIME_NEW_ROMAN", 12, 10, 700, 650);
    tuplevar = tuplevar.linevar;
    renderpdf(pdfvar,"test-operators2.pdf");
}
```

test-splitlist1.pal

```
main() {
    l : list int;
    l += 1;
    l += 2;
    l += 3;
    l += 4;
    l += 5;
    l += 6;

    l1 : list int;
    l2 : list int;

    iter : int = 0;

    for (iter = 5; iter >= 0; iter = iter - 2) {
        l1 += l[iter];
        l -= [l[iter]];
        l2 += l[iter-1];
        l -= [l[iter-1]];
    }
```

```
    r1 : string = "Result1: ";
    r2 : string = "Result2: ";

    for (iter = 0; iter < 3; iter = iter + 1) {
        r1 = r1 + l1[iter];
        r2 = r2 + l2[iter];
    }

    pdfvar : pdf;
    pagevar : page;
    pdfvar = pdfvar.pagevar;
    tuplevar : tuple(pdfvar,pagevar);
    linevar1 : line(r1, "TIME_NEW_ROMAN", 12, 10, 600, 650);
    linevar2 : line(r2, "TIME_NEW_ROMAN", 14, 10, 550, 650);
    tuplevar = tuplevar.linevar1;
    tuplevar = tuplevar.linevar2;
    renderpdf(pdfvar,"test-splitlist1.pdf");
}
```