TAPE

Tianhua Fang (tf2377)
Alexander Sato (as4628)
Priscilla Wang (pyw2102)
Edwin Chan (cc3919)

About TAPE

The Team

Tianhua Fang

Proposal, LRM, scanner, parser, codegen, makefile, demo, tests

Alexander Sato

Proposal, LRM, tests

Priscilla Wang

Proposal, LRM, scanner, parser, codegen, tests, demo, presentation

Edwin Chan

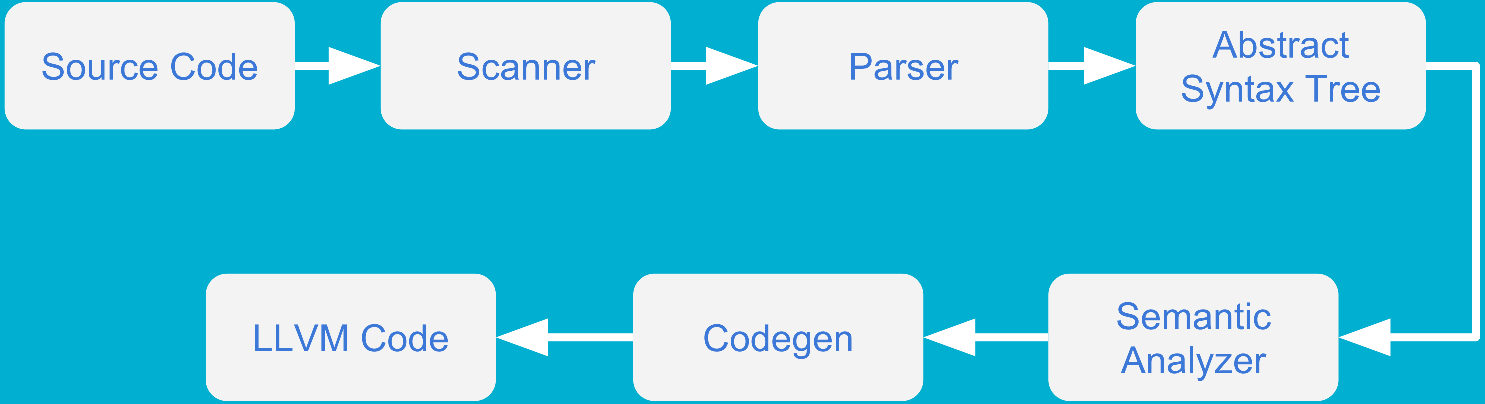
Proposal, LRM, scanner, parser, codegen, makefile, tests, demo, presentation

The goal:

To create a simple language that allows users to easily manipulate files.

Language Structure

Architecture



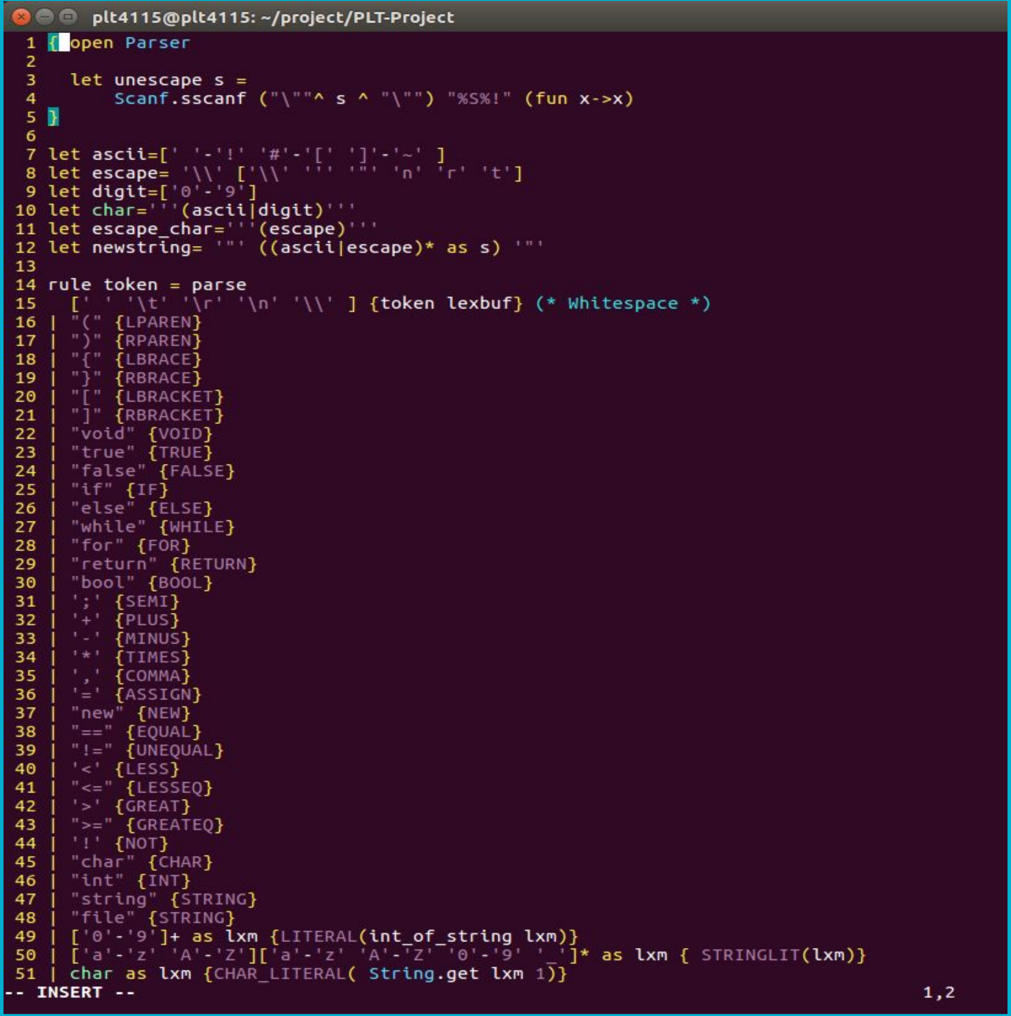
Scanner

Key Words that are reserved:

“If”, “else”, “while”, “for”, “return”

Reserved names:

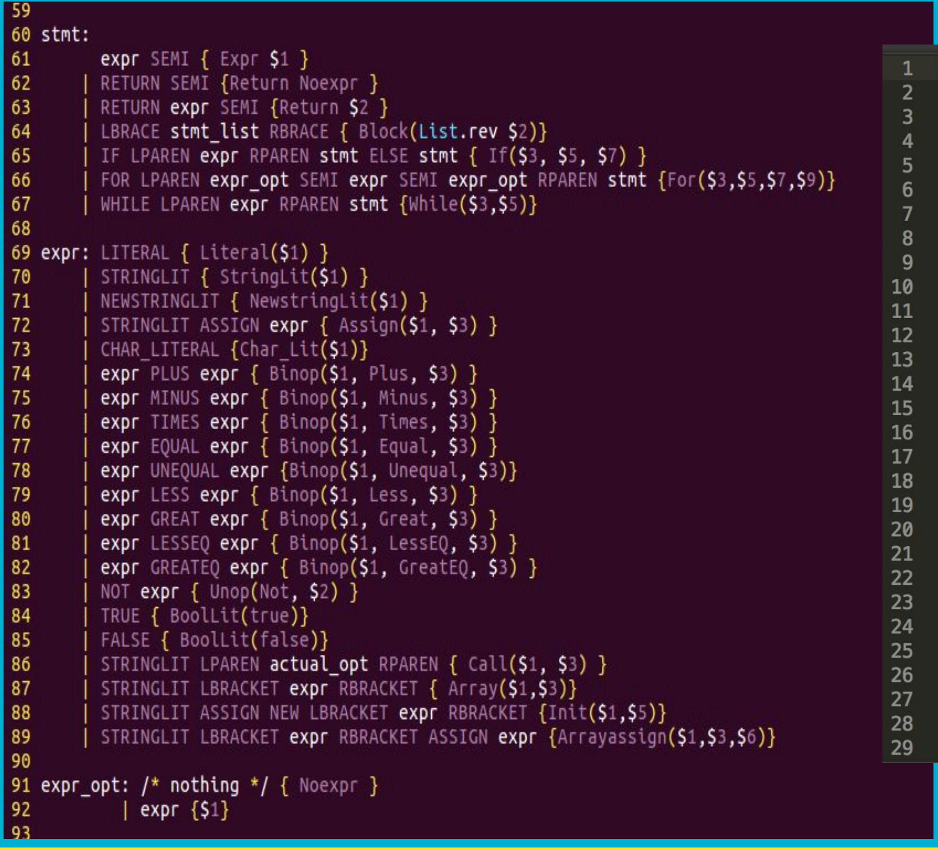
“Void”, “string”, “int”, “file”



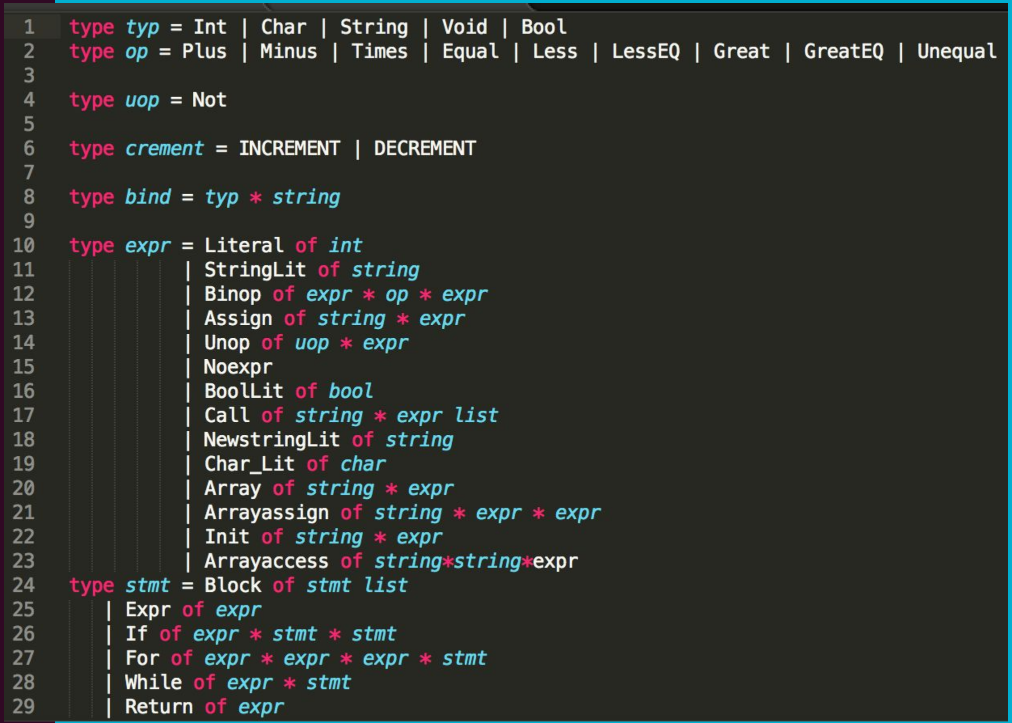
plt4115@plt4115: ~/project/PLT-Project

```
{ open Parser

  let unescape s =
      Scanf.sscanf ("\""^ s ^ "\"") "%S%!" (fun x->x)
}

let ascii=[' '-'!' '#'-'[' ']'-'~' ]
let escape= '\\' ['\\' ''' '"' 'n' 'r' 't']
let digit=['0'-'9']
let char='''(ascii|digit)'''
let escape_char='''(escape)'''
let newstring= '"' ((ascii|escape)* as s) '"'

rule token = parse
  [' ' '\t' '\r' '\n' '\\' ] {token lexbuf} (* Whitespace *)
| "(" {LPAREN}
| ")" {RPAREN}
| "{" {LBRACE}
| "}" {RBRACE}
| "[" {LBRACKET}
| "]" {RBRACKET}
| "void" {VOID}
| "true" {TRUE}
| "false" {FALSE}
| "if" {IF}
| "else" {ELSE}
| "while" {WHILE}
| "for" {FOR}
| "return" {RETURN}
| "bool" {BOOL}
| ';' {SEMI}
| '+' {PLUS}
| '-' {MINUS}
| '*' {TIMES}
| ',' {COMMA}
| '=' {ASSIGN}
| "new" {NEW}
| "==" {EQUAL}
| "!=" {UNEQUAL}
| '<' {LESS}
| "<=" {LESSEQ}
| '>' {GREAT}
| ">=" {GREATEQ}
| '!' {NOT}
| "char" {CHAR}
| "int" {INT}
| "string" {STRING}
| "file" {STRING}
| ['0'-'9']+ as lxm {LITERAL(int_of_string lxm)}
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm { STRINGLIT(lxm)}
| char as lxm {CHAR_LITERAL( String.get lxm 1)}
-- INSERT --                                                          1,2
```

Parser & ast

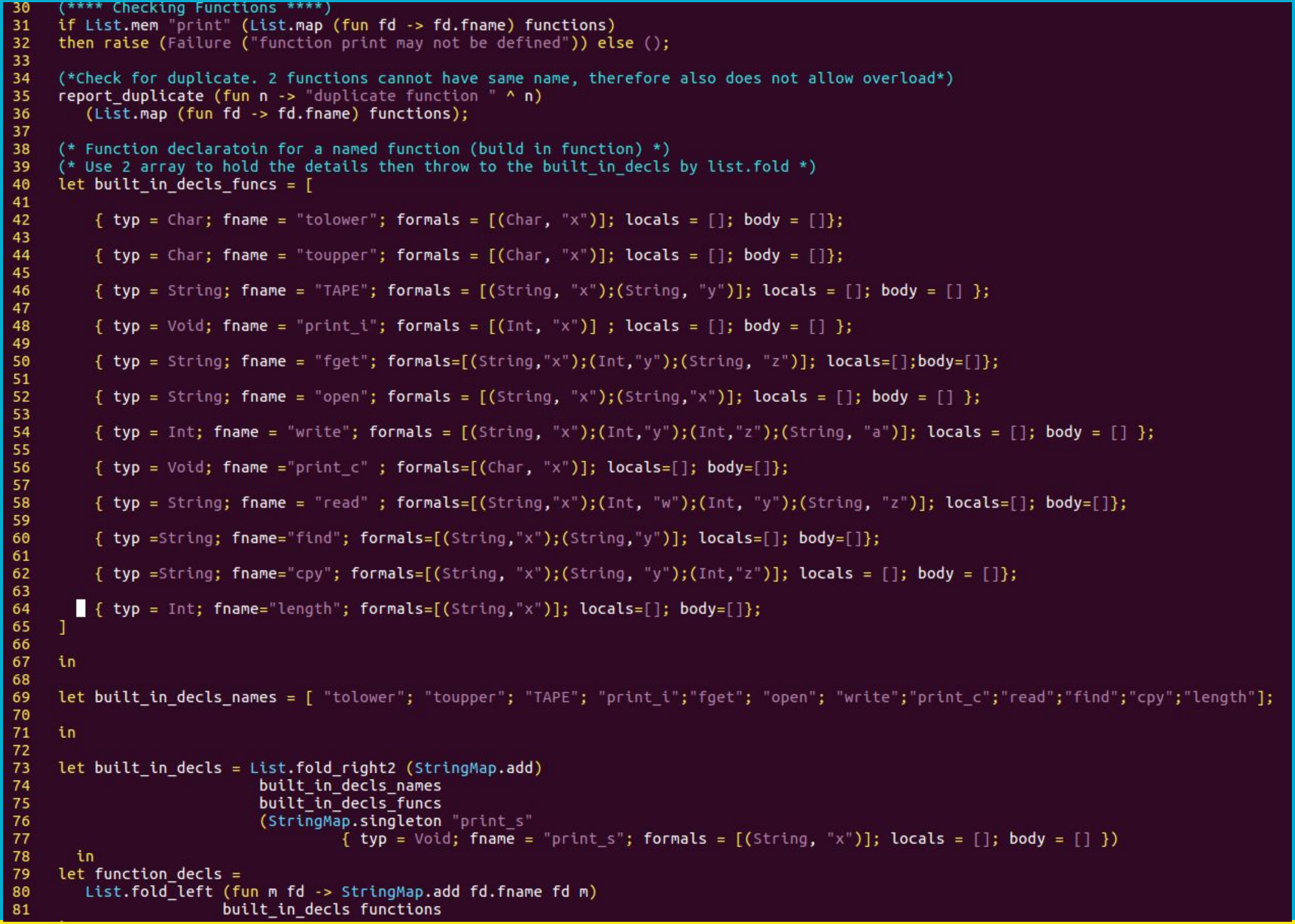


```

stmt:
        expr SEMI { Expr $1 }
      | RETURN SEMI {Return Noexpr }
      | RETURN expr SEMI {Return $2 }
      | LBRACE stmt_list RBRACE { Block(List.rev $2)}
      | IF LPAREN expr RPAREN stmt ELSE stmt { If($3, $5, $7) }
      | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt {For($3,$5,$7,$9)}
      | WHILE LPAREN expr RPAREN stmt {While($3,$5)}

expr: LITERAL { Literal($1) }
      | STRINGLIT { StringLit($1) }
      | NEWSTRINGLIT { NewstringLit($1) }
      | STRINGLIT ASSIGN expr { Assign($1, $3) }
      | CHAR_LITERAL {Char_Lit($1)}
      | expr PLUS expr { Binop($1, Plus, $3) }
      | expr MINUS expr { Binop($1, Minus, $3) }
      | expr TIMES expr { Binop($1, Times, $3) }
      | expr EQUAL expr { Binop($1, Equal, $3) }
      | expr UNEQUAL expr {Binop($1, Unequal, $3)}
      | expr LESS expr { Binop($1, Less, $3) }
      | expr GREAT expr { Binop($1, Great, $3) }
      | expr LESSEQ expr { Binop($1, LessEQ, $3) }
      | expr GREATEQ expr { Binop($1, GreatEQ, $3) }
      | NOT expr { Unop(Not, $2) }
      | TRUE { BoolLit(true)}
      | FALSE { BoolLit(false)}
      | STRINGLIT LPAREN actual_opt RPAREN { Call($1, $3) }
      | STRINGLIT LBRACKET expr RBRACKET { Array($1,$3)}
      | STRINGLIT ASSIGN NEW LBRACKET expr RBRACKET {Init($1,$5)}
      | STRINGLIT LBRACKET expr RBRACKET ASSIGN expr {Arrayassign($1,$3,$6)}

expr_opt: /* nothing */ { Noexpr }
          | expr {$1}

```



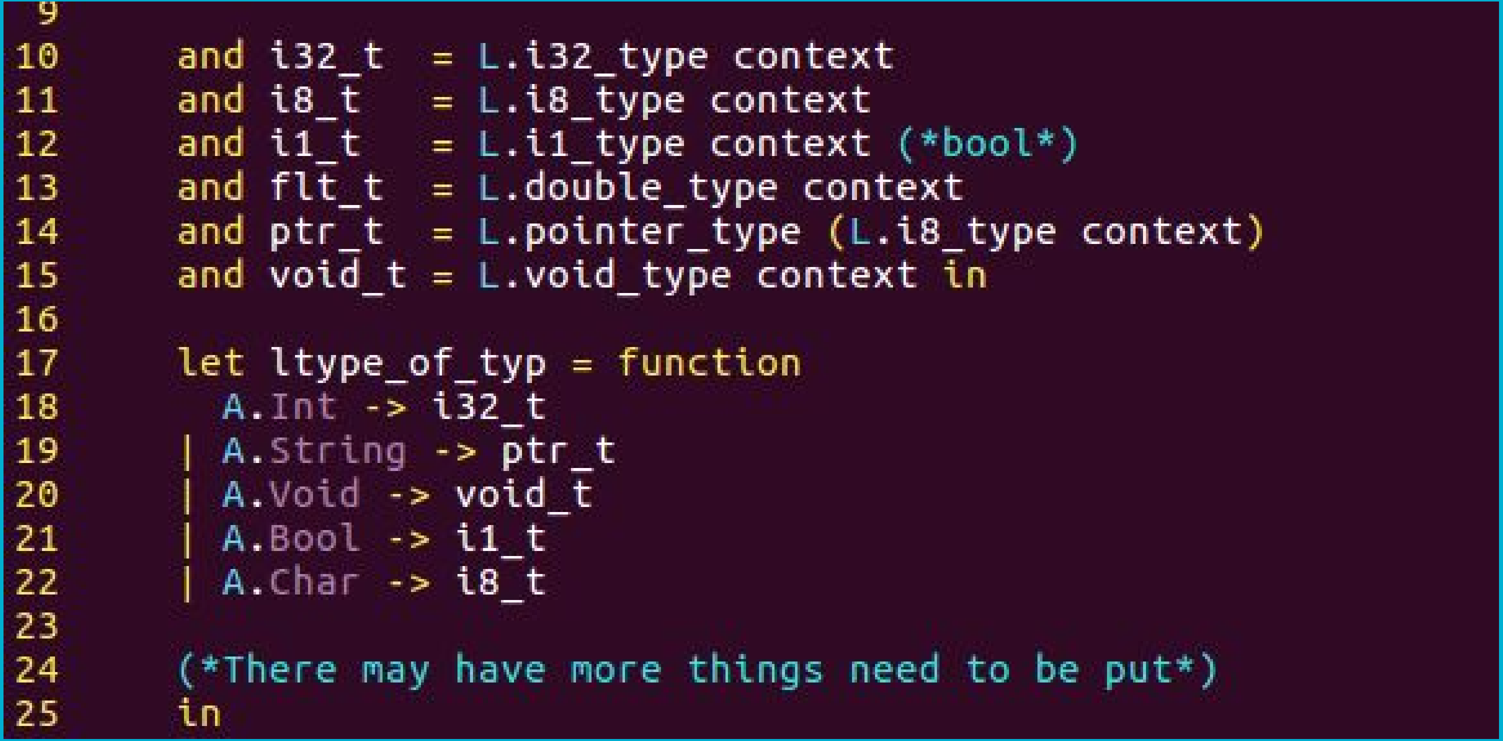
```
type typ = Int | Char | String | Void | Bool
type op = Plus | Minus | Times | Equal | Less | LessEQ | Great | GreatEQ | Unequal

type uop = Not

type crement = INCREMENT | DECREMENT

type bind = typ * string

type expr = Literal of int
          | StringLit of string
          | Binop of expr * op * expr
          | Assign of string * expr
          | Unop of uop * expr
          | Noexpr
          | BoolLit of bool
          | Call of string * expr list
          | NewstringLit of string
          | Char_Lit of char
          | Array of string * expr
          | Arrayassign of string * expr * expr
          | Init of string * expr
          | Arrayaccess of string*string*expr
type stmt = Block of stmt list
    | Expr of expr
    | If of expr * stmt * stmt
    | For of expr * expr * expr * stmt
    | While of expr * stmt
    | Return of expr
```

Semant



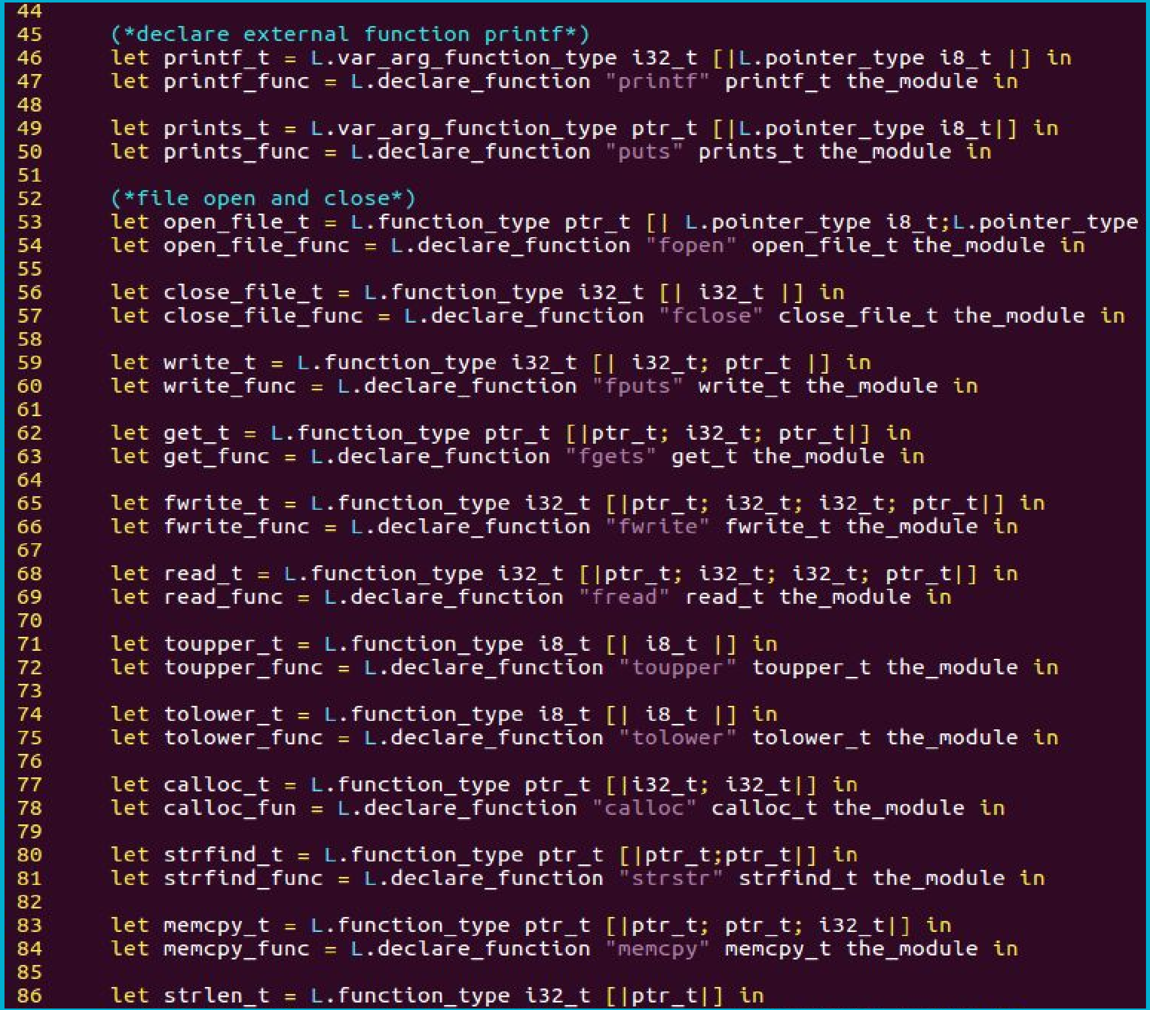
```
(**** Checking Functions ****)
if List.mem "print" (List.map (fun fd -> fd.fname) functions)
then raise (Failure ("function print may not be defined")) else ();

(*Check for duplicate. 2 functions cannot have same name, therefore also does not allow overload*)
report_duplicate (fun n -> "duplicate function " ^ n)
   (List.map (fun fd -> fd.fname) functions);

(* Function declaratoin for a named function (build in function) *)
(* Use 2 array to hold the details then throw to the built_in_decls by list.fold *)
let built_in_decls_funcs = [

    { typ = Char; fname = "tolower"; formals = [(Char, "x")]; locals = []; body = []};

    { typ = Char; fname = "toupper"; formals = [(Char, "x")]; locals = []; body = []};

    { typ = String; fname = "TAPE"; formals = [(String, "x");(String, "y")]; locals = []; body = [] };

    { typ = Void; fname = "print_i"; formals = [(Int, "x")] ; locals = []; body = [] };

    { typ = String; fname = "fget"; formals=[(String,"x");(Int,"y");(String, "z")]; locals=[];body=[]};

    { typ = String; fname = "open"; formals = [(String, "x");(String,"x")]; locals = []; body = [] };

    { typ = Int; fname = "write"; formals = [(String, "x");(Int,"y");(Int,"z");(String, "a")]; locals = []; body = [] };

    { typ = Void; fname ="print_c" ; formals=[(Char, "x")]; locals=[]; body=[]};

    { typ = String; fname = "read" ; formals=[(String,"x");(Int, "w");(Int, "y");(String, "z")]; locals=[]; body=[]};

    { typ =String; fname="find"; formals=[(String,"x");(String,"y")]; locals=[]; body=[]};

    { typ =String; fname="cpy"; formals=[(String, "x");(String, "y");(Int,"z")]; locals = []; body = []};

    { typ = Int; fname="length"; formals=[(String,"x")]; locals=[]; body=[]};
]

in

let built_in_decls_names = [ "tolower"; "toupper"; "TAPE"; "print_i";"fget"; "open"; "write";"print_c";"read";"find";"cpy";"length"];

in

let built_in_decls = List.fold_right2 (StringMap.add)
                      built_in_decls_names
                      built_in_decls_funcs
                      (StringMap.singleton "print_s"
                               { typ = Void; fname = "print_s"; formals = [(String, "x")]; locals = []; body = [] })
  in
let function_decls =
   List.fold_left (fun m fd -> StringMap.add fd.fname fd m)
                  built_in_decls functions
```

Codegen



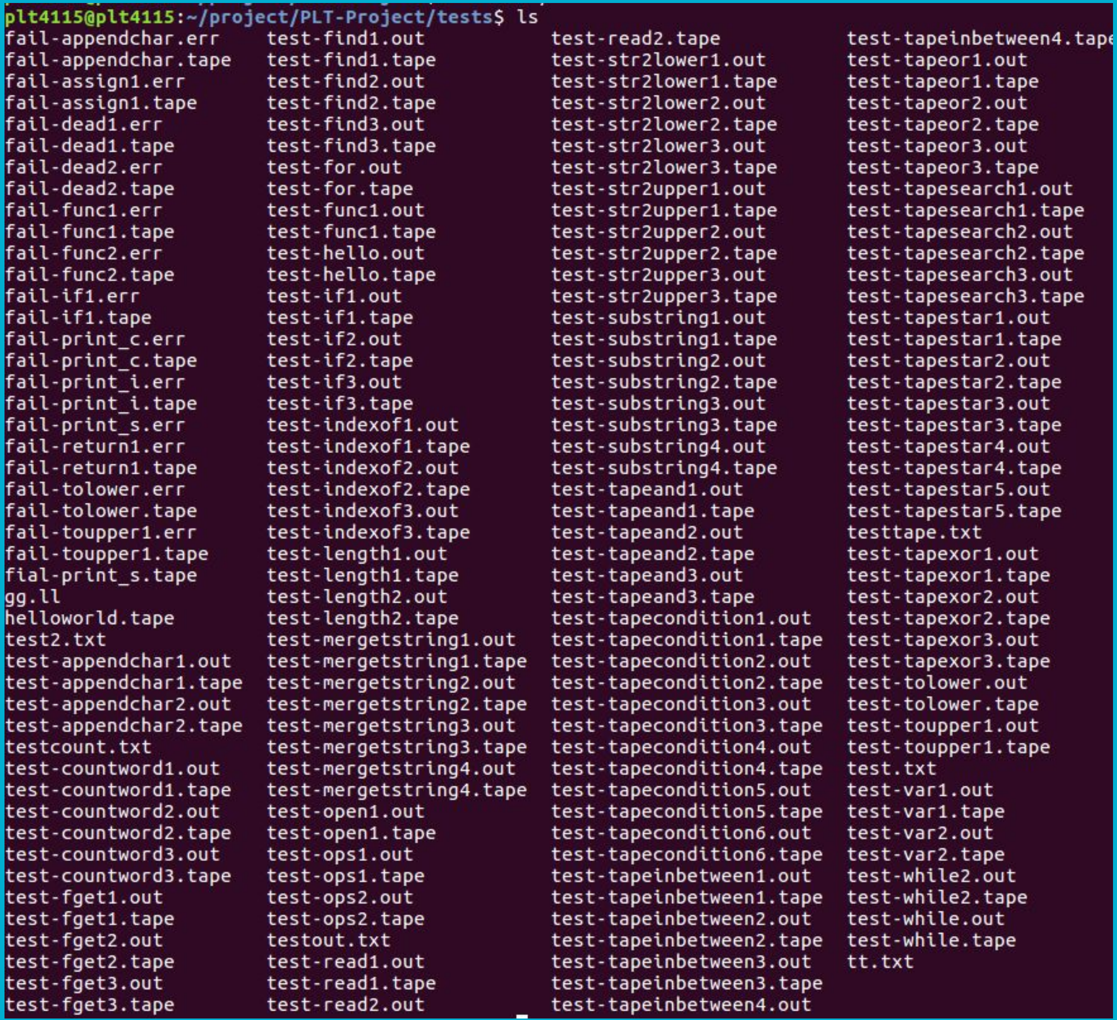
```
    and i32_t  = L.i32_type context
    and i8_t   = L.i8_type context
    and i1_t   = L.i1_type context (*bool*)
    and flt_t  = L.double_type context
    and ptr_t  = L.pointer_type (L.i8_type context)
    and void_t = L.void_type context in

    let ltype_of_typ = function
      A.Int -> i32_t
    | A.String -> ptr_t
    | A.Void -> void_t
    | A.Bool -> i1_t
    | A.Char -> i8_t

    (*There may have more things need to be put*)
    in
```

Codegen

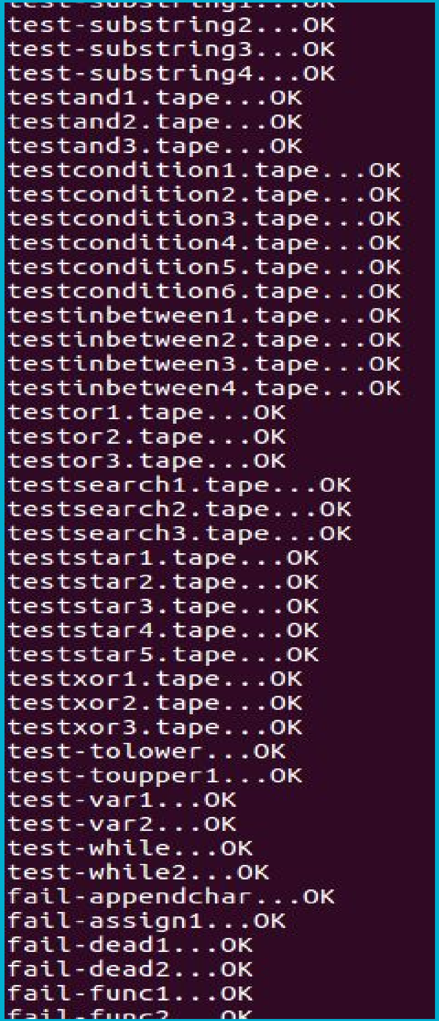


```
    (*declare external function printf*)
    let printf_t = L.var_arg_function_type i32_t [|L.pointer_type i8_t |] in
    let printf_func = L.declare_function "printf" printf_t the_module in

    let prints_t = L.var_arg_function_type ptr_t [|L.pointer_type i8_t|] in
    let prints_func = L.declare_function "puts" prints_t the_module in

    (*file open and close*)
    let open_file_t = L.function_type ptr_t [| L.pointer_type i8_t;L.pointer_type
    let open_file_func = L.declare_function "fopen" open_file_t the_module in

    let close_file_t = L.function_type i32_t [| i32_t |] in
    let close_file_func = L.declare_function "fclose" close_file_t the_module in

    let write_t = L.function_type i32_t [| i32_t; ptr_t |] in
    let write_func = L.declare_function "fputs" write_t the_module in

    let get_t = L.function_type ptr_t [|ptr_t; i32_t; ptr_t|] in
    let get_func = L.declare_function "fgets" get_t the_module in

    let fwrite_t = L.function_type i32_t [|ptr_t; i32_t; i32_t; ptr_t|] in
    let fwrite_func = L.declare_function "fwrite" fwrite_t the_module in

    let read_t = L.function_type i32_t [|ptr_t; i32_t; i32_t; ptr_t|] in
    let read_func = L.declare_function "fread" read_t the_module in

    let toupper_t = L.function_type i8_t [| i8_t |] in
    let toupper_func = L.declare_function "toupper" toupper_t the_module in

    let tolower_t = L.function_type i8_t [| i8_t |] in
    let tolower_func = L.declare_function "tolower" tolower_t the_module in

    let calloc_t = L.function_type ptr_t [|i32_t; i32_t|] in
    let calloc_fun = L.declare_function "calloc" calloc_t the_module in

    let strfind_t = L.function_type ptr_t [|ptr_t;ptr_t|] in
    let strfind_func = L.declare_function "strstr" strfind_t the_module in

    let memcpy_t = L.function_type ptr_t [|ptr_t; ptr_t; i32_t|] in
    let memcpy_func = L.declare_function "memcpy" memcpy_t the_module in

    let strlen_t = L.function_type i32_t [|ptr_t|] in
```

Testing

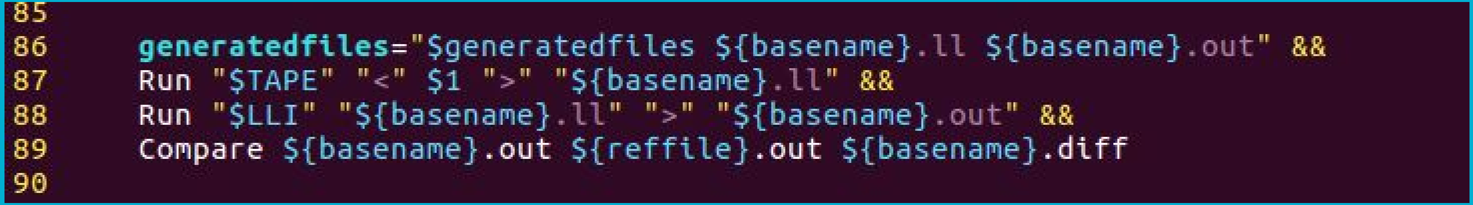


```
plt4115@plt4115:~/project/PLT-Project/tests$ ls
fail-appendchar.err    test-find1.out           test-read2.tape           test-tapeinbetween4.tape
fail-appendchar.tape   test-find1.tape          test-str2lower1.out       test-tapeor1.out
fail-assign1.err       test-find2.out           test-str2lower1.tape      test-tapeor1.tape
fail-assign1.tape      test-find2.tape          test-str2lower2.out       test-tapeor2.out
fail-dead1.err         test-find3.out           test-str2lower2.tape      test-tapeor2.tape
fail-dead1.tape        test-find3.tape          test-str2lower3.out       test-tapeor3.out
fail-dead2.err         test-for.out             test-str2lower3.tape      test-tapeor3.tape
fail-dead2.tape        test-for.tape            test-str2upper1.out       test-tapesearch1.out
fail-func1.err         test-func1.out           test-str2upper1.tape      test-tapesearch1.tape
fail-func1.tape        test-func1.tape          test-str2upper2.out       test-tapesearch2.out
fail-func2.err         test-hello.out           test-str2upper2.tape      test-tapesearch2.tape
fail-func2.tape        test-hello.tape          test-str2upper3.out       test-tapesearch3.out
fail-if1.err           test-if1.out             test-str2upper3.tape      test-tapesearch3.tape
fail-if1.tape          test-if1.tape            test-substring1.out       test-tapestar1.out
fail-print_c.err       test-if2.out             test-substring1.tape      test-tapestar1.tape
fail-print_c.tape      test-if2.tape            test-substring2.out       test-tapestar2.out
fail-print_i.err       test-if3.out             test-substring2.tape      test-tapestar2.tape
fail-print_i.tape      test-if3.tape            test-substring3.out       test-tapestar3.out
fail-print_s.err       test-indexof1.out        test-substring3.tape      test-tapestar3.tape
fail-return1.err       test-indexof1.tape       test-substring4.out       test-tapestar4.out
fail-return1.tape      test-indexof2.out        test-substring4.tape      test-tapestar4.tape
fail-tolower.err       test-indexof2.tape       test-tapeand1.out         test-tapestar5.out
fail-tolower.tape      test-indexof3.out        test-tapeand1.tape        test-tapestar5.tape
fail-toupper1.err      test-indexof3.tape       test-tapeand2.out         testtape.txt
fail-toupper1.tape     test-length1.out         test-tapeand2.tape        test-tapexor1.out
fial-print_s.tape      test-length1.tape        test-tapeand3.out         test-tapexor1.tape
gg.ll                  test-length2.out         test-tapeand3.tape        test-tapexor2.out
helloworld.tape        test-length2.tape        test-tapecondition1.out   test-tapexor2.tape
test2.txt              test-mergestring1.out    test-tapecondition1.tape  test-tapexor3.out
test-appendchar1.out   test-mergestring1.tape   test-tapecondition2.out   test-tapexor3.tape
test-appendchar1.tape  test-mergestring2.out    test-tapecondition2.tape  test-tolower.out
test-appendchar2.out   test-mergestring2.tape   test-tapecondition3.out   test-tolower.tape
test-appendchar2.tape  test-mergestring3.out    test-tapecondition3.tape  test-toupper1.out
testcount.txt          test-mergestring3.tape   test-tapecondition4.out   test-toupper1.tape
test-countword1.out    test-mergestring4.out    test-tapecondition4.tape  test.txt
test-countword1.tape   test-mergestring4.tape   test-tapecondition5.out   test-var1.out
test-countword2.out    test-open1.out           test-tapecondition5.tape  test-var1.tape
test-countword2.tape   test-open1.tape          test-tapecondition6.out   test-var2.out
test-countword3.out    test-ops1.out            test-tapecondition6.tape  test-var2.tape
test-countword3.tape   test-ops1.tape           test-tapeinbetween1.out   test-while2.out
test-fget1.out         test-ops2.out            test-tapeinbetween1.tape  test-while2.tape
test-fget1.tape        test-ops2.tape           test-tapeinbetween2.out   test-while.out
test-fget2.out         testout.txt              test-tapeinbetween2.tape  test-while.tape
test-fget2.tape        test-read1.out           test-tapeinbetween3.out   tt.txt
test-fget3.out         test-read1.tape          test-tapeinbetween3.tape
test-fget3.tape        test-read2.out           test-tapeinbetween4.out
```



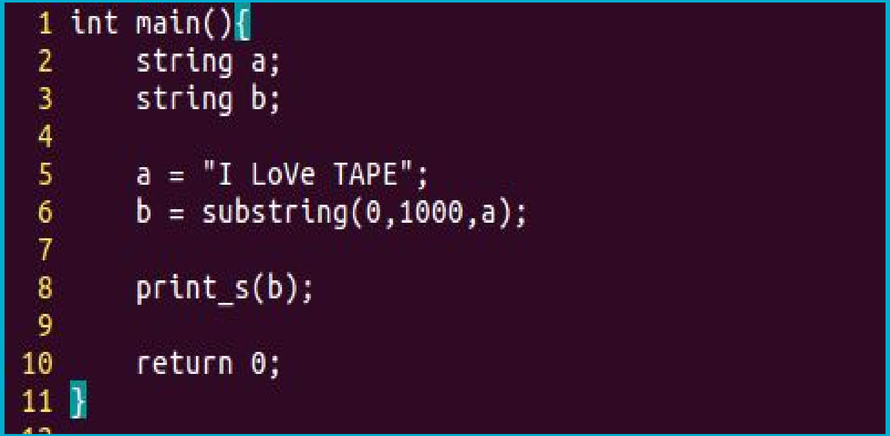
```
test-substring2...OK
test-substring3...OK
test-substring4...OK
testand1.tape...OK
testand2.tape...OK
testand3.tape...OK
testcondition1.tape...OK
testcondition2.tape...OK
testcondition3.tape...OK
testcondition4.tape...OK
testcondition5.tape...OK
testcondition6.tape...OK
testinbetween1.tape...OK
testinbetween2.tape...OK
testinbetween3.tape...OK
testinbetween4.tape...OK
testor1.tape...OK
testor2.tape...OK
testor3.tape...OK
testsearch1.tape...OK
testsearch2.tape...OK
testsearch3.tape...OK
teststar1.tape...OK
teststar2.tape...OK
teststar3.tape...OK
teststar4.tape...OK
teststar5.tape...OK
testxor1.tape...OK
testxor2.tape...OK
testxor3.tape...OK
test-tolower...OK
test-toupper1...OK
test-var1...OK
test-var2...OK
test-while...OK
test-while2...OK
fail-appendchar...OK
fail-assign1...OK
fail-dead1...OK
fail-dead2...OK
fail-func1...OK
```

Script

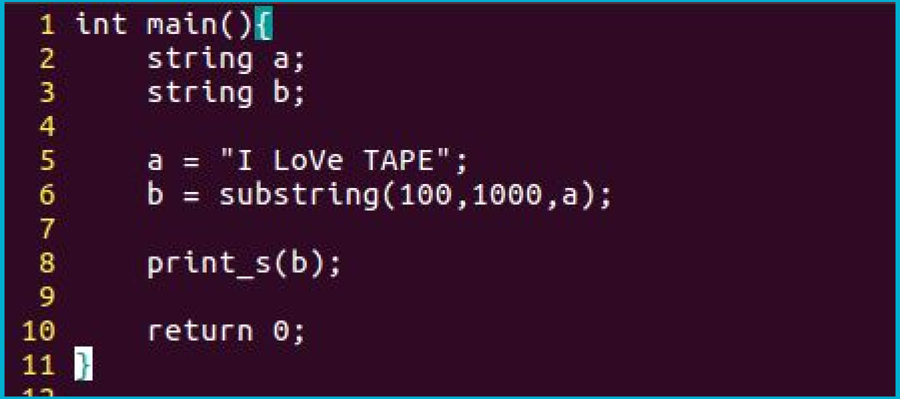


```
    generatedfiles="$generatedfiles ${basename}.ll ${basename}.out" &&
    Run "$TAPE" "<" $1 ">" "${basename}.ll" &&
    Run "$LLI" "${basename}.ll" ">" "${basename}.out" &&
    Compare ${basename}.out ${reffile}.out ${basename}.diff

```

Substring example



```
int main(){
    string a;
    string b;

    a = "I LoVe TAPE";
    b = substring(0,1000,a);

    print_s(b);

    return 0;
}
```



```
int main(){
    string a;
    string b;

    a = "I LoVe TAPE";
    b = substring(100,1000,a);

    print_s(b);

    return 0;
}
```

Library

Stdlib

Int countWord(string a, string f)

Int tape(string fn, string re)

indexOf(string t, char c)

String substring(int begin, int end, string s)

String str2Upper(string a)

String str2Lower(string a)

String mergeString(string a, string b)

String appendChar(string s, char a)

Int findreplace(string a, string b, string orig, string dest)

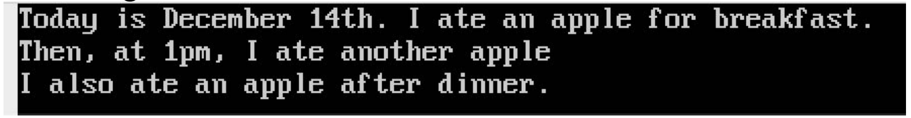
Demo

Demo 1: Count, Find & Replace

Goal:

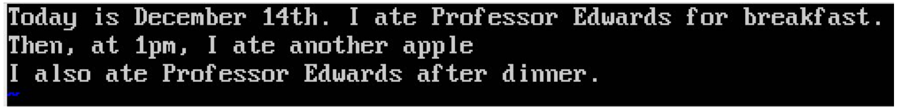
1) Find and print the number of "apple".
2) Replace "an apple" with "Professor Edwards".

originalFile.txt



```
Today is December 14th. I ate an apple for breakfast.
Then, at 1pm, I ate another apple
I also ate an apple after dinner.
```

destinationFile.txt

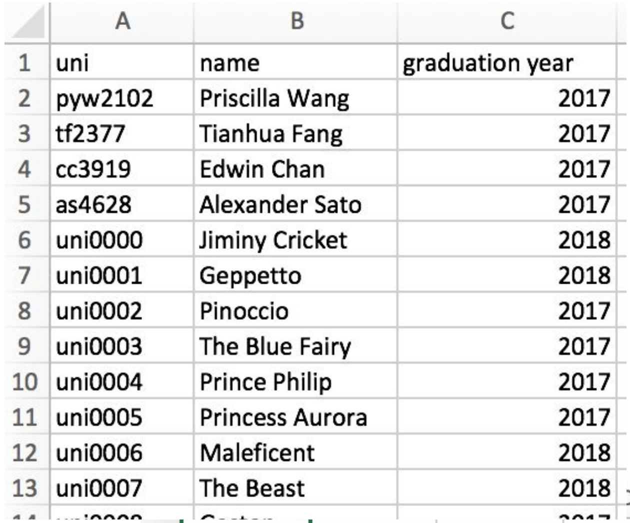


```
Today is December 14th. I ate Professor Edwards for breakfast.
Then, at 1pm, I ate another apple
I also ate Professor Edwards after dinner.
```

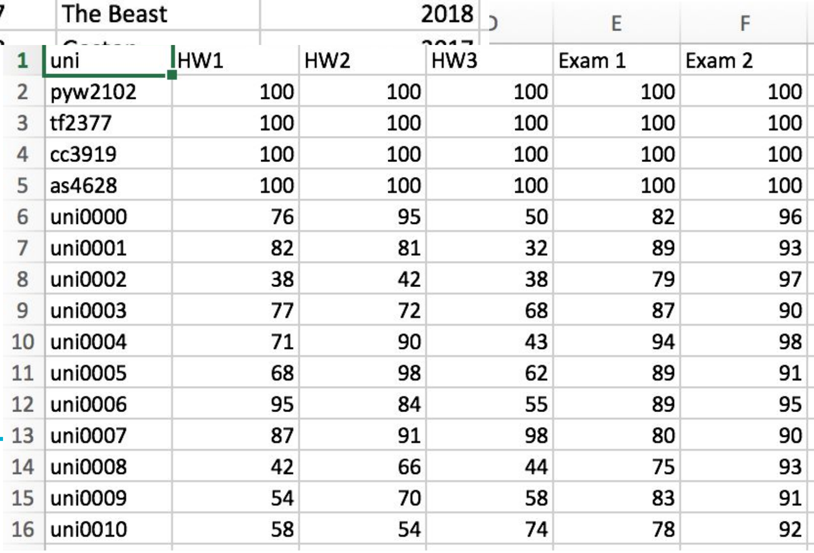
Demo 2: PLT Grading Example

studentinfo.csv gives us the student's uni, name, and graduating year.

studentgrades.csv shows the uni and the grades for each assignment



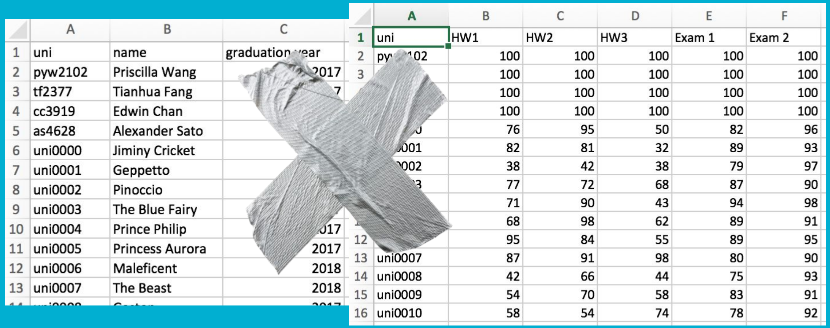
uni	name	graduation year
pyw2102	Priscilla Wang	2017
tf2377	Tianhua Fang	2017
cc3919	Edwin Chan	2017
as4628	Alexander Sato	2017
uni0000	Jiminy Cricket	2018
uni0001	Geppetto	2018
uni0002	Pinoccio	2017
uni0003	The Blue Fairy	2017
uni0004	Prince Philip	2017
uni0005	Princess Aurora	2017
uni0006	Maleficent	2018
uni0007	The Beast	2018



uni	HW1	HW2	HW3	Exam 1	Exam 2
pyw2102	100	100	100	100	100
tf2377	100	100	100	100	100
cc3919	100	100	100	100	100
as4628	100	100	100	100	100
uni0000	76	95	50	82	96
uni0001	82	81	32	89	93
uni0002	38	42	38	79	97
uni0003	77	72	68	87	90
uni0004	71	90	43	94	98
uni0005	68	98	62	89	91
uni0006	95	84	55	89	95
uni0007	87	91	98	80	90
uni0008	42	66	44	75	93
uni0009	54	70	58	83	91
uni0010	58	54	74	78	92

Goal:

Our goal is to create a file that has all of the students' unis and grades.



	A	B	C
1	uni	name	graduation year
2	pyw2102	Priscilla Wang	2017
3	tf2377	Tianhua Fang	7
4	cc3919	Edwin Chan	[illegible]
5	as4628	Alexander Sato	[illegible]
6	uni0000	Jiminy Cricket	[illegible]
7	uni0001	Geppetto	[illegible]
8	uni0002	Pinoccio	[illegible]
9	uni0003	The Blue Fairy	[illegible]
10	uni0004	Prince Philip	017
11	uni0005	Princess Aurora	2017
12	uni0006	Maleficent	2018
13	uni0007	The Beast	2018

	A	B	C	D	E	F
1	uni	HW1	HW2	HW3	Exam 1	Exam 2
2	py 102	100	100	100	100	100
3	[illegible]	100	100	100	100	100
[illegible]	[illegible]	100	100	100	100	100
[illegible]	[illegible]	100	100	100	100	100
[illegible]	0	76	95	50	82	96
[illegible]	001	82	81	32	89	93
[illegible]	002	38	42	38	79	97
[illegible]	3	77	72	68	87	90
[illegible]	[illegible]	71	90	43	94	98
[illegible]	[illegible]	68	98	62	89	91
12	[illegible]	95	84	55	89	95
13	uni0007	87	91	98	80	90
14	uni0008	42	66	44	75	93
15	uni0009	54	70	58	83	91
16	uni0010	58	54	74	78	92

We do this by "taping" the files together with tape.

Demo 3: Log Analysis

Find string with regular expressions

- List all visitor’s ip
  - Between “//-//”: => “/[/-/]/”
- Find pagetype with click event or pv event
  - And case: /cl.gif/&/game/
- Find the log for a certain pagetype
  - OR caes : /movie/|/manga/
- Find query with star or starwars
  - ? Cases: /star/?/wars/
- Find log that cannot have two words at the same time
  - XOR Cases: /movie/^/starwars/
- Kleene closure
  - * : /fo/*/d/

Typical nginx pingback log

[ip] [time] [Request][content pv/cl][link] [user agent]

Example:

```
[124.119.30.77] [23/Jun/2016:12:00:00
+0800:1466654400.023] "GET
/pv.gif?uigs_productid=index&uigs_uuid=5a84d33a-da2f-4
2d1-8ac5-e07&uigs_t=1466654412805&pagetype=rightv
r861&fQuery=%E6%90%9C%E7%8B%97%E6%B5%8F%E
8%A7%88&sub_pagetype=webgame&- HTTP/1.0"
"https://www.sogou.com/&query=%e6%90%9c%e7&ie=ut
f8" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/38.0.2125.122
Safari/537.36 SE 2.X MetaSr 1.0"
```

Lessons Learned

- Talk to people who have done this before!
  - TA has some insightful comments
  - Ask for help
- Start testing early
- Communication is important
  - Constantly update each other on the work in progress

Moving Forward

- Include stdlib
  - Encountered Problems
  - Move the lib
- Support Bash command