Graph Application Language
(GAL)
The Team

Anton: GAL language Guru
Andrew: Test Master
Donovan: Manager
Macrina: Standard Library Creator
The Aim

Creating a programming language that makes graph programming a piece of cake.
Compiler Architecture

GAL Source Code → SCANNER → Tokens → PARSER → AST → SEMANTIC CHECKER → LLVM COMPILER → LLVM → CODEGEN → SAST → Executable
Language Features

/*Types*/
int
string
node
data
type
value
array
list

/*Declaring a Function*/
int foo(int bar){
    bar = bar + 1;
    return bar;
}

/*Operators*/
+  -  *  /  =  >  <  >=  <=  ==
!=  !  |node/edge|

/*Control Flow*/
while(x == x){
    print_str("HELLO");
}
Language Features

List of Built-Ins for I/O and others:
+ print_int
+ print_str
+ print_endline
+ streq

List of Built-Ins for Graph:
+ length()
+ next()
+ pop()
+ peek()
+ add()

List of Built-Ins for Edges:
+ source()
+ dest()
+ weight()
The First Program- Hello World

/*Things written in here are comments and they are multi line compatible*/

int main(){
    string x = "HELLO WORLD!";
    print_str(x);
    /*Defining a Graph using node*/
    /*Adds an edge to the existing graph and updates it*/
    n1 = eadd(|"B",5,"E"|,n1);
    /*standard library function to print the list of edges*/
    print_elist(n1);
}

}
Test Suite

$ ./testall.sh
-n test_assignment_list1...
  OK
-n test_boolean_false...
  OK
-n test_boolean_true...
  OK
-n test_create_edge...
  OK
-n test_get_heaviest_graph_edge...
  OK

fail_assignment_edge1...
OK
fail_assignment_edge2...
OK
fail_assignment_int_to_string...
OK
fail_assignment_string_to_int...
OK
Lessons Learned

+ Programming in pairs helps to weed out bugs in more than half the time.
+ Git commit, Git add and Git push all day
+ Use Ubuntu
+ OCaml’s pattern matching is a god send.