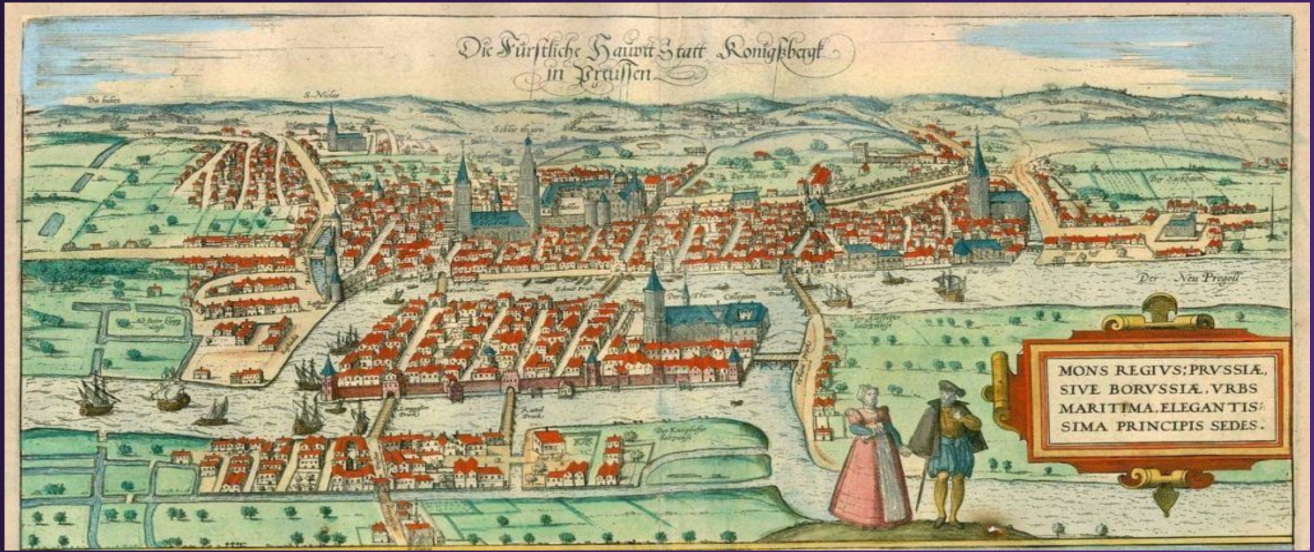
KONIG
Final Presentation

Lord Crawford
Matteo Sandrin
Delilah Beverly



The Team

Lord Crawford

SEAS '22 Computer Science

Graph theory was my fav part of data structures :)

Matteo Sandrin

SEAS '21 Computer Science

If it's broken, it's not my SEGFAULT

Delilah Beverly

Barnard '22 Computer Science

insert clever CS joke

What is Konig?

GOAL

Making the creation and manipulation of graphs easier and more enjoyable

Overview

- KONIG is graph manipulation programming language
- Named after the "Seven Bridges of **Konig**sberg"
- It provides strong primitives (Graph, Edge, Node)
- Features a mixture of Java-like and C-like syntax

```
ko int main() {

    node<string> n1;
    node<string> n2;
    node<string> n3;
    graph<string> g;
    int i;

    // initialize graph
    g = new graph{};
    n1 = new node{"Matteo"};
    n2 = new node{"Delilah"};
    n3 = new node{"Lord"};

    // add nodes to graph
    n1 @ g;
    n2 @ g;
    n3 @ g;

    // fully connected graph
    setEdge(g, n1, n2, 1.0);
    setEdge(g, n2, n3, 1.0);
    setEdge(g, n3, n1, 1.0);

    for (i = 0; i < g.nodes.length; i++) {
        printString(g.nodes[i].val);
    }

}
```

KONIG vs C vs Java

```
n = new node{}
```

Java-like object initialization

Most objects are heap-allocated (like Java)

Konig is a superset of MicroC

Java-like composite type syntax

```
node<int>
```

Key Features

Graph Features

Graphs:

```
graph<int> g1 = new graph{};

list<node<int> = g1.nodes;

list<edge> = g1.edges;

n @ g1; // add n1 to g1

n ! g2; // del n1 from g1

viz(g, "out.pdf");
```

Edges:

```
edge e = getEdge(g, n1, n2);

setEdge(g, n0, n1, 0);

setDirEdge(g, n0, n1, 0);

getEdge(g, n1, n2);

deleteEdge(g, n1, n2);

edge.weight;

edge.directed;
```

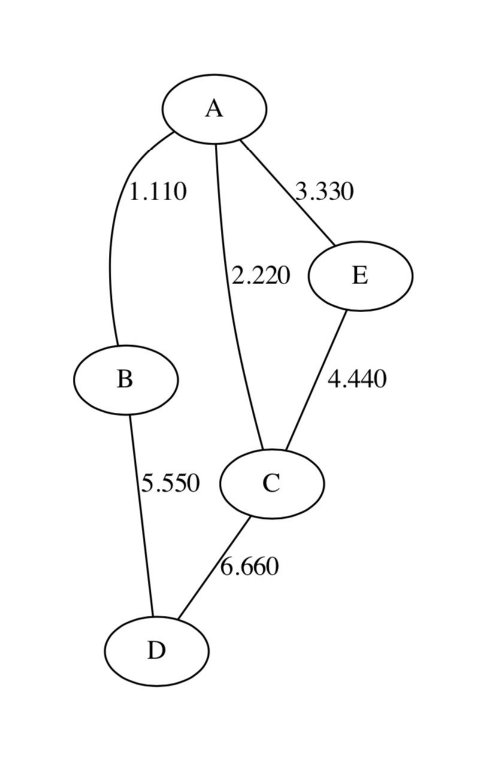
Nodes:

```
node<int> n0 = new node{0};

neighbors(g, n1);

n0.val;
```

Graph Visualization




```
ko int main() {
    node<string> a;
    node<string> b;
    node<string> c;
    node<string> d;
    node<string> e;
    graph<string> g;

    a = new node{"A"};
    b = new node{"B"};
    c = new node{"C"};
    d = new node{"D"};
    e = new node{"E"};
    g = new graph{};

    a @ g;
    b @ g;
    c @ g;
    d @ g;
    e @ g;

    setEdge(g, a, b, 1.11);
    setEdge(g, a, c, 2.22);
    setEdge(g, a, e, 3.33);
    setEdge(g, e, c, 4.44);
    setEdge(g, b, d, 5.55);
    setEdge(g, c, d, 6.66);

    viz(g, "./graph.pdf");
}
```

Types

- `int`
- `bool`
- `string`
- `float`
- `list<type>`
- `void`
- `edge`
- `node<type>`
- `graph<type>`

Functions

- Identified by the custom keyword "`ko`"
- C-style function syntax
- Extensive set of built-in functions

```
ko int add(int x, int y) {
        return x + y;
}
```

Operators

Operator	Operands	Return type

`a @ g`
 	
`a ! g`
	`a` is a `node`
	`g` is a `graph`
		`graph`

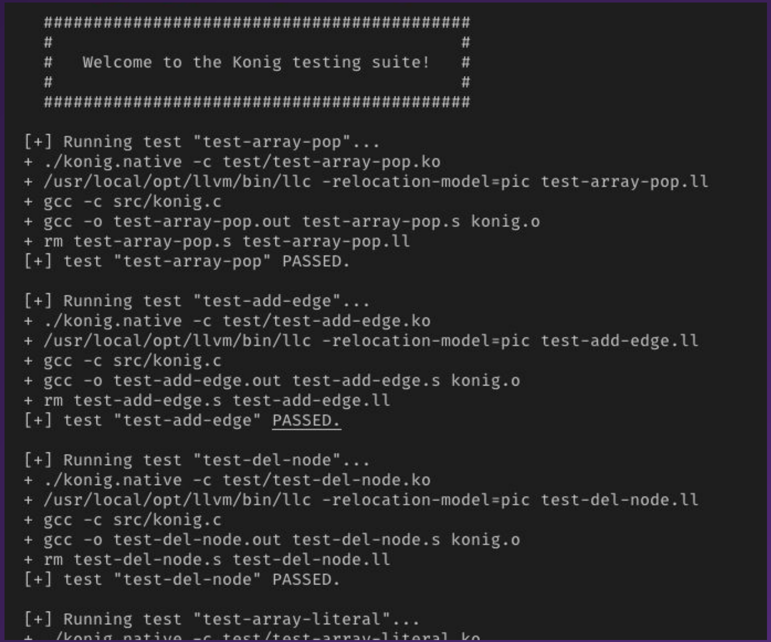
`a + b`
`a - b`
`a / b`
`a * b`
	`a` is an `int, float`
	`b` is an `int, float`
		`int, float`

`a > b`
`a < b`
`a => b`
`a <= b`
`a == b`
	`a` is any type
	`b` is any type
	
	`a` and `b` have the same type
		`bool`

`a and b`
`a or  b`
`not a`
	`a` is a `bool`
	`b` is a `bool`
		`bool`

Testing

- We built a custom testing script in Python
- We have pretty good coverage over the code base



```
###########################################
#                                         #
#   Welcome to the Konig testing suite!   #
#                                         #
###########################################

[+] Running test "test-array-pop"...
+ ./konig.native -c test/test-array-pop.ko
+ /usr/local/opt/llvm/bin/llc -relocation-model=pic test-array-pop.ll
+ gcc -c src/konig.c
+ gcc -o test-array-pop.out test-array-pop.s konig.o
+ rm test-array-pop.s test-array-pop.ll
[+] test "test-array-pop" PASSED.

[+] Running test "test-add-edge"...
+ ./konig.native -c test/test-add-edge.ko
+ /usr/local/opt/llvm/bin/llc -relocation-model=pic test-add-edge.ll
+ gcc -c src/konig.c
+ gcc -o test-add-edge.out test-add-edge.s konig.o
+ rm test-add-edge.s test-add-edge.ll
[+] test "test-add-edge" PASSED.

[+] Running test "test-del-node"...
+ ./konig.native -c test/test-del-node.ko
+ /usr/local/opt/llvm/bin/llc -relocation-model=pic test-del-node.ll
+ gcc -c src/konig.c
+ gcc -o test-del-node.out test-del-node.s konig.o
+ rm test-del-node.s test-del-node.ll
[+] test "test-del-node" PASSED.

[+] Running test "test-array-literal"...
+ ./konig.native -c test/test-array-literal.ko
```

DEMO!

THANK YOU!

Any questions?