

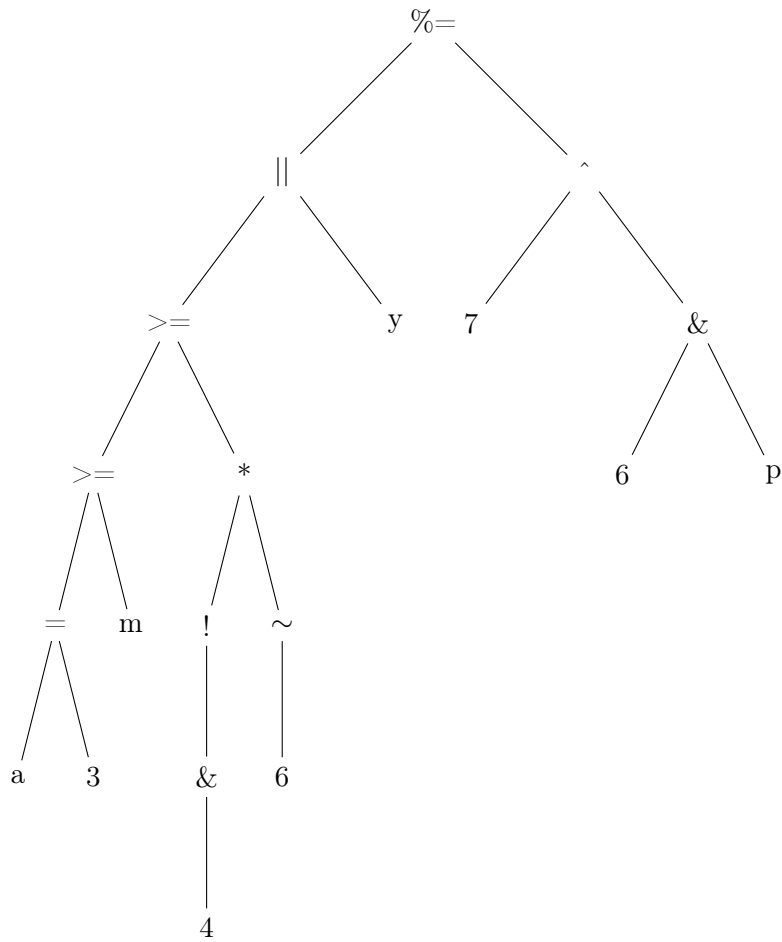
# Homework 4

Andrew Kowalczyk

October 31, 2013

## 1 Abstract Syntax Tree

$(a = 3) \geq m \geq ! \& 4 * \sim 6 \parallel y \% = 7 \wedge 6 \& p$



## 2 Javascript semicolon questions

### 2.a

When trying to call the function `f()`, it returns undefined. Javascript inserts semicolons into the program as so:

```
1 function f() {  
2     return;  
3     {x: 5};  
4 }
```

Python would remedy this because one would receive an indentation error trying to run the Python equivalent of code.

### 2.b

When trying to evaluate this expression, we receive this error `TypeError: Property 'b' of object #<Object> is not a function`. Javascript inserts semicolons into the program and interprets it as so:

```
1 var b = 8;  
2 var a = b + b(4 + 5).toString(16);
```

Python would remedy this because the lexical definition of statements says that logical lines end in with a newline.

### 2.c

After trying to evaluate the program, we receive this error, `TypeError: 'undefined' is not an object (evaluating '"mundo" ["Hola", "Ciao"].forEach')`. This is because it is attempting to use the `["Hola", "Ciao"]` as indexer operator to `"mundo"`. Again, Python would remedy this because the lexical definition of statements say that logical lines end in with a newline.

### 2.d

When trying to evaluate this, we are alerted with `"Goodbye"` then `"Hello"`. This is because `sayHello` is called with the anonymous function below due to being in a closure. Javascript interprets the code as this:

```
1 var sayHello = function () {  
2     console.log("Hello")  
3 }  
4  
5 (function () {  
6     console.log("Goodbye")  
7 })(sayHello)
```

### 3 What if local variables were allocated in static storage?

Consider this code:

```
1 #include <stdio.h>
2
3 int f(int x) {
4     int a = 0;
5     if (x == 0) {
6         a = 1;
7     } else {
8         return f(x - x) + a;
9     }
10    return 0;
11 }
12
13 int main() {
14     printf("%d\n", f(5));
15     return 0;
16 }
```

The program prints out 0. If local variables were allowed in static storage, the program would print out 1.

### 4 Scoping

#### 4.a Static Scoping

It would print out 1122 since the second setX() call changes the global variable to 2.

#### 4.b Dynamic Scoping

It would print out 1121 since the second setX() call only changes the local variable x.

### 5 Compiler operation order

Consider this python code:

```
1 a = 4
2 b = 0
3 c = 7
4 d = 7
5
6 def f(n):
7     global a
8     a = n
9     return a
10
```

```
11 print (a - f(b) - c * d)
```

If Python were to evaluate the the function first which would assign the global `a` to 0, then the expression could yield different results. If the lookup for `a` were to happen first, then `a` would stay as 4.

## 6 Explain the C declarations

**6.a** `double *a[n];`

This is an array of `n` pointers to doubles.

**6.b** `double (*b)[n];`

This is a pointer to an array of `n` doubles.

**6.c** `double (*c[n])();`

This is an array of `n` pointers pointing to functions returning a double.

**6.d** `double (*d())[n];`

This is a function returning a pointer to an array of `n` doubles.

## 7 Rewrite problem 6 in Go

**7.a** `var a [n]*float64`

**7.b** `var b (*)[n]float64`

**7.c** `var c [n] *func()float64`

**7.d** `var d func() (*[n]float64)`

## 8 Convert from infix $(-b + \sqrt{4 \times a \times c}) / (2 \times a)$

**8.a** Postfix

`0 b - c 4 a × × sqrt + 2 a × /`

**8.b** Prefix

`/ × 2 a + - 0 b sqrt × × 4 a c`

### 8.c Do you need a special symbol for unary negation? Why or why not?

There are two options. If you make parentheses required, you must assign another symbol for unary negation like  $\sim$ . If you don't make them required, you can simply subtract from 0.

## 9 Interleave using C-Style arrays

Code available at <http://codepad.org/Fw4xASrh> and on my github.

```
1 #include <iostream>
2 using namespace std;
3
4 void interleave(int a[], int lena, int b[], int lenb, int c[]) {
5     int aindex = 0;
6     int bindex = 0;
7
8     for (int i = 0; i < (lena + lenb); ) {
9         if (aindex < lena) {
10             c[i] = a[aindex];
11             aindex++;
12             i++;
13         }
14
15         if (bindex < lenb) {
16             c[i] = b[bindex];
17             bindex++;
18             i++;
19         }
20     }
21 }
22
23 int main () {
24
25     int a[50], b[50], c[100], lena, lenb, lenc, i;
26
27     cout << "Input number of elements in first array" << endl;
28     scanf("%d", &lena);
29
30     cout << "Input " << lena << " integers" << endl;
31     for (i = 0; i < lena; i++) {
32         scanf("%d", &a[i]);
33     }
34
35     cout << "Input number of elements in second array" << endl;
36     scanf("%d", &lenb);
37
38     cout << "Input " << lenb << " integers" << endl;
39     for (i = 0; i < lenb; i++) {
```

```

40         scanf("%d", &b[i]);
41     }
42
43     interleave(a, lena, b, lenb, c);
44     lenc = lena + lenb;
45
46     for (int i = 0; i < lenc; i++) {
47         cout << c[i] << " ";
48     }
49     cout << endl;
50
51     return 0;
52 }

```

## 10 Interleave using C++ vectors

Code available at <http://codepad.org/P6KKMBqY> and on my github.

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  template <typename T>
6  vector <T> zip(const vector <T> & a, const vector <T> & b, size_t len) {
7      vector <T> result;
8      int aindex = 0;
9      int bindex = 0;
10
11     for (size_t i = 0; i < len; ) {
12         if (aindex < a.size()) {
13             result.push_back(a[aindex]);
14             aindex++;
15             i++;
16         }
17
18         if (bindex < b.size()) {
19             result.push_back(b[bindex]);
20             bindex++;
21             i++;
22         }
23     }
24     return result;
25 }
26
27 int main() {
28     vector <int> a = {};
29     vector <int> b = {};
30     size_t lenc;
31     int i, n, m, lena, lenb;

```

```

32
33     cout << "Input number of elements in first array" << endl;
34     scanf("%d", &lena);
35
36     cout << "Input " << lena << " integers" << endl;
37     for (i = 0; i < lena; i++) {
38         scanf("%d", &n);
39         a.push_back(n);
40     }
41
42     cout << "Input number of elements in second array" << endl;
43     scanf("%d", &lenb);
44
45     cout << "Input " << lenb << " integers" << endl;
46     for (i = 0; i < lenb; i++) {
47         scanf("%d", &m);
48         b.push_back(m);
49     }
50
51     len = (a.size() + b.size());
52     vector<int> c = zip(a, b, lenc);
53     for(size_t i = 0; i < c.size(); i++) {
54         cout << c[i] << " ";
55     }
56     cout << endl;
57 }

```