CS 241 - Week 8 Tutorial

Error Checking and Code Generation

Spring 2018

1 Error Detection in WLP4

For each WLP4 program below, point out the error in the program and state whether it is a lexical (scanning), syntax (parsing), semantic, or run-time error.

1.
```
int wain(int x, int y) {
    return x^y;
}
```

2.
```
int wain(int x, int y) {
    int a = 100;
    int y = 0; // initialize y
    y = a*x;
    return y;
}
```

3.
```
int wain(int x, int y) {
    int a = 0;
    y = x / a;
    return y;
}
```

4.
```
int wain(int* a, int n) {
    // loop to get the last index
    while (idx < n) {
        idx = idx + 1;
    }
    return *(a + idx);
}
```

5.
```
int wain(int a, int b) {
    int *c = NULL;
    c = &a;
    int *d = NULL;
    d = &b;
    return (c - d);
}
```

6.
```
int wain(int x, int y) {
    int a = ‘a’;
    return a + x;
}
```

7.
```
int sub(int a){
    return a;
}

int sub(int a, int b){
    return sub(a) + sub(b);
}

int wain(int x, int y){
    int a = 0;
    a = sub(x,y);
    return *a;
}
```

8.
```
int f(int a, int b){
    return g(a) + g(b);
}

int g(int a){
    return a + 32;
}

int wain(int x, int y){
    return f(y, x);
}
```

9.
```
int wain(int x, int y){
    int a = 0;
    while (a < 10) {
        x = x+ y;
    }
    return x;
}
```

2 Symbol Table Error Checking

In the MIPS assembler you wrote for Assignments 3 and 4, you had to check for duplicate labels in one pass and check for missing labels in a second pass. Why don't you need a second pass for this in the WLP4 compiler?

3 Code Generation

Suppose we wanted to add `for` loops to WLP4 with the same syntax as C, except that the initializer cannot be used to declare new variables, only to set variables or evaluate other expressions.

1. Design one or more grammar rules for `for` loops. You may find the following rules a helpful starting point, and do not have to solve the other questions for these rules:

   $$\begin{aligned} &\text{aexpr} \rightarrow \text{expr} \\ &\text{aexpr} \rightarrow \text{ID BECOMES expr} \\ &\text{aexpr} \rightarrow \end{aligned}$$

2. List any changes to the scanning phase necessary for your new rules.
3. Describe type checking rules for your new rules.
4. Give pseudocode to generate assembly language code from a parse tree containing your new rules.