1 Error Detection in C and WLP4

1.1 Error Detection in WLP4

For each WLP4 program below, point out the error in the program and state whether it is a syntax error (i.e. something the scanner or parser would catch), semantic error (something semantic analysis would catch) or runtime error (any errors occurred while running the program).

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

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

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

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

```c
int wain(int x, int y) {
    int a = 'a';
    return a + x;
}
```
1.2 Error Detection in C

For each C program below, point out the error in the program and state whether it is a syntax error (i.e. something the scanner or parser would catch) or a semantic error (something semantic analysis would catch).

```c
float triple(float a) {
    return a * 3.0;
}

int main() {
    int* x, y;
    int a, b;
    a = triple(4.4);
    x = &a;
    y = &b;
    b = *x;
    return *y;
}
```

```c
int main() {
    double a = 2.0 * .4 / getRandom();
    int b;
    b = 2;
    return b;
}
```
2 Symbol Table Building and Type Checking

2.1 Symbol Tables

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 is this not necessary in the WLP4 compiler?

2.2 Type Checking

Determine if each of the following WLP4 code fragments is well-typed.

```c
int foo(int x, int y){
    return x + 7 * y + 1;
}
...
int a = 0;
int b = 0;
int* c = NULL;
int* d = NULL;

(1) *(d+(((c-&b)+d)-(c+(a*b))))=(c-d+*new int[d+b-c]);
(2) if(*(c+a%b)<(&a-&b)){println(&*&*c-(&b));}else{delete[]*d+&a-c;}
```