Abstract Syntax

Program = Declarations globals; Functions functions
Function = Function*
Function = Type t; string id; Decleretions params, locals; Block body
Type = int | boolean | float | char | void
Statement = Skip | Block | Assignment | Conditional | Loop | Call | Return
Call = String name; Expressions args
Expressions = Expression*
Return = Variable target; Expression result
Expression = Variable | Value | Binary | Unary | Call | Null

Type Rule 10.4 A Return statement must appear in the body of every non-void function except main, and its Expression must have the same Type as that function. -- modified -- A return statement must appear in the body of every function and its Expression must have the same type, except void, as that function. In the case of void function type, any return expression has a dummy meaning, and control is regained by the caller. Null could be that value

Type Rule 10.5 No return statement can appear in a void function. Return statements can appear in void functions but their effect is meaningless aside from re-establishing control to the caller.

Meaning Rule 10.2 The meaning of a Return is computed by replacing the value of the result Variable (the name of the called function) in the activation record by the value of the result Expression. -- modified -- except in the case of "return null" as in the case of a void function - the void function's activation record is popped from the stack and control returns to the system.

10.1
```
int h, i;
void B(int w){
    int j, k;
    j = 2 * w;
    w = w + 1;
}
void A(int x, int y){
    bool i, j;
    B(h);
}
int main(){
    int a, b;
    h = 5; a = 3; b = 2;
    A(a, b);
}
```

10.5
```
int fibonacci(int n){
    int fib0, fib1, temp, k;
    fib0 = 0; fib1 = 1; k = n;
    while (k > 0){
        temp = fib0;
        fib0 = fib1;
        fib1 = fib0 + temp;
        k = k - 1;
    }
    return fib0;
}
int main(){
    int answer;
    answer = fibonacci(8);
}
```

10.5
```
int main(){
    int answer;
    answer = A(1,2);
```
violates 10.6 every call expression must identify a non-void function
Here A refers to 10.1 where A is a void function;
)

```
int main() {
    int a, b;
    h=5; a=3; b=2;
    A(a, true);
```
Violates 10.7
Where the # of parameters provided by the call agree but the type of true is boolean, where the formal parameters of A require two ints.

```
int main() {
    bool answer;
    answer = fibonacci(8);
```
Violates 10.8
Because the calling function returns type: Int, and the declared variable answer is of type bool; this will result in a type error.

3. Consider the statement K=K-1; inside the while loop in the program of Figure 10.5.. Remove it.

(a) There will be no associated syntax or type errors when removing the decrement K=K-1. This does not violate any rules and will run endlessly.

(b) StackOverflowException occurs. There are no type errors or semantic/syntax errors but because the accumulator terminates the loop, stack overflow is inevitable.