/ * Long Test Case - Everything Except Boolean Declaration */

{    
/* Int Declaration */
int a
int b
a = 0
b = 0
/* While Loop */
while (a != 3) {
    print(a)
    while (b != 3) {
        print(b)
        b = 1 + b
        if (b == 2) {
/* Print Statement */
        print("there is no spoon" /* This will do nothing */) 
    }
    b = 0
    a = 1 + a
}
}

LEXER --> | T_OPENING_BRACE [ { ] on line 2...
LEXER --> | T_VARIABLE_TYPE [ int ] on line 4...
LEXER --> | T_ID [ a ] on line 4...
LEXER --> | T_VARIABLE_TYPE [ int ] on line 5...
LEXER --> | T_ID [ b ] on line 5...
LEXER --> | T_ID [ a ] on line 6...
LEXER --> | T_ASSIGNMENT_OP [ = ] on line 6...
LEXER --> | T_ID [ b ] on line 7...
LEXER --> | T_ASSIGNMENT_OP [ = ] on line 7...
LEXER --> | T_DIGIT [ 0 ] on line 7...
LEXER --> | T_WHILE [ while ] on line 9...
LEXER --> | T_OPENING_PARENTHESIS [ ( ] on line 9...
LEXER --> | T_ID [ a ] on line 9...
LEXER --> | T_INEQUALITY_OP [ != ] on line 9...
LEXER --> | T_DIGIT [ 3 ] on line 9...
LEXER --> | T_CLOSING_PARENTHESIS [ ) ] on line 9...
LEXER --> | T_OPENING_BRACE [ { ] on line 9...
LEXER --> | T_PRINT [ print ] on line 10...
LEXER --> | T_OPENING_PARENTHESIS [ ( ] on line 10...
LEXER --> | T_ID [ a ] on line 10...
LEXER --> | T_WHILE [ while ] on line 11...
LEXER --> | T_CLOSING_PARENTHESIS [ ) ] on line 11...
LEXER --> | T_OPENING_BRACE [ { ] on line 11...
LEXER --> | T_PRINT [ print ] on line 12...
LEXER --> | T_OPENING_PARENTHESIS [ ( ] on line 12...
LEXER --> | T_ID [ b ] on line 12...
LEXER --> | T_CLOSING_PARENTHESIS [ ) ] on line 12...
LEXER --> | T_ID [ b ] on line 13...
LEXER --> | T_ASSIGNMENT_OP [ = ] on line 13...
LEXER --> | T_DIGIT [ 1 ] on line 13...
LEXER --> | T_ASSIGNMENT_OP [ + ] on line 13...
LEXER --> | T_ID [ b ] on line 13...
LEXER --> | T_IF [ if ] on line 14...
LEXER --> | T_OPENING_PARENTHESIS [ { ] on line 14...
LEXER --> | T_EDOPS [ $ ] on line 23...
1 /*LongTestCase-EverythingExceptBooleanDeclaration*/ {/*IntDeclaration*/
int a = 0, b = 0; /*WhileLoop*/ while (a != 3) { print(a) while (b != 3) { print(b) b = 1 + b; if (b == 2) {
/*PrintStatement*/ print("there is no spoon") /*This will do nothing*/ } } b = 0; a = 1 + a; }$
Remove the comments?

Start lexing.

```
1 /*LongTestCase-EverythingExceptBooleanDeclaration*/{/*IntDeclaration*/
intaintba=0b=0/*WhileLoop*/while(a!=3){print(a)while(b!=3){print(b)b=1+bif(b==2)
{/*PrintStatement*/print("there is no spoon"/*Thiswilldonothing*/)}}b=0a=1+a}}$
```
```python
1 {int a=0, b=0;
while (a != 3)
    {print (a);
     while (b != 3)
        {print (b);
         b = 1 + b;
         if (b == 2) {print ("there is no spoon")};
     };
    };
} b=0; a=1+a
```
```python
int a = 0; b = 0;
while (a != 3)
{
print(a);
while (b != 3)
{
print(b);
b = 1 + b;
if (b == 2)
{
print("there is no spoon");
}
}
b = 0;
a = 1 + a;
}
```

Not a keyword.
Not an id.
Is a symbol. Note that and its positions. found symbol in positions 0-0
```plaintext
1 {int a=int b=0
while(a!=3)
{print(a)
while(b!=3)
{print(b)
b=1+b
if(b==2){print("there is no spoon")}}}
b=0}
a=1+a}
```
```c
{int a = 0, b = 0
while (a != 3) {
    printf("%d", a);
    while (b != 3) {
        printf("%d", b);
        if (b == 2) {
            printf("there is no spoon!");
        }
        b = 1 + b;
    }
    b = 0;
    a = 1 + a;
}
```

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.
```plaintext
{intaintba=0b=0while(a!=3)
 {print(a)while(b!=3)
 {print(b)b=1+bif(b==2){print("there is no spoon")}}}
b=0a=1+a}
```

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.
int a = 0; int b = 0;
while (a != 3) {
    print(a);
    while (b != 3) {
        print(b);
        b = 1 + b;
        if (b == 2) {
            print("there is no spoon");
        }
    }
    b = 0;
    a = 1 + a;
} $

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing. (Can we stop yet?)
```python
{int a, b = 0; while (a != 3) {
    print(a);
    while (b != 3) {
        print(b);
        b = 1 + b;
        if (b == 2) {
            print("there is no spoon");
        }
    }
    b = 0;
    a = 1 + a;
}
```
```plaintext
{intaintba=0b=0while(a!=3)
{print(a)while(b!=3)
{print(b)b=1+bif(b==2){print("there is no spoon")}}b=0a=1+a}}$
```

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.
```cpp
{int a = 0; b = 0;
while (a != 3)
{
    print(a);
    while (b != 3)
    {
        print(b);
        b = 1 + b;
        if (b == 2)
        {
            print("there is no spoon");
        }
    }
}
b = 0;
a = 1 + a;
}
```
```python
{intaint}
a=0 b=0
while(a!=3)
{
print(a) while(b!=3)
{
print(b)b=1+b
if(b==2){
print("there is no spoon")}
}
}
b=0 a=1+a
}
```
```plaintext
{intaintba=0b=0while(a!=3)
{print(a)while(b!=3)
{print(b)b=1+bif(b==2){print("there is no spoon")}}b=0a=1+a}}$
```

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.
1 {int\texttt{a}} b=0 while (a!=3) {
    print(a) while (b!=3) {
        print(b) b=1+b if (b==2) {print("there is no spoon")}
    }
} b=0 a=1+a

Not a keyword.  
Not an id.  
Is a symbol !!  

Symbols, like white space (if present and outside of quotes) mean that we can stop moving ahead and see what we’ve got so far.

We’ve got a symbol, so consume the input, emit the token, and reset the pointers.
{int a = 0; b = 0;
while (a != 3)
{
print(a);
while (b != 3)
{
print(b);
b = 1 + b;
if (b == 2) {
print("there is no spoon");
}
} b = 0;
a = 1 + a;
}
}
1 {int a=0,b=0}
while(a!=3)
{print(a) while(b!=3)
{print(b)b=1+b if(b==2) {print("there is no spoon")}}}
b=0
a=1+a} $
```c
int a = 0, b = 0;
while (a != 3)
{
    print(a);
    while (b != 3)
    {
        print(b);
        b = 1 + b;
        if (b == 2)
            print("there is no spoon");
    }
    b = 0;
    a = 1 + a;
}
```

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.
```java
int a = 0;
int b = 0;
while (a != 3) {
    print(a);
    while (b != 3) {
        print(b);
        b = 1 + b;
        if (b == 2) {
            print("there is no spoon");
        }
    }
    b = 0;
    a = 1 + a;
}
```
```c
1 {int a, b; a = 0; b = 0; while (a != 3) {
   print(a);
   while (b != 3) {
      print(b);
      b = 1 + b;
      if (b == 2) {
         print("there is no spoon");
      }
   }
   b = 0;
   a = 1 + a;
}}
```
1 \{ int \a = 0 \b = 0 \texttt{while}(a!=3) \\
\{ \texttt{print}(a) \texttt{while}(b!=3) \\
\{ \texttt{print}(b) \texttt{b = 1 + b} \texttt{if}(b==2) \{ \texttt{print}("there is no spoon") \} \} \} \b = 0 \a = 1 + a \} \}$
```plaintext
1 {intaintba=0b=0while(a!=3)
{print(a)while(b!=3)
{print(b)b=1+bif(b==2){print("there is no spoon")}}}
b=0a=1+a}$
```

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.

**lastPosition** 1
**currentPosition** 6
found int in positions 1-3
1 \{\texttt{int} \texttt{a}=0; \texttt{b}=0; \texttt{while} (a!=3) \{\texttt{print} (a) \texttt{while} (b!=3) \{\texttt{print} (b) \texttt{b}=1+b; \texttt{if} (b==2) \{\texttt{print} (\texttt{"there is no spoon"})\}\}\}\b=0; \texttt{a}=1+a\}\}$

Not a keyword.  
Not an id.  
Not a symbol.  
Not a digit.  
Not a char.  
Nothing.

lastPosition 1  
currentPosition 7  
found \texttt{int} in positions 1-3
int a=0; b=0; 
while (a!=3) 
{
  print(a);
  while (b!=3) 
  {
    print(b); 
    b=1+b; 
    if (b==2) 
    {
      print("there is no spoon");
    }
  }
  b=0; 
  a=1+a;
}
```c
int a, b;
while (a != 3) {
    print(a);
    while (b != 3) {
        print(b);
        b = 1 + b;
        if (b == 2) {
            print("there is no spoon");
        }
    }
}
```
```
1 {intaintba=0b=0while(a!=3)
 {print(a)while(b!=3)
 {print(b)b=1+bif(b==2){print("there is no spoon")}}b=0a=1+a}}$
```

Not a keyword.
Not an id.
Is a symbol.

Symbols, like white space (if present and outside of quotes) mean that we can stop moving ahead and see what we’ve got so far.

We’ve got a symbol, so consume the input, emit the token, and reset the pointers.
```c
1 {int a = 0; int b = 0;
while (a != 3)
    {print(a);
     while (b != 3)
         {print(b);
          b = 1 + b;
          if (b == 2)
              {print("there is no spoon");}
         }
     b = 0;
     a = 1 + a;}
}```
```plaintext
{
int a, b = 0;
while (a != 3)
{
    print(a);
    while (b != 3)
    {
        print(b);
        b = 1 + b;
        if (b == 2)
        {
            print("there is no spoon");
        }
    }
    b = 0;
    a = 1 + a;
}
```

Not a keyword.
Is an id.
(Actually, it is a char. But ...)
```c
1 {int a, int b; = 0
while (a != 3)
    {print(a)
     while (b != 3)
         {print(b)
          b = 1 + b;
          if (b == 2) {print("there is no spoon")}}}
    b = 0
    a = 1 + a}}$
```

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.
1 {int a, b; a = 0; b = 0; while (a != 3)
2     {print(a); while (b != 3)
3         {print(b); b = 1 + b; if (b == 2) {print("there is no spoon");}}}}
b = 0; a = 1 + a}
```plaintext
1 \{int\taboa=0\tabb=0\while(a!=3)\{print(a)\while(b!=3)\{print(b)b=1+b\if(b==2)\{print("there is no spoon")\}\}b=0a=1+a\}\}$
```
```java
{int a=0; b=0; while (a!=3) {
    print(a);
    while (b!=3) {
        print(b);
        b=1+b;
        if (b==2) {print("there is no spoon");}
    }
    b=0;
    a=1+a;
}}$
```

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.
```python
1 {int a, b = 0; while (a != 3)
2     {print(a) while (b != 3)
3         {print(b) b = 1 + b if (b == 2) {print("there is no spoon")}}}
b = 0; a = 1 + a}
```

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.
1 \{ int a, int b = 0; \\
while (a != 3) \\
{ print(a); \\
while (b != 3) \\
{ print(b); b = 1 + b; \\
if (b == 2) { print("there is no spoon"); } } } b = 0; a = 1 + a; \}$
```c
1 { int a, b; 
   a = 0; 
   b = 0; 
   while (a != 3) {
     print(a);
     while (b != 3) {
       print(b);
       b = 1 + b;
       if (b == 2) {
         print("there is no spoon");
       }
     }
   }
   b = 0;
   a = 1 + a;
}
```
```python
1 \{int a=0; b=0; while (a!=3) {print(a); while (b!=3) {print(b); b=1+b; if (b==2) {print("there is no spoon")};};}; b=0; a=1+a};}
```

Not a keyword. Is an id. found id in positions 9-9
{int a = 0; b = 0; while (a != 3) {
    print (a);
    while (b != 3) {
        print (b);
        b = 1 + b;
        if (b == 2) {
            print ("there is no spoon");
        }
    }
} b = 0; a = 1 + a; }

lastPosition 9
currentPosition 10
found id in positions 9-9

Not a keyword.
Not an id.
Is a symbol.
Consume input.
Emit token.
Reset pointers.
{int a, b; while (a != 3)
    {print(a) while (b != 3)
        {print(b) b = 1 + b; if (b == 2) {print("there is no spoon")}}}
} b = 0; a = 1 + a}
{int a, b = 0; while (a != 3) {
    print(a);
    while (b != 3) {
        print(b);
        b = 1 + b;
        if (b == 2) {
            print("there is no spoon");
        }
    }
    b = 0;
    a = 1 + a;
}}

Not a keyword.
Not an id.
Is a symbol.
```c
{int a, b; b = 0; while (a != 3) {
    print(a); while (b != 3) {
        print(b); b = 1 + b;
    }
    if (b == 2) {print("there is no spoon");}
} b = 0; a = 1 + a;}
```

Remember how we wondered about stopping here? We can because there are no symbols with length > 2 in our grammar. This is either one of those symbols, or it’s whatever we currently matched, or it’s an error. In any case we don’t need to look further.

int a = 0; b = 0;
while (a != 3) {
    print(a);
    while (b != 3) {
        print(b);
        b = 1 + b;
        if (b == 2) {
            print("there is no spoon");
        }
    }
}
}
```python
1 {intaintba=0b=0while(a!==3)
 {print(a)while(b!=3)
 {print(b)b=1+bif(b==2){print("there is no spoon")}}b=0a=1+a}}$
```
It's on and on and on and on...
The lex don't stop until the break of dawn.