/* 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
    }
    b = 0
    a = 1 + a
}
int ia = 0;
int ib = 0;
while (ia != 3) {
    print(ia);
    while (ib != 3) {
        print(ib);
        ib = 1 + ib;
        if (ib == 2) {
            print("there is no spoon");
        }
    }
}
ib = 0;
a = 1 + a;

b = a;
while (b != 3) {
    print(b);
    b = 1 + b;
    if (b == 2) {
        print("there is no spoon");
    }
}

print("This will do nothing");

Remove the comments.

Start lexing.
{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}$
```plaintext
{
  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");
}
} 
} 
```

Not a keyword.
Not an id.
Is a symbol. Note that and its positions.

---

**lastPosition**: 0
**currentPosition**: 0

---

**found symbol in positions 0-0**
```c
{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");
        }
    }
}
}```
```c
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");
        }
    }
}
```
{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; 
}
```bash
{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. (Can we stop yet?)
```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=0a=1+a}}$
```

Not a keyword.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.
```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");
        }
    }
}
```

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

lastPosition 0
currentPosition 6
found symbol in positions 0-0
```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. (Are we there yet?)
```python
{intaintb
  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
{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.
```python
{intaintba=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 !!

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.
```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}}$
```
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}}$
```plaintext
{int}intba=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}
```
```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;
}}$
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
{\texttt{int} a = 0; \texttt{int} b = 0;
while (a != 3) {
    \texttt{print}(a);
    while (b != 3) {
        \texttt{print}(b);
        b = 1 + b;
        if (b == 2) {
            \texttt{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.
{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
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.
Not an id.
Not a symbol.
Not a digit.
Not a char.
Nothing.
```plaintext
1 {intaint\{ba=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=0a=1+a}\$
```

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

```
lastPosition  1
currentPosition  8
found int in positions 1-3
```

```
LEXER --> | T_OPENING_BRACE [ { ] on line 1...
LEXER --> | T_VARIABLE_TYPE [ int ] on line 1...
LEXER --> | T_ID [ a ] on line 1...
LEXER --> | T_VARIABLE_TYPE [ int ] on line 1...
LEXER --> | T_ID [ b ] on line 1...
LEXER --> | T_ID [ a ] on line 1...
LEXER --> | T_ASSIGNMENT_OP [= ] on line 1...
LEXER --> | T_DIGIT [ 0 ] on line 1...
LEXER --> | T_ID [ b ] on line 1...
LEXER --> | T_ASSIGNMENT_OP [= ] on line 1...
LEXER --> | T_DIGIT [ 0 ] on line 1...
LEXER --> | T_WHILE [ while ] on line 1...
LEXER --> | T_OPENING_PARENTHESIS [ ( ] on line 1...
LEXER --> | T_ID [ a ] on line 1...
LEXER --> | T_INEQUALITY_OP [ != ] on line 1...
LEXER --> | T_DIGIT [ 3 ] on line 1...
LEXER --> | T_CLOSING_PARENTHESIS [ ) ] on line 1...
LEXER --> | T_OPENING_BRACE [ { ] on line 1...
LEXER --> | T_PRINT [ print ] on line 1...
LEXER --> | T_OPENING_PARENTHESIS [ ( ] on line 1...
LEXER --> | T_ID [ b ] on line 1...
LEXER --> | T_OPENING_PARENTHESIS [ ( ] on line 1...
LEXER --> | T_ID [ b ] on line 1...
LEXER --> | T_CLOSED_PARENTHESIS [ ] ] on line 1...
LEXER --> | T_OPENING_BRACE [ { ] on line 1...
LEXER --> | T_PRINT [ print ] on line 1...
LEXER --> | T_OPENING_PARENTHESIS [ ( ] on line 1...
LEXER --> | T_ID [ a ] on line 1...
LEXER --> | T_INEQUALITY_OP [ != ] on line 1...
LEXER --> | T_CLOSED_PARENTHESIS [ ] ] on line 1...
LEXER --> | T_OPENING_BRACE [ { ] on line 1...
LEXER --> | T_PRINT [ print ] on line 1...
LEXER --> | T_OPENING_PARENTHESIS [ ( ] on line 1...
LEXER --> | T_ID [ b ] on line 1...
LEXER --> | T_CLOSED_PARENTHESIS [ ] ] on line 1...
LEXER --> | T_OPENING_BRACE [ { ] on line 1...
LEXER --> | T_ID [ b ] on line 1...
LEXER --> | T_ASSIGNMENT_OP [= ] on line 1...
LEXER --> | T_DIGIT [ 1 ] on line 1...
LEXER --> | T_ADDITION_OP [ + ] on line 1...
LEXER --> | T_ID [ b ] on line 1...
LEXER --> | T_WHILE [ if ] on line 1...
LEXER --> | T_OPENING_PARENTHESIS [ { ] on line 1...
LEXER --> | T_ID [ b ] on line 1...
LEXER --> | T_VARIABLE_TYPE [ int ] on line 1...
LEXER --> | T_ID [ a ] on line 1...
LEXER --> | T_ASSIGNMENT_OP [= ] on line 1...
LEXER --> | T_DIGIT [ 0 ] on line 1...
LEXER --> | T_ID [ b ] on line 1...
LEXER --> | T_ASSIGNMENT_OP [= ] on line 1...
LEXER --> | T_DIGIT [ 1 ] on line 1...
LEXER --> | T_ADDITION_OP [ + ] on line 1...
LEXER --> | T_ID [ a ] on line 1...
LEXER --> | T_IF [ if ] on line 1...
LEXER --> | T_OPENING_PARENTHESIS [ { ] on line 1...
LEXER --> | T_EDPS [ $ ] on line 1...
```
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} $
{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}
```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;
}}
```
```c
{int a, int 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;}
```
```c
1 {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;
}}$
```c
{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.
Not a symbol.
Not a digit.
Not a char.
Nothing.
```c
1 {int a, b; a = 0; b = 0; while (a != 3) {
   printf(a); 
   while (b != 3) {
      printf(b); 
      b = 1 + b; 
      if (b == 2) {
         printf("there is no spoon");
      }
   }
} 

lastPosition 4
currentPosition 7
found id in positions 4-4
```
```c
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");
        }
    }
    a = 1 + a;
}
```

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

```c
    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");
            }
        }
        a = 1 + a;
    }
```
```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; 
} $
```
```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;
} $
```python
# {inta}
# {intb}
# {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
# {b}
```
```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
```
```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;
```
```c
1 {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}
}
```
```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; }
```
Not a keyword.
Not an id.
Not a symbol.
Is a digit.

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.

1
{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;
}$
}

```python
1 {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;
}

```

It's on and on and on and on...
The lex don't stop until the break of dawn.