Note: corrections are in red.

[1] Do we have const as a keyword ? Do we have an exhaustive list of supported keywords ?

No const is not a keyword here. You can treat it just like any other variable.

Exhaustive list of keywords: {`break`, `case`, `char`, `continue`, `default`, `do`, `double`, `else`, `extern`, `float`, `for`, `if`, `int`, `long`, `return`, `short`, `sizeof`, `struct`, `switch`, `void`, `while`}.

[2] Are things like long long int supported?

Yes.

[3] In C expressions like a+-b and a-+b are valid but a++b and a - - b are not valid. Do we have to handle those?

`a+-b`, `a-+b`: valid.

`a++b`, `a--b`: invalid.

[4] short float, long double?

Yes. Any combination thereof.

[5] should we give error if there are 2 defaults inside a switch statement?

No.

[6] Adding to Pragyan's question, what about a+++b?

```
int main()
{
        a+++b; //valid. also a++-b
        main(a+++b); //valid. also main(a++-b);

        main(a+++b;); //invalid
}
```

[7] int n = "none" ? is it supposed to be valid

Yes.

[8] Why would itonly require ; after ++

[9] int c = a+++b; This actually runs in C

Yes. This statement is also valid here.

[10] what about “System.out.print();”

Valid. This “.” operator of structure will handle this. The parser thinks System, out, and print are identifiers. Here are more valid examples.

```
int main()
{
        iitm.chennai.cs3300(); //valid
        a.a.a.a.a.a.a.main()=9; //valid
        iitm.chennai.a=h.g(g(g())); //valid
}
```

[11] Random strings?

You do not need worry about these. Parser will anyway throw syntax error as it is gramatically incorrect.

```
,,,<>()
{
        S#!\~();
        nit **ff%%;
}
```

```
***parsing terminated*** [syntax error]
```

[12] distinguish unary + binary +

To be handled in YACC specification. In LEX, include the following:

```
"+" {return '+'; //this is for both unary and binary.}
```

Here is an example.

```
int main()
{
        {
                a=-b;
                aa=b+a;
                main();
        }
}

***parsing successful***
#global_declarations = 1
#function_definitions = 1
#integer_constants = 0
#pointers_declarations = 0
#ifs_without_else = 0
if-else max-depth = 0
```

[13] Do we need to take care of the context in break, continue, return?

No. Here is an example.

```
int main()
{
        int a;
        {
                break;
                continue;
                return;
        }
}

***parsing successful***
#global_declarations = 1
#function_definitions = 1
#integer_constants = 0
#pointers_declarations = 0
#ifs_without_else = 0
if-else max-depth = 0
```

[14] What are multiple strings in same line?

Here is a valid example.

```
int main()
{
        char *test_str = "Hello World" " part 1" "part 2"; //valid
}

***parsing successful***
#global_declarations = 1
#function_definitions = 1
#integer_constants = 0
#pointers_declarations = 1
#ifs_without_else = 0
if-else max-depth = 0
```