***Context Free Grammar***

**Start:**

<start> → <start\_body> main(){<main\_body> }

<start\_body> → <dec><start\_body>|<func\_st><start\_body> | <struct\_dec><start\_body>| <class\_dec><start\_body>|<struct\_obj><start\_body> | <obj\_decl><start\_body> | ε

**Main:**

<main\_body> → <dec><main\_body>|<if\_else><main\_body>| <break><main\_body> | <continue><main\_body> | <OE>;<main\_body>| <try\_catch><main\_body>| <for\_st><main\_body>| <print><main\_body>| <struct\_obj><main\_body> | <obj\_decl><main\_body> |<assign\_st><main\_body> | <struct\_dec><main\_body> |<return><main\_body>|<class\_dec><main\_body>|ε

**Declaration and Initialization:**

<dec>→<DT1> ID <init> ;

<init>→ <SA><init5> | [<init1> |,ID<init>| ε

<init1>→ <OE>]<Arr><init6> | ]<Arr><init6>

<init6> → <SA><init7> | ε

<init5>→<OE> | <take>

<init7> → {<init2>}

<init2>→ <or><init8>| {<OE><init2>}<init2> | <OE><init2> | ε | }<init2>

<init8> → <OE><init2> | {<OE><init2>}

<SA>→ =

<or> → ,

<const>→int\_const|float\_const|char\_const|string\_const

<DT>→ int | float | char | string | bool

<DT1>→ const <DT> | <DT>

<Arr>→ [<OE> ]<Arr> | ε

<dec11>→<DT1> ID <init22>

<init22>→ <SA><OE> | [<init33> | ε

<init33>→ <OE>]<Arr>| ]<Arr>

<SA>→ =

<const>→int\_const|float\_const|char\_const|string\_const

<DT>→ int | float | char | string | bool

<DT1>→ const <DT> | <DT>

**Assignment\_statement:**

<assign\_st>→ assign <D2><X2><list>;

<D2> -> Inc\_Dec <inc\_dec\_op><p> | <p1> | ε

<list> → =<SSA>

<SSA> → <OE> | <take>

<dot22> → .ID<dot22> | ε | (para).ID<dot22> | [<OE>]<dot22>

<X2> → ID<dot22> | (ID<dot22>)

**Print():**

<print> →print<index>;

<index>→(<OE>)

**Throw():**

<throw> →throw<index> ;

**If Else:**

<if\_else>→ if<if2>

<if2>→(<cond>)<if\_body><elif>

<elif>→elif <if2> | else:<if\_body> | ε

< if\_body>→ {< if\_SST>} |;

<cond>→ <OE>

<if\_SST> -> <dec><if\_SST> | <if\_else><if\_SST> | <try\_catch><if\_SST> | <OE>;<if\_SST> | <struct\_obj><if\_SST> | <obj\_decl><if\_SST>| <for\_st><if\_SST> | <print><if\_SST>| <return><if\_SST> | <break><if\_SST> | <throw><if\_SST> |<continue><if\_SST> |<assign\_st><if\_SST> | ε

**Function:**

<func\_st> → <key> def <DT2> ID <func\_I>

<func\_I> →(<param>) <override> < funct\_body>

<param> →<dec11><param1> | <struct\_obj><param1> | <obj\_decl><param1> | ε

<param1> →,<param2> <param1>| ε

<param2> → <dec11> | <struct\_obj> | <obj\_decl>

<key> →virtual | ε

< funct\_body>→ {< funct\_SST>} |;

<funct\_SST>→<dec><funct\_SST>|<if\_else><funct\_SST>| <break><funct\_SST> |<continue><funct\_SST> | <OE>;<funct\_SST>| <return><funct\_SST>|<try\_catch><funct\_SST>| <for\_st><funct\_SST>|<print><funct\_SST>|<struct\_obj> <funct\_SST> |<obj\_decl><funct\_SST>|<assign\_st><funct\_SST>| <struct\_dec><funct\_SST>|<class\_dec><funct\_SST>|ε

<DT2> → void | int | float | char | string | bool

**Function\_Call:(fixed)**

<func\_call1> → <p1> ID (<para>);

<para>→<OE><para1>| ε

<para1>→,<OE><para1>| ε

<p1> → this.

**Return:**

<return> → return<index1>;

<index1> → <OE> |ε

**For:**

<for\_st>→for(<init11>;<cond11>;<inc\_dec11>)<for\_body>

<init11>→ <dec> | <assign\_st>| ε

<cond11>→ <OE> |ε

<LO>→ <&> | <|>

<inc\_dec11>→ <Inc\_Dec\_obj\_call>| ε

<inc\_dec\_op>→ ++ | --

< for\_body>→ {< funct\_SST>} | ;

<for\_SST> → <dec> <for\_SST> | <if\_else> <for\_SST>| <break> <for\_SST> | <continue><for\_SST> | <return><for\_SST>| <OE>; <for\_SST> | <try\_catch><for\_SST>| <for\_st><for\_SST>| <print><for\_SST>| <struct\_obj><for\_SST> | <obj\_decl><for\_SST> | <assign\_st><for\_SST> |ε

**Break:**

<break> → break;

**Continue:**

<continue> → continue;

**Struct declaration:**

<struct\_dec> → struct ID < struct \_Body>

< struct\_Body>→{< struct\_SST>}| ;

<struct\_SST>→<dec><struct\_SST> | <assign\_st><struct\_SST> | <struct\_obj><struct\_SST> | <obj\_decl><struct\_SST>| <func\_st><struct\_SST>| <print><struct\_SST> | ε

**Struct object:**

<struct\_obj>→st ID ID<listA><initA>;

<listA>→,ID<listA> | [<OE>]<listA> |ε

<initA>→ ={<OE><listB>} | ε

<listB>→,<OE><listB> | ε

**Take:**

<take> → take <indexA>;

<indexA>→ (<const>)

**Increment and decrement:**

<Inc\_dec>→ Inc\_Dec <IDC>;

<IDC> → <p> ID <Arr><inc\_dec\_op> | <Inc\_dec\_op> ID<Arr>

<inc\_dec\_op>→ ++ | --

<Arr>→ [<OE> ]<Arr> | ε

**Exceptional Handling:**

<try\_catch >→check{<try\_SST>}<catch><catch1>

<try\_SST>→<dec>< try\_SST>| <If\_else>< try\_SST>|<try\_catch>< try\_SST>| <for\_loop>< try\_SST> | <print>< try\_SST> | <throw>< try\_SST> | <OE>; <try\_SST>| <assign\_st>< try\_SST> |<return>< try\_SST>|<struct\_obj>< try\_SST> | <obj\_decl> < try\_SST>| ε

<catch>→ catch(<catch\_para>){<catch\_SST>}

<catch\_para>→ <obj\_decl> | …

<catch\_SST>→<dec><catch\_SST>|<If\_else><catch\_SST>|<OE>;<catch\_SST>|<assign\_st><catch\_SST> |<return><catch\_SST> | <for\_st><catch\_SST>|<print><catch\_SST>| <throw><catch\_SST> |<struct\_obj><catch\_SST>| <obj\_decl> <catch\_SST> | ε

<catch1>→ <catch><catch1> | ε

**Class:**

<class\_dec> → <sealed> class ID <class\_dec1>;

<class\_dec1>→<inherit> <class\_body>

<inherit> →:<access\_Mod> ID <Multiple\_inherit> | ε

<Multiple\_inherit> → ,<access\_Mod>ID<Multiple\_inherit>| ε

< class\_body> → {< class\_SST>} | ε

<access\_Mod> → public | private | protected | ε

<class\_SST>→<access\_Mod>:<class\_SST>|<dec><class\_SST>| <func\_st><class\_SST>|<constructor><class\_SST>| <destructor> <class\_SST>|<if\_else><class\_SST>|<try\_catch><class\_SST>| <for\_st><class\_SST>|<print><class\_SST>| <class\_dec> <class\_SST> | <struct\_decl> <class\_SST> |<OE>;<class\_SST> |<struct\_obj> <class\_SST> | <obj\_decl><class\_SST> |<assign\_st> <class\_SST> | ε

**Sealed:**

**<sealed> → sealed | ε**

**Interface:**

**<interface>→ interface ID <semi> {<int\_body>}**

**<semi> → : ID | ε**

**<int\_body> → <var> <int\_body> |** <int\_funct>**<int\_body> | ε**

**<var> → <DT1> ID;**

**<int\_funct> → def <DT2> ID (<param>);**

**Object declaration:**

<obj\_decl>→ obj ID ID <N1> ;

<N1> → [OE]<Arr><N2> | (parameter) <N5> | <SA> <OE> | ε

<N2> → <SA><N3> | ε

<N3> → {<OE><N4>}<N4>

<N4> → <or><OE><N4>| {<OE><N4>}<N4>| ε

<N5> → <SA><OE> | ε

<parameter> → <OE><parameter1>

<parameter1> → ,<OE><parameter1> | ε

**Object Calling:(fixed)**

**LEFT FACTORED CFG (obj\_call):**

**(ID wise):**

<obj\_call> → <B’>

<B’> → ID <dot> | <p1> ID <dot>

<para> →<OE><para1>| ε

<para1> →,<OE> <para1> | ε

<dot> → .ID<dot> | ε | [<OE>]<dot> | (<para>)<dot>

**INC\_DEC\_OBJECT :-**

<Inc\_Dec\_obj\_call> → Inc\_Dec <B1>

<B1> → ID <dot><inc\_dec\_op> | <inc\_dec\_op> <X11>|<B2> <inc\_dec\_op>

<B2>→ (ID <dot>) | <p1>ID <dot>

<X11> -> (ID <dot>)| ID<dot>

<para> →<OE><para1> | ε

<para1> →,<OE> <para1> | ε

<dot> → .ID<dot> | ε | [<OE>]<dot> | (<para>)<dot>

**Constructor:**

<constructor> → construct ID<indexC>;

<indexC> → (<param>)<constructor\_Body>

<param>→<dec11><param1> | <struct\_obj><param1> | <obj\_decl> <param1> | ε

<param1> →,<param2> <param1>| ε

<param2> → <dec11> | <struct\_obj> | <obj\_decl>

< constructor\_Body>→

{< constructor\_SST>}| ε

<constructor\_SST>→<dec><constructor\_SST>|<if\_else> <constructor\_SST>|<try\_catch><constructor\_SST>|<for\_st> <constructor\_SST> | <print><constructor\_SST>|<struct\_dec> <constructor\_SST>| <OE>; <constructor\_SST>|<struct\_obj> <constructor\_SST>|<obj\_decl> <constructor\_SST> |<assign\_st> <constructor\_SST> | ε

**Destructor:**

<destructor> → ~ ID<indexD>

<indexD> → () {<destructor\_SST>}

<destructor\_SST>→<dec><destructor\_SST>|<if\_else><destructor\_SST> | <OE>;<destructor\_SST>| <try\_catch><destructor\_SST>| <for\_st><destructor\_SST>|<print><destructor\_SST>|<struct\_obj> <destructor\_SST> |<obj\_decl><destructor\_SST>| <assign\_st> <destructor\_SST>| <struct\_dec><destructor\_SST>|<class\_dec> <destructor\_SST> | ε

**Expression:(Update required)**

<OE> → <AE><OE’>

<OE’> → | <AE> <AE’> | ε

<AE> → <RE><AE’>

<AE’> → & <RE> <AE’> | ε

<RE> → <E> <RE’>

<RE’> → <RO> <E> <RE’> | ε

<E> → <T> <E’>

<E’> → PM <T> <E’> | ε

<T> → <F> <T’>

<T’> → MDM <F> <T’> | ε

<F>→<const> | <Inc\_Dec\_obj\_call>| (<OE>) |!<F> |<obj\_call>