CFGs

<BODY> -> ; |<SST> |<MST>

<MST> -> <SST> <MST> | null

<EXPR> -> ……

<SST> -> <IF\_ELSE> | <SHIFT> | <INC\_DEC\_ST> ; | <DEC> | <OBJ\_DEC> | <LOOP> | <DO\_WHILE> ; | <BREAK> ; | <CONTINUE> ; | <ASSIGN> ;| <TRY\_CATCH\_ST> | <FN\_CALL> | <RET\_ST> ;

Declaration-statement:

<DEC> -> <PROP> id <INIT> <LIST>

<PROP> -> const dt | dt

<INIT> -> = <INIT2> | null

<INIT2> -> id <INIT> | <expression>

<LIST> -> , id <INIT> <LIST> | ;

Assignment-statement:

<ASSIGN> -> id <ASS\_OP><NN>

<NN>-> id <ASSIGN1> | <exp>

<ASSIGN1> -> null | <ASS\_OP> <exp> | <ASS\_OP> id <ASSIGN1>

<ASS\_OP> -> = | cma

Increment-decrement:

<INC\_DEC\_ST> ->

If-else-statement:

//<IF> -> if(<exp>)<body>

<IF\_ELSE > -> if(expr) <body> <OELSE>

OELSE-> else <body> | null

Switch-case-statement:

<SHIFT> -> shift(<exp>) {<STATE>}

<STATE> -> state <EXPR>: <BODY> <STATE> | default:<BODY>| null

**(LF)**

Loop:

<LOOP> -> loop <LT>

<LT> -> <WHILE\_ST> | <FOR\_ST>

While/do-while loop:

<WHILE\_ST> -> till ( <EXPR> ) <body>

<DO\_WHILE> -> do <BODY> till ( <EXPR> )

For-loop

<FOR\_ST> -> thru (dt id in <F1>) <BODY>

<F1> -> id | (<EXPR>,<EXPR>,<EXPR>)

Break-continue:

<BREAK> -> stop <L>

<CONTINUE> -> cont <L>

<L> -> id : | null

Try-catch:

<TRY\_CATCH\_ST> ->

Object-declaration-statement:

<OBJ\_DEC> ->

Function-call-statement:

<FN\_CALL> ->

Return-statement:

<RET\_ST> -> ret <EXPR>