

Team Number: 36

Group 2

Members:

Shreyansh Gandhi

2012A7PS070P

Himanshu Sharma

2012C6PS545P

<program> -> <structures> <declarations> <functions> <main-func>

<structures> -> <structure> <structures> | e

<structure> -> *create id* { <declarations> };

<declarations> -> <type> <id-list> ; <declarations> | <arrayinit> | e
// <P> -> constant | e

<id-list> -> **id** <T> | e

<T>-> , <id-list> | e

<functions> -> *function* <type> <funcname> : <funcsignature> <Y> <functions> | e

<Y> -> <block> | <try-block>

<funcsignature> -> (<args>)

<returntype> -> <type> | void

<args> -> <type> **id** <T2> | void T2

<T2>-> , <args> | e

<funcname> -> **id**

<main-func> -> *main* <block>

BLOCK

<block> -> {<statements>}

<try-block> -> *try* <block> *catch* <block> *finally* <block>

STATEMENTS

<statements>-> <statement> <statements> | e

<statement> -> <assignstat> | <declarations> | <returnstat> | <ifstat>| <iterativestat> |
<instat>| <outstat> | *end*; | *next*; | <functioncall> ;

<assignstat> -> <arraystmt> := <Exp>;

<functioncall> -> *call* **id**(<toSend>)
<toSend> -> <arraystmt> <S> | <literal> <S>
<S> -> ,<toSend> | e

<return stat>: *return* <to return>;
<to return>: <arraystmt> | <literal>

<ifstat> -> *if*(<condExp>){ <statements> } <elsestat>
<elsestat> -> *else* { <statements> } | e

<iterativestat> -> *while* (<condExp>){ <statements> }

<instat> -> *input* >> **id**;
<outstat> -> *output* << **id**;

<arrayinit> -> <arraypart> **id** ;
<arraypart> -> *array*(<types>)[<arithmeticexp>] <Z>
<Z> -> [<arithmeticexp>] <Z> | e
<arraystmt> -> **id** <X> | # **id** <X>
<X> -> [<arithmeticexp>]<X> | .id | e

<id> -> TK_Identifier

<Exp> -> <ORexp> | <functioncall>

<condExp> -> <ORexp>

<ORexp> -> <ANDexp> <F>
<F> -> || <ANDexp> <F> | e

<ANDexp> -> <equalityexp><G>
<G> -> &&<equalityexp><G> | e

<equalityexp> -> <relationalexp><H>
<H> -> <equalOp> <relationalexp> | e

<equalOp> -> == | !=

$\langle \text{relationalexp} \rangle \rightarrow \langle \text{arithmeticexp} \rangle \langle J \rangle$
 $\langle J \rangle \rightarrow \langle \text{relOp} \rangle \langle \text{arithmeticexp} \rangle \mid e$
 $\langle \text{relOp} \rangle \rightarrow > \mid < \mid \leq \mid \geq$

$\langle \text{arithmeticexp} \rangle \rightarrow \langle \text{addexp} \rangle$

$\langle \text{addexp} \rangle \rightarrow \langle \text{mulexp} \rangle \langle B \rangle$
 $\langle B \rangle \rightarrow + \langle \text{mulexp} \rangle \langle B \rangle \mid - \langle \text{mulexp} \rangle \langle B \rangle \mid e$

$\langle \text{mulexp} \rangle \rightarrow \langle \text{bitexp} \rangle \langle C \rangle$
 $\langle C \rangle \rightarrow * \langle \text{mulexp} \rangle \langle C \rangle \mid / \langle \text{mulexp} \rangle \langle C \rangle \mid \% \langle \text{mulexp} \rangle \langle C \rangle \mid e$

$\langle \text{bitexp} \rangle \rightarrow \langle \text{unaryexp} \rangle \langle D \rangle$
 $\langle D \rangle \rightarrow \langle \text{bitOp} \rangle \langle \text{bitexp} \rangle \langle D \rangle \mid e$
 $\langle \text{bitOp} \rangle \rightarrow \& \mid \mid$

$\langle \text{unaryexp} \rangle \rightarrow \langle \text{notexp} \rangle \langle K \rangle$ //only post increment or decrement
 $\langle K \rangle \rightarrow \text{inc} \mid \text{dec} \mid e$

$\langle \text{notexp} \rangle \rightarrow \langle \text{notOp} \rangle \langle \text{simple} \rangle \mid \langle \text{simple} \rangle$
 $\langle \text{notOp} \rangle \rightarrow !$

$\langle \text{simple} \rangle \rightarrow \langle \text{literal} \rangle \mid \langle \text{arraystmt} \rangle \mid (\langle \text{Exp} \rangle)$
 $\langle \text{literal} \rangle \rightarrow \langle \text{integerliteral} \rangle \mid \langle \text{booleanliteral} \rangle \mid \langle \text{charliteral} \rangle$

$\langle \text{booleanliteral} \rangle \rightarrow \text{true} \mid \text{false}$
 $\langle \text{integerliteral} \rangle \rightarrow \text{TK_Num_Integer}$
 $\langle \text{charliteral} \rangle \rightarrow \text{TK_character_literal}$

$\langle \text{type} \rangle \rightarrow \text{int} \mid \text{char} \mid \text{boolean} \mid (\text{user defined data type's id})$

Changes

We have removed the $\langle P \rangle$ non-terminal and constant terminal.

$\langle P \rangle \rightarrow \text{constant} \mid e$

Also we made a slight change in $\langle \text{functioncall} \rangle$

earlier:	$\langle \text{statement} \rangle$	\rightarrow	$\langle \text{functioncall} \rangle$	//partial
	$\langle \text{functioncall} \rangle$	\rightarrow	$\text{call id}(\langle \text{toSend} \rangle);$	
now:	$\langle \text{statement} \rangle$	\rightarrow	$\langle \text{functioncall} \rangle;$	//partial
	$\langle \text{functioncall} \rangle$	\rightarrow	$\text{call id}(\langle \text{toSend} \rangle)$	

