Assignment 1

Question 1. Group Details

Name	**Roll No.**	**Email ID**
Mukul Chaturvedi	150430	mukulck@iitk.ac.in
Vinayak Soni	150802	vinayaks@iitk.ac.in

Question 3.
Source language S of our compiler - Cool
Implementation language I of our compiler - Python
Target language T of our compiler - x86

We have borrowed the grammar for our language from
http://www.cs.virginia.edu/~cs415/cool-manual/cool-manual.html

Keywords :
class , inherits , case , of , esac, if , fi , then , else , true , false , new , isvoid , not , let , in , while , for , do , loop , pool , break , continue , return

Lexical Rules for our language:

<Digit>	0|1|....|9
<Integer>	[[<Digit>]]+
<Letter>	<Small_letter>| <Capital_letter>
<Small_letter>	a|b|....|z
<Capital_letter>	A|B|...|Z
<Id>	<Small_letter> [[<Letter> | <Digit> | _]]*
<Type>	<Capital_letter> [[<Letter> | <Digit> | _]]*
<Strings>	[[ch]]+ ch is any ascii character

Syntax :

<Compilation_unit>	[<Package_declaration>] [<Import_declarations>] [<Program>]
<Package_declaration>	package <Package_name> ;
<Package_name>	<Package_name>.<ID> | <ID>
<Import_declarations>	<Import_declarations> <Import_declaration> | <Import_declaration>
<Import_declaration>	import <Package_name>; | import <Package_name>.*
<Program>	[[<Class>;]]*
<Class>	class <TYPE> <Inheritance> { <Features_list_opt> }
<Inheritance>	inherits <TYPE> | <Empty>
<Features_list_opt>	<Features_list> | <Empty>
<Features_list>	<Features_list> <Feature> ; | <Feature> ;
<Feature>	<ID> (<Formal_params_list_opt>) : <TYPE> { <Expression> } | <Formal>
<Formal_params_list_opt>	<Formal_params_list> | <Empty>
<Formal_params_list>	<Formal_params_list> , <Formal_param> | <Formal_param>
<Formal_param>	<ID> : <TYPE> ['['']']
<Formal>	<ID> : <TYPE> ['[' [<Expression>] ']'] <- <Expression> | <ID> : <TYPE> ['[' <Expression> ']']
<Expression>	<ID> <- <Expression> | <Expression>.<ID>(<Arguments_list_opt>) | <Expression>@<TYPE>.<ID>(<Arguments_list_opt>) | <Conditionals> | <Loops> | <Block_Expression> | <Let_Expression> | new <TYPE> | isvoid <Expression> | <Return_statement> | <Break_statement> | <Continue_statement> | <Expression> + <Expression> | <Expression> - <Expression> | <Expression> * <Expression> | <Expression> / <Expression> | ~ <Expression>

	| <Expression> < <Expression> | <Expression> <= <Expression> | <Expression> = <Expression> | <Expression> > <Expression> | <Expression> >= <Expression> | <Expression> && <Expression> | <Expression> || <Expression> | <Expression> & <Expression> | <Expression> | <Expression> | <Expression> ^ <Expression> | not <Expression> | (<Expression>) | SELF | <ID> | <ID> '[' <Expression> ']' | '[' [[<Expression>]]+ ']' | true | false | SELF_TYPE | <INTEGER> | <STRING>
<Conditionals>	<Case> | <If_then_else>
<Loops>	<While> | <For> | <Do_while>
<Arguments_list_opt>	<Arguments_list> | <Empty>
<Arguments_list>	<Arguments_list> , <Expression> | <Expression>
<Case>	case <Expression> of <Actions> esac
<Actions>	<Action> | <Action> <Actions>
<Action>	<ID> : <TYPE> => <Expression>
<If_then_else>	if <Expression> then <Expression> else <Expression> fi
<While>	while <Expression> loop <Expression> pool
<For>	for (<Expression> ; <Expression> ; <Expression>) loop <Expression> pool
<Do_while>	do loop <Expression> pool while <Expression>
<Break_statement>	break
<Continue_statement>	continue
<Return_statement>	return

<Block_Expression>	{ <Block_list> }
<Block_list>	<Block_list> <Expression> ; | <Expression> ;
<Let_Expression>	let <Formal> [[,<Formal>]]* in <Expression>
<Empty>	

White Space :- consists of any sequence of the characters: blank (ascii 32), \n (newline, ascii 10), \f (form feed, ascii 12), \r (carriage return, ascii 13), \t (tab, ascii 9), \v (vertical tab, ascii 11)

Comments

There are two forms of comments in Cool. Any characters between two dashes "--" and the next newline (or EOF, if there is no next newline) are treated as comments. Comments may also be written by enclosing text in between two *. The latter form of comment may be nested but may not contain EOF. Comments cannot cross file boundaries.

New Constructs added
1) Import statements are added same as java import function , to import code from libraries .
Syntax is import <Package name >
2) Package declaration used to name a program. It is helpful in importing it in other programs
Syntax is package <package name>
3)Arrays can be created like below and elements can be accessed like in C .

```
a : Int[4] <- [1,2,34,5]
```

4) Break , Continue and Return statements as in C .
5) "For" and "Do while" loops
6) '>' , '>=' relational operators
7)Boolean operators : '&&' , '\\'
8)Bitwise operators : & (and), | (or) , ^ (xor)