CP471

Assignment 2

Austin Bursey | 160165200

Due: Feb 14th 2020

1 Grammar

<program>::= <fdecls> <declarations> < *statement_seq* >.
<fdecls> ::= <fdec> ;<fdecls> |ϵ

<fdec> ::=def <type> <fname> (<params>) <declarations> < *statement_seq* >
fed
<params> ::= <type> <var> <params-rest>| ϵ
<params-rest> ::= , <params> | ϵ |
<fname>::= <id>

<declarations>::= <decl>; <declarations>| ϵ|

<decl>::= <type> <varlist>
<type>::= int | double
<varlist>::= <var> <varlist-rest>
<varlist-rest>::= , <varlist>| ϵ

< *statement_seq* >::= <statement> < *statement_seq − rest* >
< *statement_seq − rest* >::= ; < *statement_seq* >| ϵ |

<statement>::= <var> = <expr> |
if <bexpr> then < *statement_seq* > <statement-rest>fi |
while <bexpr> do < *statement_seq* > od |
print <expr>|
return <expr>|ϵ

<statement-rest>::= else < *statement_seq* >| ϵ

<expr>::= <term> <expr-rest>
<expr-rest>::= + <term> <expr-rest> | - <term> <expr-rest>| ϵ

<term>::= <factor> <term-rest>
<term-rest>::= *<factor> <term-rest>|/<factor> <term-rest>|% <factor>
<term-rest>|ϵ
<factor>::= <id><factor-rest>| <number>| (<expr>)
<factor-rest>::= <var-rest>|(<exprseq>)
<exprseq>::= <expr> <exprseq-rest>|ϵ
<exprseq-rest>::= , <exprseq> | ϵ |

<bexpr>::= <bterm> <bexpr-rest>
<bexpr-rest>::= or <bterm> <bexpr-rest>| ϵ

<bterm>::= <bfactor> <bterm-rest>
<bterm-rest>::= and <bfactor> <bterm-rest> | ϵ

<bfactor>::= (<bfactor-rest >)| not <bfactor> |
<bfactor-rest >::= <bexpr>| <expr> <comp> <expr>
<bfactor-rest-rest> ::=
<comp>::= < | > | == | <= | >= |< >

<var>::= <id> <var-rest>
<var-rest>::= [<expr>]| ϵ

<letter>::= a | b | c | ... | z
<digit>::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
<id>::= <letter> <id-rest>
<id-rest>::= <letter> <id-rest>| <digit> <id-rest> | ϵ

<number>::= <integer> | <double>

<integer>::= <digit> <int shortcut>
<int shortcut>::= <e-int>|<digit> <int shortcut>|. <lead-off> <e-int>
|ϵ

<e-int>::= e<e-rest-int> <lead-off>|ϵ
<double>::= <digit> <double shortcut>
<double shortcut>::=. <lead-off> <e-double>| <digit> <double shortcut> |
<e-double>| <lead-off>| ϵ

<e-double>::= e-<e-rest-int> <lead-off>|ϵ
<lead-off>:= <digit> <lead-off-rest>
<lead-off-rest>:= <lead-off>| ϵ

<e-rest-int>:= +|ϵ

2 First

first(<program>) = {def, int, double,a , b , c , ... , z, if,while,print,return, . }
first(<fdecls>)= first(<fdec>) = {def,ϵ }

first(<declarations>)= { int , double,ϵ}

first(<params>) = { int , double, ϵ}
first(<type>) = first(<decl>) = { int , double}

first(<params-rest>) = {',',ϵ}

first(<fname>) = first(<id>) = first(<letter>) = first(<varlist>) = first(<var>)=
{ a , b , c , ... , z}

first(<comp>) = {<, >, ==, <=, >=,< > }

first(<varlist-rest>) = {',' , ϵ}

first(< *statement_seq* >) = first(<statement>) = {if, while, print, return, a , b , c , ... , z, ϵ }

first(< $statement\_seq - rest$ >) = {; , ϵ}

first(<statement-rest>)= {else , ϵ }

first(<exprseq>)={ (,1,2,3,4,5,6,7,8,9,0, a , b , c , ... , z, ϵ}
first(<expr>) = first(<term>) = first(<factor>)=first(<factor-rest>)= { (,1,2,3,4,5,6,7,8,9,0, a , b , c , ... , z}

first(<expr-rest>) = {+, -, ϵ}

first(<term-rest>) = {*, /, %, ϵ}

first(<exprseq-rest>)= {',', ϵ}

first(<bexpr>) = first(<bterm>)= first(<bfactor>) = {(,not }

first(<bexpr-rest>)= {or,ϵ }

first(<bterm-rest>)= {and,ϵ }

first(<bfactor-rest >)= {(,not,a , b , c , ... , z ,1,2,3,4,5,6,7,8,9,0}

first(<var-rest>)= {[,ϵ }

first(<digit>)= first(<number>) =first(<integer>)=first(<double>)=first(<lead-off>)= {1,2,3,4,5,6,7,8,9,0}

first(<id-rest>)= {a , b , c , ... , z ,1,2,3,4,5,6,7,8,9,0,ϵ }

first(<int shortcut>)= {e,1,2,3,4,5,6,7,8,9,0, . , ϵ }

first(<double shortcut>)= {.,1,2,3,4,5,6,7,8,9,0,e- , ϵ}

first(<lead-off-rest>)= {1,2,3,4,5,6,7,8,9,0,ϵ}

first(<e-rest-int>)= {+,ϵ}
first(<e-int>)= {e,ϵ}
first(<e-double>)= {e-,ϵ}

3 Follow

follow(<program>)= {.}

follow(<fdecls>)= { int , double,.}

follow(<declarations>)= {if, while, print, return, a , b , c , ... , z, . }

follow(<fdec>) = {;}

follow(<statement_seq>)=follow(statement_seq-rest) = { . , fed , fi ,od,else }

follow(<type>)= { a , b , c , ... , z}

follow(<fname>)= { (}

follow(<params>)= follow(<params-rest>) = {) }

follow(<var>) = {"," ,) , ; ,*,/,%,+, -,;,',',),], <,>,==,<=,>=,<> , = }

follow(<decl>) ={;}

follow(<varlist>) = follow(<varlist-rest>)= {;,) }

follow(<id>)=follow(<id-rest>) {(, [, "," ,) , ; ,*,/,%,+, -,], <,>,==,<=,>=,<> , =, . , fed , fi ,od,else }

follow(<statement>)= {;, . , fed , fi ,od,else}

follow(<expr>)= follow(<expr-rest>)= {;,',',),], <,>,==,<=,>=,<>, . , fed , fi ,od,else }

follow(<term>)=follow(<term-rest>)= {+, -,;,',',),], <,>,==,<=,>=,<> , . , fed , fi ,od,else}

follow(<factor>)=follow(<factor-rest>) =follow(<number>)= {*,/,%,+, -,;,',',),], <,>,==,<=,>=,<> , . , fed , fi ,od,else }

follow(<exprseq>)=follow(<exprseq-rest>)={) }

follow(<bexpr>)=follow(<bexpr-rest>)={then, do ,) }

follow(<bterm>)=follow(<bterm-rest>)={then, do ,) ,or }

follow(<bfactor>) ={then, do ,) ,or , and}

follow(<bfactor-rest>) = {)}

follow(<comp>)={ (,1,2,3,4,5,6,7,8,9,0, a , b , c , ... , z}

follow(<letter>)= {(, [, "," ,) , ;, ,*,/,%,+, -,], <,>,==,<=,>=,<> , =
,a , b , c , ... , z ,1,2,3,4,5,6,7,8,9,0, . , fed , fi ,od,else }

follow(<double>)=follow(<e-double>)=follow(<double-shortcut>)=follow(<integer>)=follow(<int-shortcut>)=follow(<e-int>)={*,/,%,+, -,;,',',),], <,>,==,<=,>=,<> , . , fed
, fi ,od,else }

follow(<lead-off>)=follow(<lead-off-rest>)={*,/,%,+, -,;,',',),], <,>,==,<=,>=,<>,e,1,2,3,4,5,6,7,8,9,0
}

follow(<digit>)={*,/,%,+, -,;,',',),], <,>,==,<=,>=,<>,e,1,2,3,4,5,6,7,8,9,0,
. , fed , fi ,od,else }

follow(<e-rest-int>)={1,2,3,4,5,6,7,8,9,0,(, [, "," ,) , ; }