<start> —> <expr> EOF

<expr> -> <term> <exprp>

<exprp> —> + <term> <exprp> | - <term> <exprp | ε

<term> —> <fact> <termp>

<termp> —> * <fact> <termp> | / <fact> <termp> | ε

<fact> —> (<expr>) | NUM

First (<start>) = First (<expr>) = {(, NUM}

First (<expr>) = First (<term>) = {(, NUM}

First (<exprp) = { +, - }

First (<term>) = First (<fact>) = {(, NUM}

First (<termp>) = {*, /}

First (<fact>) = {(, NUM}

$ ∈ Follow (<start>)

{EOF} ∈ Follow (<expr>)

Follow (expr) ⊆ Follow (exprp)

{+, -} ∈ Follow (<term>)

Follow (<expr>) ⊆ Follow (<term>)

Follow (<exprp>) ⊆ Follow (<term>)

. . . .

Follow (<term>) ⊆ Follow (<termp>)

{*, /} ∈ Follow (<fact>)

Follow (<term>) ⊆ Follow (<fact>)

Follow (<termp>) ⊆ Follow (<fact>)

. . . .

{)} ∈ Follow (<expr>)

A —> α B β

<start> —> <expr> EOF

<expr> -> <term> <exprp>

<expr> -> <term> <exprp>
" NULL

<exprp> -> + <term> <exprp>
" NULL

<exprp> -> + <term> <exprp>

<term> -> <fact> <termp>

<term> -> <fact> <termp>
" NULL

<termp> -> * <fact> <termp>
" NULL

<termp> -> * <fact> <termp>

<fact> -> (<expr>)

X	Follow(X)
<start>	$
<expr>	EOF ,)
<exprp>	EOF ,)
<term>	EOF, +, - ,)
<termp>	EOF, +, - ,)
<fact>	EOF, +, -, *, /,)

A -> α	Guida (A -> α)
<start> -> <expr> EOF	First(<expr>) = { (, NUM }
<expr> -> <term> <exprp>	First(<term>) = { (, NUM }
<exprp> -> + <term> <exprp>	{ + }
<exprp> -> - "	{ - }
<exprp> -> ε	Follow(<exprp>) = { EOF,) }
<term> -> <fact> <termp>	First(<fact>) = { (, NUM }
<termp> -> * <fact> <termp>	{ * }
<termp> -> / "	{ / }
<termp> -> ε	Follow(<termp>) = { EOF, +, -,) }
<fact> -> (<expr>)	{ (}
<fact> -> NUM	{ NUM }

ES funzionanti

NUM + (NUM) EOF

(NUM) EOF

<Prog> = P

<statlist> = Sl

<statlistp> = Slp NULL (Slp)

<stat> = S

<whenlist> = Wl

<whenlistp> = Wlp NULL (Wlp)

<when item> = Wi

<bexpr> = B

<expr> = E

<exprlist> = El

<exprlistp> = Elp NULL (Elp)

First (x)

First (P) = First (Sl) = { =, print, read, cond, while, { }

First (Sl) = First (S) = { =, print, read, cond, while, { }

First (Slp) = { ; }

First (S) = { =, print, read, cond, while, { }

First (Wl) = First (Wi) = { when }

First (Wlp) = First (Wi) = { when }

First (Wi) = { when }

First (B) = { RELOP }

First (E) = { +, -, *, /, NUM, ID }

First (El) = First (E) = { +, -, *, /, NUM, ID }

First (Elp) = First (E) = { +, -, *, /, NUM, ID }

$ ∈ Follow (P)

{EOF} ∈ Follow (SL)

Follow (SL) ⊆ Follow (SLp)

{;} = first (SLp) ∈ Follow (S)

Follow (SL) ⊆ Follow (S)

~~{;} = first (SLp) ∈ follow (S)~~
follow (SLp) ⊆ Follow (S)
~~follow (SLp) ⊆ follow (SLp)~~

Follow (S) ⊆ Follow (E)

{)} ∈ Follow (EL)

{else} ∈ Follow (WL)

~~follow (S) ⊆ Follow (S)~~

{)} ∈ Follow (B)

~~follow (S) ∈ Follow (S)~~

{}} ∈ Follow (SL)

Follow (WL) ⊆ Follow (WLp)

{when} = First (WLp) ∈ Follow (Wi)

Follow (WL) ⊆ Follow (Wi)

~~follow (WLp) ⊆ Follow (WLp)~~

~~{when} = First (WLp) ∈ Follow (Wi)~~

Follow (WLp) ⊆ Follow (Wi)

~~{)} ∈ Follow (B)~~

Follow (Wi) ⊆ Follow (S)

{+, -, *, /, NUM, ID} = First (E) ⊆ Follow (E)

Follow (B) ⊆ Follow (E)

{)} ∈ Follow (EL)

~~{+, -, *, /, NUM, ID} = First (E) ⊆ Follow (E)~~

~~follow (E) ⊆ Follow (E)~~

Follow (EL) ⊆ Follow (ELp)
Follow (EL) ⊆ follow (E)
~~follow (ELp) ⊆ Follow (ELp)~~
Follow (ELp) ⊆ ~~Follow~~ follow (E)

A	→	α	B	B
P	→		SL	EOF
SL	→	S	SLp	
SL	→		S	SLp | NULL
SLp	→	;	S	SLp | NULL
SLp	→	; S	SLp	
S	→	=ID	E	
S	→	print(	EL	)
S	→	cond	WL	else S
S	→	cond WL else	S	
S	→	while(	B	) S
S	→	while(S B)		
S	→	{	SL	}
WL	→	Wi	WLp	
WL	→		Wi	WLp | NULL
WLp	→	Wi	WLp	
WLp	→		Wi	WLp | NULL
Wi	→	when(	B	) do S
Wi	→	when (B) do	S	
B	→	Relop	E	E
B	→	Relop E	E	
E	→	+ | * |	(EL	)
E	→	- | /	E E	E E
E	→	- | /	E E	
EL	→	E	ELp E	ELp | NULL
ELp	→	E	ELp E	ELp | NULL

X	Follow(X)
P	$
Sl	EOF }
Slp	EOF }
S	EOF, i, else, }, when
Wl	else
Wlp	else
Wi	else when
B	)
E	EOF, ;,), else, }, when, +, -, *, /, NUM, ID
El	)
Elp	)

A -> α	Guida (A -> α)
P -> Sl EOF	First(Sl) = {i, print, read, cond, while, {}
Sl -> S Slp	First(S) = {i, print, read, cond, while, {}
Slp -> ; S Slp	{ ; }
Slp -> ε	Follow(Slp) = { EOF, } }
S -> = ID E	{ = }
S -> print (El)	{ print }
S -> read (ID)	{ read }
S -> cond (Wl) else S	{ cond }
S -> while (B) S	{ while }
S -> { Sl }	{ { }
Wl -> Wi Wlp	First(Wi) = { when }
Wlp -> Wi Wlp	First(Wi) = { when }
Wlp -> ε	Follow(Wlp) = { else }
Wi -> when (B) do S	{ when }
B -> Relop (E) (E)	{ relop }
E -> + (El)	{ + }

E —> - E E

E —> *(El)

E —> / E E

E —> NUM

E —> ID

El —> E Elp

Elp —> E Elp

Elp —> ε

{ - }

{ * }

{ / }

{ NUM }

{ ID }

First(E) = { +, -, *, /, NUM, ID }

First(E) = { +, -, *, /, NUM, ID }

Follow(Elp) = {) }