

tests

Justin Chin

May 1, 2018

Test1

```
! this is comment for this sample code which converts Fahrenheit into Celcius !
```

```
function convert$ [fahr:int]
{
    return 5 * (fahr -32) / 9;
}
```

```
%%
int    low,  high, step$;      ! declarations !
```

```
put (9.0);
get (low, high, step$);
while (low <  high )
```

```
{  put (low);
    put (convert$ (low));
    low =  low  +  step$;
}
```

Results

<Rat18S> ::= <OptFunctionDefinitions> %% <OptDeclarationlist> <StatementList>

<OptFunctionDefinitions> ::= <FunctionDefinitions> | <Empty>

<FunctionDefinitions> ::= <Function> <FDPrime>

Token: Keyword Lexeme: "function"

<Function> ::= function <Identifier> [<OptParameterList>] <OptDeclarationList> <Body>

Token: Identifier Lexeme: "convert\$"

Token: LBracket Lexeme: "["

<OptParameterList> ::= <ParamaterList> | <Empty>

<ParamaterList> ::= <Parameter> <ParameterListPrime>

<Parameter> ::= <IDs> : <Qualifier>

Token: Identifier Lexeme: "fahr"

<IDs> ::= <Identifier> <IDsPrime>

<IDsPrime> ::= <Empty>

Token: Colon Lexeme: ":"

Token: Keyword Lexeme: "int"

<Qualifier> ::= int
 <ParameterListPrime> ::= <Empty>
 Token: RBracket Lexeme: "]"
 <OptDeclarationList> ::= <Empty>
 Token: LBrace Lexeme: "{"
 <Body> ::= { <StatementList> }
 <StatementList> ::= <Statement> <StatementListPrime>
 <Statement> ::= <Return>
 Token: Keyword Lexeme: "return"
 <Return> ::= return <ReturnPrime>
 <Expression> ::= <Term> <ExpressionPrime>
 <Term> ::= <Factor> <TermPrime>
 <Factor> ::= - <Primary> | <Primary>
 Token: Int Lexeme: "5"
 <Primary> ::= <Integer>
 Token: Times Lexeme: "*"
 <TermPrime> ::= * <Factor> <TermPrime>
 <Primary> ::= (<Expression>)
 Token: LParen Lexeme: "("
 Token: Identifier Lexeme: "fahr"
 <Primary> ::= <Identifier>
 <TermPrime> ::= <Empty>
 Token: Minus Lexeme: "-"
 <ExpressionPrime> ::= - <Term> <ExpressionPrime>
 Token: Int Lexeme: "32"
 <Primary> ::= <Integer>
 <TermPrime> ::= <Empty>
 <ExpressionPrime> ::= <Empty>
 Token: RParen Lexeme: ")"
 Token: Div Lexeme: "/"
 <TermPrime> ::= / <Factor> <TermPrime>
 Token: Int Lexeme: "9"
 <Primary> ::= <Integer>
 <TermPrime> ::= <Empty>
 <ExpressionPrime> ::= <Empty>
 <ReturnPrime> ::= <Expression> ;
 Token: Semicolon Lexeme: ";"
 <StatementListPrime> ::= <Empty>
 Token: RBrace Lexeme: "}"
 <FDPrime> ::= <Empty>
 Token: EndOfDefs Lexeme: "%%"
 <OptDeclarationList> ::= <DeclarationList>
 <DeclarationList> ::= <Declaration> <DeclarationListPrime>
 <Declaration> ::= <Qualifier> <IDs>
 Token: Keyword Lexeme: "int"
 <Qualifier> ::= int
 Token: Identifier Lexeme: "low"
 <IDs> ::= <Identifier> <IDsPrime>

Token: Comma Lexeme: ","
 $\langle \text{IDsPrime} \rangle ::= , \langle \text{IDs} \rangle$
 Token: Identifier Lexeme: "high"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 Token: Comma Lexeme: ","
 $\langle \text{IDsPrime} \rangle ::= , \langle \text{IDs} \rangle$
 Token: Identifier Lexeme: "step\$"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 $\langle \text{IDsPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: Semicolon Lexeme: ";"
 $\langle \text{DeclarationListPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{Statement} \rangle ::= \langle \text{Print} \rangle$
 Token: Keyword Lexeme: "put"
 $\langle \text{Print} \rangle ::= \text{put} (\langle \text{Expression} \rangle) ;$
 Token: LParen Lexeme: "("
 Token: Real Lexeme: "9.0"
 $\langle \text{Primary} \rangle ::= \langle \text{Real} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: RParen Lexeme: ")"
 Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{StatementList} \rangle$
 $\langle \text{Statement} \rangle ::= \langle \text{Scan} \rangle$
 Token: Keyword Lexeme: "get"
 $\langle \text{Scan} \rangle ::= \text{get} (\langle \text{IDs} \rangle) ;$
 Token: LParen Lexeme: "("
 Token: Identifier Lexeme: "low"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 Token: Comma Lexeme: ","
 $\langle \text{IDsPrime} \rangle ::= , \langle \text{IDs} \rangle$
 Token: Identifier Lexeme: "high"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 Token: Comma Lexeme: ","
 $\langle \text{IDsPrime} \rangle ::= , \langle \text{IDs} \rangle$
 Token: Identifier Lexeme: "step\$"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 $\langle \text{IDsPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: RParen Lexeme: ")"
 Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{StatementList} \rangle$
 $\langle \text{Statement} \rangle ::= \langle \text{While} \rangle$
 Token: Keyword Lexeme: "while"
 $\langle \text{While} \rangle ::= \text{while} (\langle \text{Condition} \rangle) \langle \text{Statement} \rangle$
 Token: LParen Lexeme: "("
 $\langle \text{Condition} \rangle ::= \langle \text{Expression} \rangle \langle \text{Relop} \rangle \langle \text{Expression} \rangle$
 Token: Identifier Lexeme: "low"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$

$\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: Less Lexeme: "<"
 $\langle \text{Relop} \rangle ::= <$
 Token: Identifier Lexeme: "high"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: RParen Lexeme: ")"
 $\langle \text{Statement} \rangle ::= \langle \text{Compound} \rangle$
 Token: LBrace Lexeme: "{"
 $\langle \text{Compound} \rangle ::= \{ \langle \text{StatementList} \rangle \}$
 $\langle \text{Statement} \rangle ::= \langle \text{Print} \rangle$
 Token: Keyword Lexeme: "put"
 Token: LParen Lexeme: "("
 Token: Identifier Lexeme: "low"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: RParen Lexeme: ")"
 Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{StatementList} \rangle$
 $\langle \text{Statement} \rangle ::= \langle \text{Print} \rangle$
 Token: Keyword Lexeme: "put"
 Token: LParen Lexeme: "("
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle (\langle \text{IDs} \rangle)$
 Token: Identifier Lexeme: "convert\$"
 Token: LParen Lexeme: "("
 Token: Identifier Lexeme: "low"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 $\langle \text{IDsPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: RParen Lexeme: ")"
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: RParen Lexeme: ")"
 Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{StatementList} \rangle$
 $\langle \text{Statement} \rangle ::= \langle \text{Assign} \rangle$
 Token: Identifier Lexeme: "low"
 $\langle \text{Assign} \rangle ::= \langle \text{Identifier} \rangle = \langle \text{Expression} \rangle ;$
 Token: Assign Lexeme: "="
 Token: Identifier Lexeme: "low"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: Plus Lexeme: "+"
 $\langle \text{ExpressionPrime} \rangle ::= + \langle \text{Term} \rangle \langle \text{ExpressionPrime} \rangle$
 Token: Identifier Lexeme: "step\$"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$

```

<ExpressionPrime> ::= <Empty>
Token: Semicolon Lexeme: ";"
<StatementListPrime> ::= <Empty>
Token: RBrace Lexeme: "}"
<StatementListPrime> ::= <Empty>

```

Test2

```

! squares and returns the argument !

function square [x:int] {
    return x * x;
}

function timesFiveMinusTwo$ [x:int] {
    return 5 * x - 2;
}

%%

int x, y, z, x2$, y2$;

get (x, y, z);

put (timesFiveMinusTwo$ (x));
while (y < z)
    { put (y);
      put (square (y));
      y = y + 1;
    }

x2$ = square(x);
y2$ = square(y);

if (x2$ > y2$)

    return true;

endif

```

Results

```

<OptFunctionDefinitions> ::= <FunctionDefinitions> | <Empty>
<FunctionDefinitions> ::= <Function> <FDPrime>
Token: Keyword Lexeme: "function"
<Function> ::= function <Identifier> [ <OptParameterList> ] <OptDeclarationList> <Body>

```

Token: Identifier Lexeme: "square"
 Token: LBracket Lexeme: "["
 $\langle \text{OptParameterList} \rangle ::= \langle \text{ParamaterList} \rangle \mid \langle \text{Empty} \rangle$
 $\langle \text{ParamaterList} \rangle ::= \langle \text{Parameter} \rangle \langle \text{ParameterListPrime} \rangle$
 $\langle \text{Parameter} \rangle ::= \langle \text{IDs} \rangle : \langle \text{Qualifier} \rangle$
 Token: Identifier Lexeme: "x"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 $\langle \text{IDsPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: Colon Lexeme: ":"
 Token: Keyword Lexeme: "int"
 $\langle \text{Qualifier} \rangle ::= \text{int}$
 $\langle \text{ParameterListPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: RBracket Lexeme: "]"
 $\langle \text{OptDeclarationList} \rangle ::= \langle \text{Empty} \rangle$
 Token: LBrace Lexeme: "{"
 $\langle \text{Body} \rangle ::= \{ \langle \text{StatementList} \rangle \}$
 $\langle \text{StatementList} \rangle ::= \langle \text{Statement} \rangle \langle \text{StatementListPrime} \rangle$
 $\langle \text{Statement} \rangle ::= \langle \text{Return} \rangle$
 Token: Keyword Lexeme: "return"
 $\langle \text{Return} \rangle ::= \text{return} \langle \text{ReturnPrime} \rangle$
 $\langle \text{Expression} \rangle ::= \langle \text{Term} \rangle \langle \text{ExpressionPrime} \rangle$
 $\langle \text{Term} \rangle ::= \langle \text{Factor} \rangle \langle \text{TermPrime} \rangle$
 $\langle \text{Factor} \rangle ::= - \langle \text{Primary} \rangle \mid \langle \text{Primary} \rangle$
 Token: Identifier Lexeme: "x"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 Token: Times Lexeme: "*"
 $\langle \text{TermPrime} \rangle ::= * \langle \text{Factor} \rangle \langle \text{TermPrime} \rangle$
 Token: Identifier Lexeme: "x"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ReturnPrime} \rangle ::= \langle \text{Expression} \rangle ;$
 Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: RBrace Lexeme: "}"
 $\langle \text{FDPrime} \rangle ::= \langle \text{FunctionDefinitions} \rangle$
 Token: Keyword Lexeme: "function"
 Token: Identifier Lexeme: "timesFiveMinusTwo\$"
 Token: LBracket Lexeme: "["
 Token: Identifier Lexeme: "x"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 $\langle \text{IDsPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: Colon Lexeme: ":"
 Token: Keyword Lexeme: "int"
 $\langle \text{Qualifier} \rangle ::= \text{int}$
 $\langle \text{ParameterListPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: RBracket Lexeme: "]"
 $\langle \text{OptDeclarationList} \rangle ::= \langle \text{Empty} \rangle$

Token: LBrace Lexeme: "{"
 $\langle \text{Body} \rangle ::= \{ \langle \text{StatementList} \rangle \}$
 $\langle \text{Statement} \rangle ::= \langle \text{Return} \rangle$
 Token: Keyword Lexeme: "return"
 Token: Int Lexeme: "5"
 $\langle \text{Primary} \rangle ::= \langle \text{Integer} \rangle$
 Token: Times Lexeme: "*"
 $\langle \text{TermPrime} \rangle ::= * \langle \text{Factor} \rangle \langle \text{TermPrime} \rangle$
 Token: Identifier Lexeme: "x"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: Minus Lexeme: "-"
 $\langle \text{ExpressionPrime} \rangle ::= - \langle \text{Term} \rangle \langle \text{ExpressionPrime} \rangle$
 Token: Int Lexeme: "2"
 $\langle \text{Primary} \rangle ::= \langle \text{Integer} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ReturnPrime} \rangle ::= \langle \text{Expression} \rangle ;$
 Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: RBrace Lexeme: "}"
 $\langle \text{FDPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: EndOfDefs Lexeme: "%%"
 $\langle \text{OptDeclarationList} \rangle ::= \langle \text{DeclarationList} \rangle$
 $\langle \text{DeclarationList} \rangle ::= \langle \text{Declaration} \rangle \langle \text{DeclarationListPrime} \rangle$
 $\langle \text{Declaration} \rangle ::= \langle \text{Qualifier} \rangle \langle \text{IDs} \rangle$
 Token: Keyword Lexeme: "int"
 $\langle \text{Qualifier} \rangle ::= \text{int}$
 Token: Identifier Lexeme: "x"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 Token: Comma Lexeme: ","
 $\langle \text{IDsPrime} \rangle ::= , \langle \text{IDs} \rangle$
 Token: Identifier Lexeme: "y"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 Token: Comma Lexeme: ","
 $\langle \text{IDsPrime} \rangle ::= , \langle \text{IDs} \rangle$
 Token: Identifier Lexeme: "z"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 Token: Comma Lexeme: ","
 $\langle \text{IDsPrime} \rangle ::= , \langle \text{IDs} \rangle$
 Token: Identifier Lexeme: "x2\$"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 Token: Comma Lexeme: ","
 $\langle \text{IDsPrime} \rangle ::= , \langle \text{IDs} \rangle$
 Token: Identifier Lexeme: "y2\$"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 $\langle \text{IDsPrime} \rangle ::= \langle \text{Empty} \rangle$
 Token: Semicolon Lexeme: ";"

<DeclarationListPrime> ::= <Empty>
 <Statement> ::= <Scan>
 Token: Keyword Lexeme: "get"
 <Scan> ::= get (<IDs>) ;
 Token: LParen Lexeme: "("
 Token: Identifier Lexeme: "x"
 <IDs> ::= <Identifier> <IDsPrime>
 Token: Comma Lexeme: ","
 <IDsPrime> ::= , <IDs>
 Token: Identifier Lexeme: "y"
 <IDs> ::= <Identifier> <IDsPrime>
 Token: Comma Lexeme: ","
 <IDsPrime> ::= , <IDs>
 Token: Identifier Lexeme: "z"
 <IDs> ::= <Identifier> <IDsPrime>
 <IDsPrime> ::= <Empty>
 Token: RParen Lexeme: ")"
 Token: Semicolon Lexeme: ";"
 <StatementListPrime> ::= <StatementList>
 <Statement> ::= <Print>
 Token: Keyword Lexeme: "put"
 <Print> ::= put (<Expression>) ;
 Token: LParen Lexeme: "("
 <Primary> ::= <Identifier> (<IDs>)
 Token: Identifier Lexeme: "timesFiveMinusTwo\$"

Token: LParen Lexeme: "("
 Token: Identifier Lexeme: "x"
 <IDs> ::= <Identifier> <IDsPrime>
 <IDsPrime> ::= <Empty>
 Token: RParen Lexeme: ")"
 <TermPrime> ::= <Empty>
 <ExpressionPrime> ::= <Empty>
 Token: RParen Lexeme: ")"
 Token: Semicolon Lexeme: ";"
 <StatementListPrime> ::= <StatementList>
 <Statement> ::= <While>
 Token: Keyword Lexeme: "while"
 <While> ::= while (<Condition>) <Statement>
 Token: LParen Lexeme: "("
 <Condition> ::= <Expression> <Relop> <Expression>
 Token: Identifier Lexeme: "y"
 <Primary> ::= <Identifier>
 <TermPrime> ::= <Empty>
 <ExpressionPrime> ::= <Empty>
 Token: Less Lexeme: "<"
 <Relop> ::= <
 Token: Identifier Lexeme: "z"
 <Primary> ::= <Identifier>

$\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: RParen Lexeme: ")"
 $\langle \text{Statement} \rangle ::= \langle \text{Compound} \rangle$
Token: LBrace Lexeme: "{"
 $\langle \text{Compound} \rangle ::= \{ \langle \text{StatementList} \rangle \}$
 $\langle \text{Statement} \rangle ::= \langle \text{Print} \rangle$
Token: Keyword Lexeme: "put"
Token: LParen Lexeme: "("
Token: Identifier Lexeme: "y"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: RParen Lexeme: ")"
Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{StatementList} \rangle$
 $\langle \text{Statement} \rangle ::= \langle \text{Print} \rangle$
Token: Keyword Lexeme: "put"
Token: LParen Lexeme: "("
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle (\langle \text{IDs} \rangle)$
Token: Identifier Lexeme: "square"
Token: LParen Lexeme: "("
Token: Identifier Lexeme: "y"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 $\langle \text{IDsPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: RParen Lexeme: ")"
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: RParen Lexeme: ")"
Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{StatementList} \rangle$
 $\langle \text{Statement} \rangle ::= \langle \text{Assign} \rangle$
Token: Identifier Lexeme: "y"
 $\langle \text{Assign} \rangle ::= \langle \text{Identifier} \rangle = \langle \text{Expression} \rangle ;$
Token: Assign Lexeme: "="
Token: Identifier Lexeme: "y"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: Plus Lexeme: "+"
 $\langle \text{ExpressionPrime} \rangle ::= + \langle \text{Term} \rangle \langle \text{ExpressionPrime} \rangle$
Token: Int Lexeme: "1"
 $\langle \text{Primary} \rangle ::= \langle \text{Integer} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: RBrace Lexeme: "}"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{StatementList} \rangle$

$\langle \text{Statement} \rangle ::= \langle \text{Assign} \rangle$
Token: Identifier Lexeme: "x2\$"
 $\langle \text{Assign} \rangle ::= \langle \text{Identifier} \rangle = \langle \text{Expression} \rangle ;$
Token: Assign Lexeme: "="
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle (\langle \text{IDs} \rangle)$
Token: Identifier Lexeme: "square"
Token: LParen Lexeme: "("
Token: Identifier Lexeme: "x"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 $\langle \text{IDsPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: RParen Lexeme: ")"
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{StatementList} \rangle$
 $\langle \text{Statement} \rangle ::= \langle \text{Assign} \rangle$
Token: Identifier Lexeme: "y2\$"
 $\langle \text{Assign} \rangle ::= \langle \text{Identifier} \rangle = \langle \text{Expression} \rangle ;$
Token: Assign Lexeme: "="
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle (\langle \text{IDs} \rangle)$
Token: Identifier Lexeme: "square"
Token: LParen Lexeme: "("
Token: Identifier Lexeme: "y"
 $\langle \text{IDs} \rangle ::= \langle \text{Identifier} \rangle \langle \text{IDsPrime} \rangle$
 $\langle \text{IDsPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: RParen Lexeme: ")"
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: Semicolon Lexeme: ";"
 $\langle \text{StatementListPrime} \rangle ::= \langle \text{StatementList} \rangle$
 $\langle \text{Statement} \rangle ::= \langle \text{If} \rangle$
Token: Keyword Lexeme: "if"
 $\langle \text{If} \rangle ::= \text{if} (\langle \text{Condition} \rangle) \langle \text{Statement} \rangle \text{endif} \mid \text{if} (\langle \text{Condition} \rangle) \text{else} \langle \text{Statement} \rangle \text{endif}$
Token: LParen Lexeme: "("
Token: Identifier Lexeme: "x2\$"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: Greater Lexeme: ">"
 $\langle \text{Relop} \rangle ::= >$
Token: Identifier Lexeme: "y2\$"
 $\langle \text{Primary} \rangle ::= \langle \text{Identifier} \rangle$
 $\langle \text{TermPrime} \rangle ::= \langle \text{Empty} \rangle$
 $\langle \text{ExpressionPrime} \rangle ::= \langle \text{Empty} \rangle$
Token: RParen Lexeme: ")"
 $\langle \text{Statement} \rangle ::= \langle \text{Return} \rangle$
Token: Keyword Lexeme: "return"
Token: Keyword Lexeme: "true"

```

<Primary> ::= true
<TermPrime> ::= <Empty>
<ExpressionPrime> ::= <Empty>
<ReturnPrime> ::= <Expression> ;
Token: Semicolon Lexeme: ";"
Token: Keyword Lexeme: "endif"
<StatementListPrime> ::= <Empty>

```

Test3

```

function convert$ [fahr:int]

{

    return 5*(fahr -32)/9;

}

int low, high step$ !declarations!

put (9.0);
get (low, high, step$);
while (low < high )
{ put (low);
  put (convert$ (low));
  low = low + step$;
}

```

Results

```

<Rat18S> ::= <OptFunctionDefinitions> %% <OptDeclarationlist> <StatementList>
<OptFunctionDefinitions> ::= <FunctionDefinitions> | <Empty>
<FunctionDefinitions> ::= <Function> <FDPrime>
Token: Keyword Lexeme: "function"
<Function> ::= function <Identifier> [ <OptParameterList> ] <OptDeclarationList> <Body>
Token: Identifier Lexeme: "convert$"
Token: LBracket Lexeme: "["
<OptParameterList> ::= <ParamaterList> | <Empty>
<ParamaterList> ::= <Parameter> <ParameterListPrime>
<Parameter> ::= <IDs> : <Qualifier>
Token: Identifier Lexeme: "fahr"
<IDs> ::= <Identifier> <IDsPrime>
<IDsPrime> ::= <Empty>
Token: Colon Lexeme: ":"
Token: Keyword Lexeme: "int"
<Qualifier> ::= int

```

```

<ParameterListPrime> ::= <Empty>
Token: RBracket Lexeme: "]"
<OptDeclarationList> ::= <Empty>
Token: LBrace Lexeme: "{"
<Body> ::= { <StatementList> }
<StatementList> ::= <Statement> <StatementListPrime>
<Statement> ::= <Return>
Token: Keyword Lexeme: "return"
<Return> ::= return <ReturnPrime>
<Expression> ::= <Term> <ExpressionPrime>
<Term> ::= <Factor> <TermPrime>
<Factor> ::= - <Primary> | <Primary>
Token: Int Lexeme: "5"
<Primary> ::= <Integer>
Token: Times Lexeme: "*"
<TermPrime> ::= * <Factor> <TermPrime>
<Primary> ::= ( <Expression> )
Token: LParen Lexeme: "("
Token: Identifier Lexeme: "fahr"
<Primary> ::= <Identifier>
<TermPrime> ::= <Empty>
Token: Minus Lexeme: "-"
<ExpressionPrime> ::= - <Term> <ExpressionPrime>
Token: Int Lexeme: "32"
<Primary> ::= <Integer>
<TermPrime> ::= <Empty>
<ExpressionPrime> ::= <Empty>
Token: RParen Lexeme: ")"
Token: Div Lexeme: "/"
<TermPrime> ::= / <Factor> <TermPrime>
Token: Int Lexeme: "9"
<Primary> ::= <Integer>
<TermPrime> ::= <Empty>
<ExpressionPrime> ::= <Empty>
<ReturnPrime> ::= <Expression> ;
Token: Semicolon Lexeme: ";"
<StatementListPrime> ::= <Empty>
Token: RBrace Lexeme: "}"
<FDPrime> ::= <Empty>
line 9
Error at int: Expecting '%%' after function definitions.
main: Maybe.fromJust: Nothing

```