Recursive Descent Parser for a PL/0 like programming language in pseudo code:

As follows you will find the pseudo code for a PL/0 like parser. This pseudo code will help you out to develop your parser and intermediate code generator for tiny PL/0. You must first double check the grammar on HW3 to identify the parts of this parser that do not apply to your parser:

```
procedure PROGRAM;
  begin
    GET(TOKEN);
    BLOCK;
    if TOKEN != "periodsym" then ERROR
  end;

procedure BLOCK;
  begin
    if TOKEN = "constsym" then begin
      repeat
        GET(TOKEN);
        if TOKEN != "identsym" then ERROR;
        GET(TOKEN);
        if TOKEN != "eqsym" then ERROR;
        GET(TOKEN);
        if TOKEN != NUMBER then ERROR;
        GET(TOKEN)
      until TOKEN != "commasym";
      if TOKEN != "semicolomsym" then ERROR;
      GET(TOKEN)
    end;
    if TOKEN = "varsym" then begin
      repeat
        GET(TOKEN);
        if TOKEN != "identsym" then ERROR;
        GET(TOKEN)
      until TOKEN != "commasym";
      if TOKEN != "semicolomsym" then ERROR;
      GET(TOKEN)
    end;
    while TOKEN = "procsym" do begin
      GET(TOKEN);
      if TOKEN != “identsym” then ERROR;
      GET(TOKEN);
      if TOKEN != "semicolomsym" then ERROR;
      GET(TOKEN);
```

```
        BLOCK;
        if TOKEN != "semicolomsym" then ERROR;
        GET(TOKEN)
      end;
      STATEMENT
    end;

procedure STATEMENT;
  begin
    if TOKEN = "identsym" then begin
      GET(TOKEN);
      if TOKEN != "becomessym" then ERROR;
      GET(TOKEN);
      EXPRESSION
    end
    else if TOKEN = "callsym" then begin
      GET(TOKEN);
      if TOKEN != "identsym" then ERROR;
      GET(TOKEN)
    end
    else if TOKEN = "beginsym" then begin
      GET TOKEN;
      STATEMENT;
      while TOKEN = "semicolomsym" do begin
        GET(TOKEN);
        STATEMENT
      end;
      if TOKEN != "endsym" then ERROR;

      GET(TOKEN)
    end
    else if TOKEN = "ifsym" then begin
      GET(TOKEN);
      CONDITION;
      if TOKEN != "thensym" then ERROR;
      GET(TOKEN);
      STATEMENT
    end
    else if TOKEN = "whilesym" then begin
      GET(TOKEN);
      CONDITION;
      if TOKEN != "dosym" then ERROR;
      GET(TOKEN);
      STATEMENT
    end
  end;

procedure CONDITION;
  begin
    if TOKEN = "oddsym" then begin
```

```
        GET(TOKEN);
        EXPRESSION
      else begin
        EXPRESSION;
        if TOKEN != RELATION then ERROR;
        GET(TOKEN);
        EXPRESSION
      end
    end;

procedure EXPRESSION;
    begin
      if TOKEN = "plussym"or "minussym" then GET(TOKEN);
      TERM;
      while TOKEN = "plussym" or "minussym" do begin
        GET(TOKEN);
        TERM
      end
    end;

procedure TERM;
    begin
      FACTOR;
      while TOKEN = "multsym" or "slashsym" do begin
        GET(TOKEN);
        FACTOR
      end
    end;

procedure FACTOR;
    begin
      if TOKEN = "identsym then
        GET(TOKEN)
      else if TOKEN = NUMBER then
        GET(TOKEN)
      else if TOKEN = "(" then begin
        GET(TOKEN);
        EXPRESSION;
        if TOKEN != ")" then ERROR;
        GET(TOKEN)
      end
      else ERROR
    end;
```