level specifications. The main reasons for insisting that indenting programs also accept 'incorrect' text are:

(i) Syntactic checking unnecessarily overburdens indenting programs.
(ii) Properly indented text helps us quickly identify syntax errors.
(iii) There are many variations of Pascal language in existence.

In fact, only minor modifications should be sufficient to produce an indenting program for other Pascal-like languages.

2. The output from an indenting program should appear 'properly indented'. This notion is made precise in later sections. (The particular indentation scheme that we suggest may not appear 'pretty' to some, but we remind that our interest in the scheme here is only as a concrete running example.) Proper indentation involves essentially three independent activities:
   (i) Each line should be started at the appropriate left margin.
   (ii) Certain constructs of the language should not be hidden in the middle of a line. For instance, it appears important that reserved words such as **while**, **repeat**, **procedure** always appear at the beginning of a line and are never embedded in a line. Reading many Pascal programs convinced us that no line of a listing should contain more than one statement. Multiple assignments on one input line should be split up.
   (iii) Adjusting the inter-word blank spacing so that it is visually appealing. Subjective preferences and special circumstances abound in this matter, and we specify here that this spacing be left unaltered except for splitting.

   We believe that each input line should generate an integral number of output lines. Combining two or more input lines and then splitting them up is often unsatisfactory and leads to complicated 'control language' to specify (in the program being indented) how the lines are to be split.
3. We also believe that indenting programs which produce output always different from their inputs are undesirable. More specifically, if we feed the indented text back to the indenting program as input, the output must be identical to the input. This characteristic of indenting programs is important from a psychological point of view.

It should be borne in mind that no matter how well we specify the input to output transformation whether an alleged indenting program should indeed be called an indenting program, and for what language, has to be judged by subjective considerations. For instance, our definition of lexical analysis is sure to startle some.

Figure 1 gives an example function indented according to our specifications. There are many inputs which produce that indented output. For the sake of concreteness, assume that the input was the text of the Figure shown, but with all leading white space in each line deleted. The reader is encouraged to compute the various functions and predicates that we define below on this input.

3. NOTATION

We denote by $\backslash b$, $\backslash t$, $\backslash n$, $\backslash e$ the characters blank, tab, end-of-line and end-of-file marker, respectively. The first three of these characters are referred to as white characters; we denote by % any one of these. For simplicity in this paper, we replace each tab by a fixed number, say 8, of blanks and assume from now on that tabs do not occur. By *white space* we mean any (possibly mixed) string of white characters. A *line* is