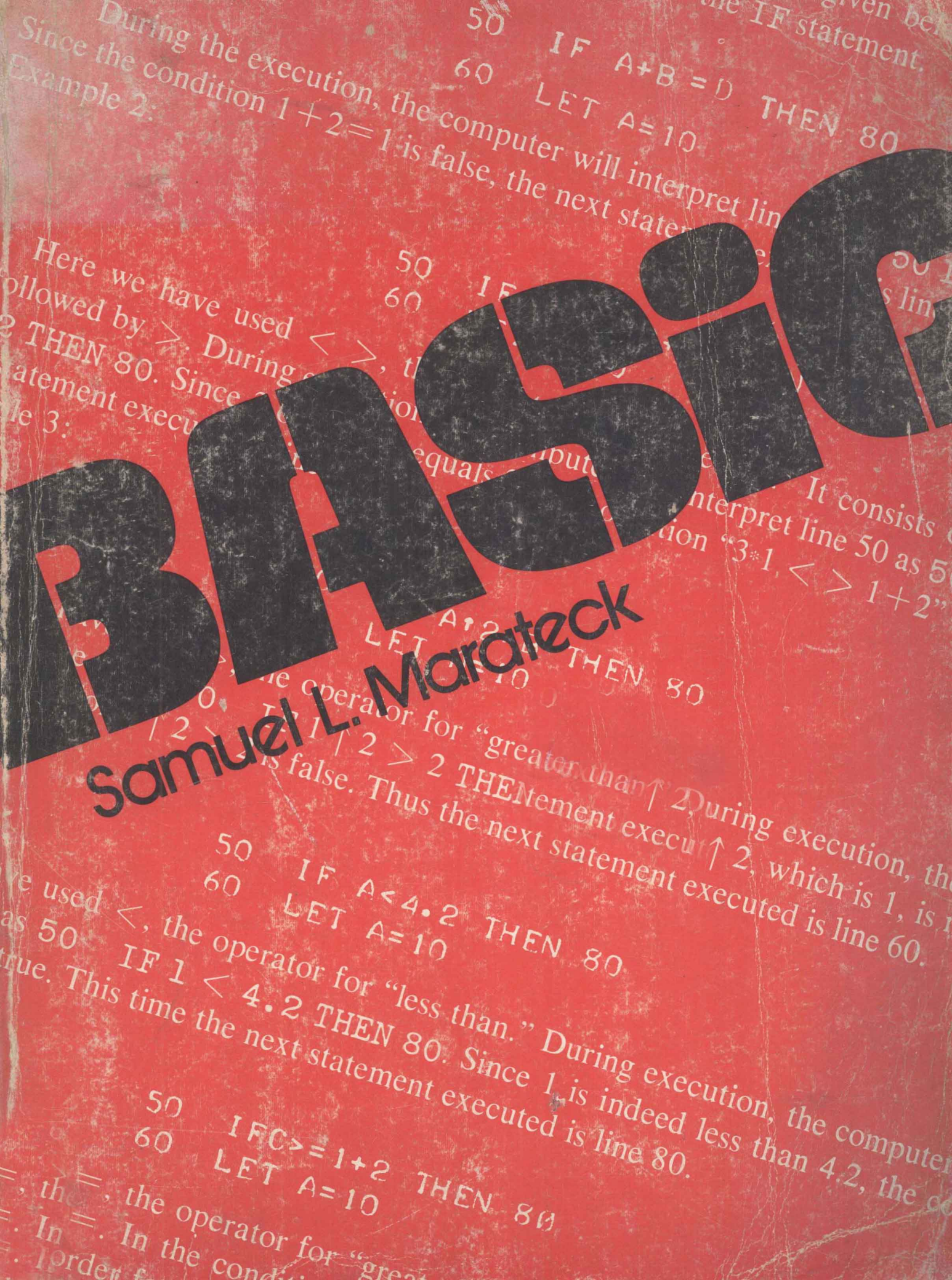

BASIC
Samuel L. Marateck

BASIC

Samuel L. Marateck

The Courant Institute of Mathematical Sciences
New York University

ACADEMIC PRESS *NEW YORK SAN FRANCISCO LONDON*

A Subsidiary of Harcourt Brace Jovanovich, Publishers

ACADEMIC PRESS, INC.
111 Fifth Avenue, New York, New York 10003

United Kingdom Edition published by
ACADEMIC PRESS, INC. (LONDON) LTD.
24/28 Oval Road, London NW1

Library of Congress Cataloging in Publication Data

Marateck, Samuel.
BASIC.

Bibliography: p.
Includes index.
1. Basic (Computer program language) I. Title.
QA76.73.B3M37 001.6'424 74-27787
ISBN 0-12-470450-6

PRINTED IN THE UNITED STATES OF AMERICA

BASIC

To my parents Harold and Rita

Preface

This book is an outgrowth of notes the author uses in a course in BASIC that he teaches to undergraduates at New York University. Its purpose is to teach the student how to program using the BASIC language, and it has been written for students who have no prior knowledge of computers or programming. In order to make the book easier for such students to understand, the author in most cases has introduced only one new programming concept per program. Thus, many of the programs in the book are first written as a series of smaller programs, each of which serves as a step in understanding the entire larger program. The author has found this to be an effective teaching technique.

Simple programs have been used to illustrate the various programming techniques discussed in this book. These programs are the solutions to problems drawn from various disciplines, and they are of a nature such that students, whatever their major field, should understand them without difficulty.

The author has also included in the book examples of common programming mistakes made by beginning students when they are not explicit enough in translating their thoughts into programming instructions. The author has found this type of example to be another effective teaching technique. Now a word on why BASIC is the ideal language to use in an introductory course.

The vast majority of BASIC programs are typed and run in an interactive environment; that is, the student and computer can interact with each other while the student is typing and running his program. For instance, the computer informs the student that he has made an error almost as soon as he has made it. The student can then correct his mistakes and rerun his program without delay. This immediate feedback enables students to learn the elements of programming quite easily. In order to make this task even easier, the author has designed the book in the following way: the right-hand pages contain pictorial material on programming, which should be easily digestible; this is described in detail in "To the Reader" (page xi). The left-hand pages contain the text. You will see that strict adherence to this design has resulted in a number of partially filled pages. It has been the author's experience that, in many cases, students who have sat down at the teletypewriter without having previously gone to class, and have used only the right-hand pages of preliminary versions of this book, have been able to write programs in their first session at the teletypewriter. In order for the student to understand all the ramifications of the programming techniques described he should also read the text on the left-hand facing page. We now make some remarks about BASIC itself.

The BASIC language was developed under the direction of J. Kemeny and T. Kurtz at Dartmouth College in the 1960s. Since then, the various manufacturers of computers in developing their own versions of BASIC have added programming instructions to the original Dartmouth version. Fortunately, the significant additions incorporated in these various versions

of BASIC for the most part have been quite similar. Therefore, the reader should encounter no difficulty in running on any computer that supports BASIC the overwhelming majority of the programs presented in this book. This includes the programs involving matrix operations.

The format of the matrix statements is such that one general form can be used on all computers. However, depending on the version of BASIC you are using, you may have options on how and where in the program you can dimension matrices other than in a `DIM` statement. Therefore, in order to enable you to run the programs that involve matrices on any computer, we have written these programs without using any of the available dimension options. However, we have described all the possible dimension options in a table that appears at the end of the chapter on matrices.

Perhaps the only way that your interaction with the computer may differ somewhat from what we describe here is how you instruct the computer to process your program. We have described the processing instructions—called system commands—that are in most general use, i.e., those used in Dartmouth BASIC [GE, DEC(PDP), CDC, UNIVAC, and RTB]; however, we have detailed in the footnotes how the system on another popular system, the one used on the Hewlett-Packard 2000 series, differs from what we describe in the text. In the other chapters we have described in the footnotes how some aspects of the instructions used on other systems—especially the Hewlett-Packard but also the XDS Sigma series, IBM 360, IBM 370, and CDC 6600—differ from what we describe in the text.

The chapters in this book, with the exception of the one on system commands (Chapter 3), should be read in sequential order. The reader may want to defer reading Chapter 3 until some later time. Chapter 3 was placed toward the beginning of the book because it was felt that the knowledge of system commands would enable the reader to write and correct programs more easily and effectively.

The programs shown in this book were checked out and run on one or more of the following computers: Hewlett-Packard 2000C, CDC 6600, XDS Sigma 7, IBM 370, and UNIVAC 1108.

It is a pleasure to thank Professor J. T. Schwartz and Professor Max Goldstein for their kindness to me while I was writing this book, and Jeffrey Akner and Martin Mathiot for granting me free time on their machines.

To the Reader

This book has been written with the premise that it is at times easier to learn a subject from pictorial representations supported by text than from text supported by pictorial representations. With this in mind, beginning with Chapter 2 we have used a double-page format for our presentation. On the left-hand page (we call it the text page) appears the text, and on the right-hand page (we call it the picture page) appears the pictorial representation, consisting mostly of programs and tables.

Each picture page was written to be as self-contained as possible, so that the reader, if he so desires, may read that page first and absorb the essence of the contents of the entire double page before going on to read the text. The text page consists of a very thorough discussion of the programming techniques presented on the picture page. It refers to parts of the programs and tables on the picture page; when reference is made on the text page to a given line of print on the picture page, that line—whenever it is feasible to do so—is reproduced in the text to promote readability. Students who have a previous background in programming languages and others who understand the picture page completely may find that in some chapters they can skip the text (left-hand) pages and concentrate on the picture pages.

The following techniques are used as aids in making the picture page self-contained.

1. As many as possible of the ideas discussed in the text are illustrated in the programs and tables. The captions beneath these capsulate much of what is said in the text.

2. Words underlined in the captions describe lines underlined in the figures. To illustrate this, Fig. 2.1a is reproduced below.

```
10      LET A=21.2
20      LET B=30.1
30      END
```

Figure 2.1a. A simple BASIC program illustrating the use of the assignment statement.

The statements `10 LET A = 21.2` and `20 LET B = 30.1` are underlined to show that they are described by the words underlined in the caption. Thus they are both assignment statements.

3. To the right of most programs appears a table that describes what effect certain lines in the program have on the computer's memory. For instance, the following table describes the effect that the line of programming to its left has on the memory:

```
10        LET A=21.2
```

Line no.	A
10	21.2

We see from the table that line number 10 of the program causes the number 21.2 to be associated with A in the computer's memory. The line-by-line analysis afforded by these tables should aid the reader in understanding the program.

Contents

4

READ, DATA, GO TO, IF, and INPUT Statements

5

The FOR-NEXT Loop

6

Strings and Library Functions

7

The STOP, Multiple Assignment, and ON-GO TO Statements and Subscripted Variables

8

Subscripted String Variables, Subroutines and User-Defined Functions

9

Doubly Subscripted Variables and Matrices

10

`PRINT USING` **and Files**

1

Introduction to Computers and Programming

1.1. General Remarks

Most of you reading this book have had very little previous experience with computers. You therefore may think of a computer as an electronic brain that makes important decisions for astronauts or predicts the outcome of political elections. A computer is certainly this, and more. However, it is not a machine that does things of its own volition, independent of what man orders it to do. At the present stage of their development, computers only follow the instructions that people give them. These instructions are called programs, and the people who write programs are called programmers.

The form that these instructions take depends on the programming language the programmer uses. Some of these languages are hard to learn whereas others are relatively easy. It is the purpose of this book to teach you how to write programs in the language that is the easiest to learn. It is called BASIC, an acronym for Beginner's All-purpose Symbolic Instruction Code. First, a few comments on computers and programming languages in general.

One way of picturing a computer is as a maze of on–off electrical switches connected by wires. Thus, you might imagine that, if a programmer wished to instruct a computer to do something, he would have to feed it a program composed of a series of on–off types of instructions. As a matter of fact, the first programs written were like this. The type of language that uses this form of instruction is called "machine language," and it is, to a limited extent, still used today by certain categories of expert programmers. Writing programs in machine language is very tedious. For this reason, computer languages closer in form to English and algebra have been devised. The easiest of these to learn is BASIC.

A program written in BASIC cannot be directly understood by the computer; it must first be translated into machine language. A special program, called a compiler, which is already present in the computer, does this. BASIC has grammatical rules that must be followed by the programmer. These rules are similar to those in English which govern the sequence of words in a sentence, punctuation, and spelling—we shall learn these rules in

later chapters. Before the compiler translates your program, it checks whether you have written your program instructions according to the grammatical rules. If you make grammatical errors in writing an instruction, don't worry; the compiler has been written so that it will inform you of these. You must correct the errors before the compiler will translate your program. Do not be surprised if your programs are full of errors—it is a rare person who can write a program without making them.

Now, we give a more detailed view of the computer. Essentially, the computer consists of an *input unit,* a *memory unit,* an *arithmetic unit,* an *output* unit, and a *control unit.* We communicate our program to the computer through the *input unit.* All the mathematics and decisions in the program are done in the *arithmetic unit.* Numbers are stored in the *memory,* which consists of thousands of memory locations. We shall speak about these memory locations in more detail later in this chapter. The computer communicates the results of our program to us through the *output unit.* Finally the *control unit* directs the activities of the other four units.

The word hardware is used to describe the physical components of the computer, such as these units, whereas the word software is used to describe the programs. We shall use the word system to describe the programs, such as the compiler, that process the programs you write.

In general, the system used on one make of computer will differ somewhat from the system used on another make. In fact, it is possible to implement more than one system on a given computer. Therefore, when we describe the differences—most of them minor—that one may encounter when running the same BASIC program on different computers, we shall, for the most part, attribute these differences to the system employed.

We now describe that piece of hardware which, for most of you, will be the only part of the computer you will ever encounter: the teletypewriter.

1.2. The Teletypewriter

The teletypewriter is both an input and an output device. Basically, it is a typewriter that is electrically connected to the rest of the computer. From now on we will follow the common practice of using the word *computer* to refer to the rest of the computer as opposed to the teletypewriter. The programmer types his program on the teletypewriter, and the computer types the results on it. The part of the teletypewriter that comes into actual contact with the teletypewriter paper while doing the printing is called the printing head. In Fig. 1.1 we show a typical teletypewriter.

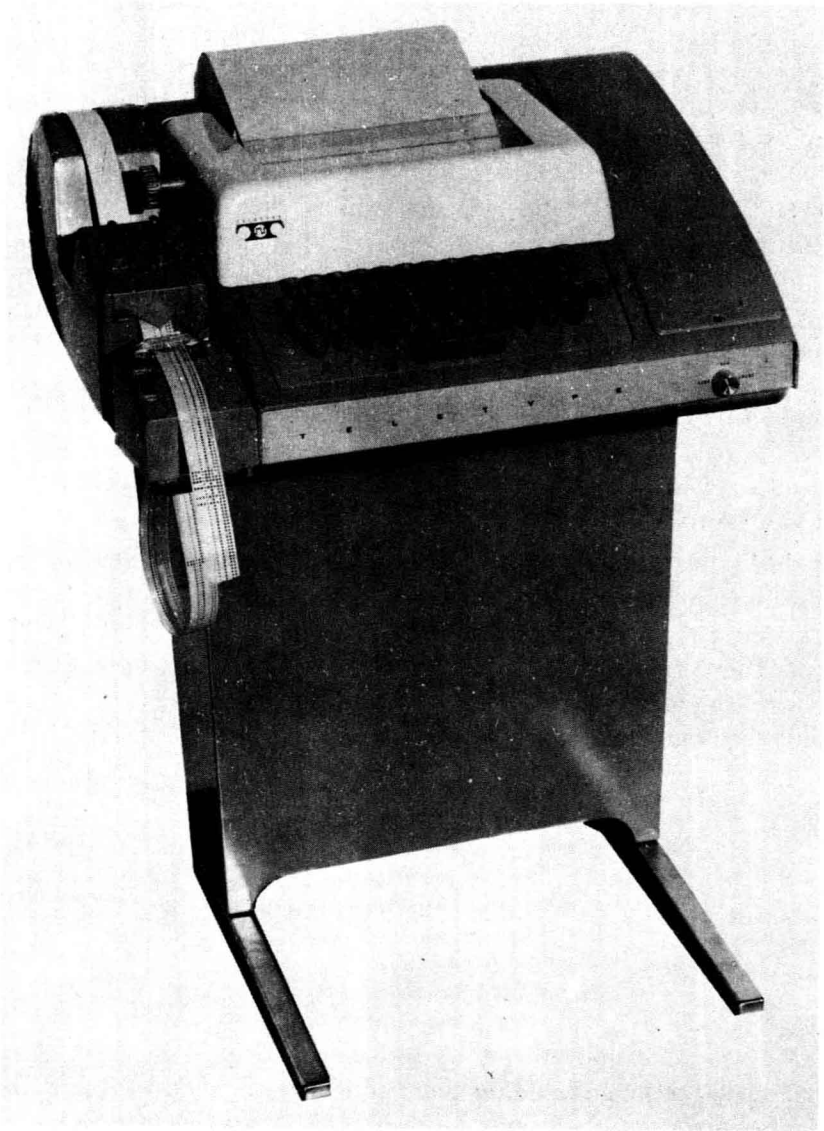
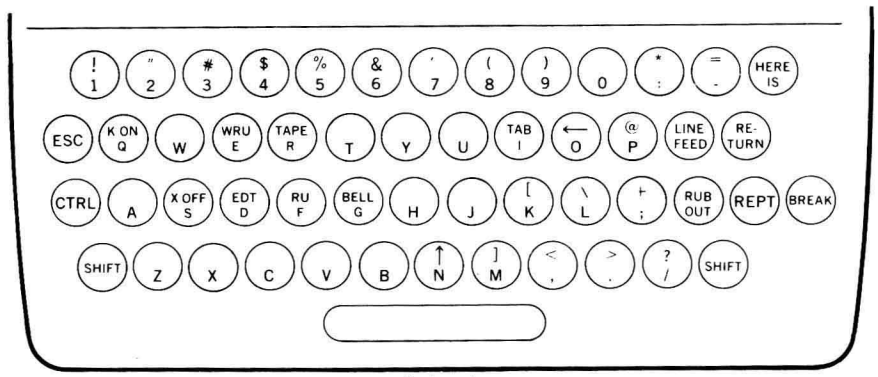


Figure 1.1. Teletype Model 33 unit. (Courtesy of Teletype Corporation.)



Keyboard

Figure 1.2. Teletypewriter keyboard. (Courtesy of Teletype Corporation.)

The teletypewriter keyboard—the one shown in Fig. 1.2 is for the Teletype Corporation teletypewriter—differs somewhat from the typewriter keyboard. Although you can type lowercase letters—for example, a—on a typewriter, you cannot type them on the teletypewriter; when you depress the teletypewriter key marked A, a capital letter A appears on the teletypewriter paper. If you simultaneously depress the SHIFT key and one of the keys on which there is both a letter of the alphabet and above it a symbol—for example,

the ↑, but not a capital N, will be printed. On the other hand, if you try to depress simultaneously the SHIFT key and one of the keys on which there is both a letter and above it a group of letters—for example,

nothing will be typed. These groups of letters, in almost all cases, are not used to communicate with the computer. If they are used, they must be depressed simultaneously with the CTRL key.

We now briefly describe the function of the other keys. Some of these functions will be described in greater detail in the appropriate places later in the book.

LINE FEED: This key advances the paper in the teletypewriter one line.

RETURN: This key returns the typing head to the beginning of the next line, signaling the computer that you have finished typing a line. The information on that line is then recorded by the computer.

RUB OUT: In some systems, if this key is depressed before RETURN, it deletes the line just typed so that the line is not recorded by the computer. It is also used in punching a paper tape.

BREAK: This key is used on some systems to stop your program or the automatic retyping of your program.

ALT MODE or, on some teletypewriters, ESC: If this key is depressed before RETURN, it deletes the line just typed so that the line is not recorded by the computer.

CTRL: In some systems, when this key is depressed simultaneously with some other key, it has the same effect as BREAK.

REPT: This key, when depressed simultaneously with another key, causes the action of that key to be repeated.

HERE IS: This key is used in the process of punching a paper tape to produce holes in the tape for a leader or trailer.

The operation of the remaining keys is the same as on a typewriter. If the key is depressed, the lower symbol appearing on the key is typed. If the key is depressed together with the SHIFT key, the upper symbol is typed. We shall explain the function of the symbols that are neither numeric nor alphabetic as we encounter them in the book. For the remainder of the book we shall refer to the symbols on the keyboard—except for the groups of letters—as characters.