This report discusses the general philosophy of coding problems for a large scale digital computing machine, and applies it to a number of relatively simple examples, using the orders introduced in the report reviewed above. The authors first observe that the coding of almost all interesting mathematical problems is not static in the sense that the orders are given in a fixed sequence which is to be traversed once by the control of the machine, but rather it is dynamic in the sense that the control of the machine goes repeatedly through some sets of storage registers and that each time it finds that some of the orders have been changed. These changes may be of a predictable type as one would find in an induction process, or they may depend on the numbers to be computed as in a method of successive approximation where the number of iterations is not known in advance. In order to grasp this dynamic situation in its entirety, the authors propose the use of a “flow diagram” which symbolically represents the path the control of the machine is to follow as it goes about carrying out the instructions it finds. Simple and multiple inductions are indicated by one or more closed oriented loops. Each such loop has an alternative box which decides when to go around again and when to go elsewhere. Further boxes are added to the flow diagram to indicate the logical situation at selected points. These flow diagrams emphasize the logical aspects of the problem and subordinate the purely arithmetical aspects. After completion of the flow diagram, the actual coding is fairly easy, though tedious.

The report gives a number of examples, showing details, of the coding of simple problems. These problems include coding a floating binary point for division, square rooting using an iterative system, conversion to and from the binary system and “double accuracy” arithmetical operations. It also includes a rule of thumb for estimating times of operations in microseconds.