FOREWARD

This Tutorial Text on Decision Tables using DETAB-X was developed for the Decision Table Symposium held in New York City on September 20 and 21, 1962 under joint sponsorship of the JUG-CODASYL Systems Development Group and the ACM Joint Users Group. This self-instruction type text was used on the second day of the Symposium with instructors present to clarify questions which might arise. An average of approximately one (1) instructor to 25 students was found to be satisfactory. If the attached corrections are made, it is thought that one (1) instructor should be able to handle at least 50 students and that much could be gained without any instructors. It is assumed that the student will start at the front of the text and go straight through to the end.

The text was developed:

1. To give an introductory learning experience in the use of Decision Tables themselves.

2. To give specific instructions in the use of DETAB-X, an experimental language combining Decision Table structure and COBOL 61 language with minor modifications. In the design of DETAB-X, a number of somewhat arbitrary decisions were made, such as: the use of vertical rules as opposed to horizontal rules, and the sequence in which rules are given having no influence on how subsequent rules using the same tables are formed.

Problems contained in Sections I, II and III were designed to demonstrate certain concepts which it was desired for the student to understand. In some of the problems, it will be noted that the condition statements for a rule are not necessarily independent of those in another rule, and that the stated sequence is implied. The problems in Section IV adhere to the rules spelled out in DETAB-X with minor modifications, and demonstrate how it would be possible to make use of existing programming languages in conjunction with Decision Table structure. Throughout the text some problems have been over-simplified and others are quite trivial. However, it was thought that the main objective might best be met using many problems rather than one. In answer to numerous requests, it is hoped an article presenting a complete problem will be forthcoming before long.

It is highly desirable that the student realize that the reason behind

FOREWARD

Feedback should be directed to:

Mr. Solomon L. Pollack
The Rand Corporation
1700 Main Street
Santa Monica, California

Questions may be addressed to:

Mrs. Mary K. Hawes
Radio Corporation of America
Bldg. 204-2, Cherry Hill
Camden 8, New Jersey

Mary K. Hawes

Subject: ERRATA - DECISION TABLE TUTORIAL USING DETAB-X

October 1, 1962

The following changes should be made to the Decision Table Tutorial Manual, and the comments noted.

I-4 Paragraph 2 line 1 should read:

"If the specimen is an animal, and it has 4 legs, and it has a long nose..."

I-5 Comment: Since the meaning was not altered in any way, these rules were stated by a shortened version.

Rule 2 may also be stated: "If the specimen is an animal, and it has 4 legs, and a short nose, and a long neck, then it is a giraffe and go to Table G."

Rule 3 may also be stated:

"If the specimen is an animal, and it has 4 legs, and a long nose, and a long neck, then it is an hallucination and go to a psychiatrist."

I-7 Line 1 should read: "...in Rule 1 are not met, then try Rule 2."

I-8 Rule number row in table header:

Change 5 to ELSE.

I-9 Rule number row in table header:

Change 9 to ELSE.

I-10 Line 1: change page 7 to page 8.

I-10 Line 2 should read:

"In TABLE CHANGE we would have searched until we reached the EISE Rule."

II-4 The Stub of Condition Row 5 in the table should read: "TAPE FAULTY"

II-5 Comment: The actions noted in this over-simplified table are the ones normally required of the tape librarian. An ELSE Rule is not required in this table because it is understood that the librarian will investigate all situations out of the ordinary that occur.

II-6 The Stub of Condition Row 5 in the table should read: "TAPE FAULTY"

II-8 Condition Row 3 Rule 2 should read: "LR CURR YEAR".

LR = LesseR than

II-8 and II-10 The Stub of Condition Row 5 in the tables for added clarity may be read: "VACATION WEEKS EARNED UNEQUAL TO".

III-4 Double vertical line should separate stub from entries in table.

III-6 Double vertical line should separate stub from entries in table.

III-6 Comment: Note that it is permissible to have either an additional Rule number (such as Rule 3 in the table) or an ELSE Rule to insure that all conditions have been tested.

III-7 Line 13:

(PCGRS) should be changed to (RDGRS).

III-8 Double vertical line should separate stub from entries in table.

III-9 Line 2 paragraph 2 should read: "When the read operation is executed, a record..."

III-10 Comment: Regarding the Stub of Action Row 1 in table, see IV-31 last paragraph.

III-11 Line 2 should read:

III-12 Comment: Answer 3 may include any 2 of the 4 data names in the action area:

RATE
PURATE
RSKFAC
SPRFAC

III-13 Line 1 word 2 should be row instead of rule.

III-13 Line 3: Spaces should be allowed in formula (RATE - SPRFAC). When spaces are omitted, formula becomes a hyphenated data name.

III-13 Line 7 last word should be row instead of rule.

III-13 Question 3 should read: "Which condition row is in extended entry form."

III-16 Answer 4. "Blank" should be changed to "BANK".

III-16 The Stub of Action Row 3 in table should read:

"SET BNDAMT EQ BNDAMT"

III-16 The Stub of Action Row 5 in table should read:

"SET STKAMT EQ STKAMT"

III-19 The Stub of Condition Row 8 should read:

"QUANTITY GR ON-HAND-A"

III-21 Last entry heading should read: ELSE

III-21 The Stub of Condition Row 8 should read:

"QUANTITY GR ON-HAND-A"

TTT-21 Insert "N" in table entries:

IV-2 Comment: The Evaluation Sequence indicates that in evaluating a formula the normal algebraic rules are followed, e.g., multiplication and division take precedence over addition and subtraction, and portions of the formula within the parentheses are evaluated before the portions outside the parentheses.

IV-2 Insert vertical line between Meaning and Evaluation Sequence columns.

IV-9 Double horizontal lines should appear at the bottom of the Condition rows in the first two tables.

IV-11 Double horizontal line should appear at bottom of Condition rows in first table.

IV-16 Double vertical lines should separate stub from entries in tables.

IV-19 Paragraph 1 last sentence should read:

"The condition stub may not be left blank."

IV-19 First Table, Stub of Condition Row 2 insert "Sex-Code".

IV-19 Double vertical lines should separate stub from entries in tables.

IV-20 Condition Row 2, Rule 1 should read: "LR 1".

IV-21 Condition Row 3 should read:

	1	2	3	4	5
STATUS			NEW	USED	USED

IV-28 Line 5 should read:

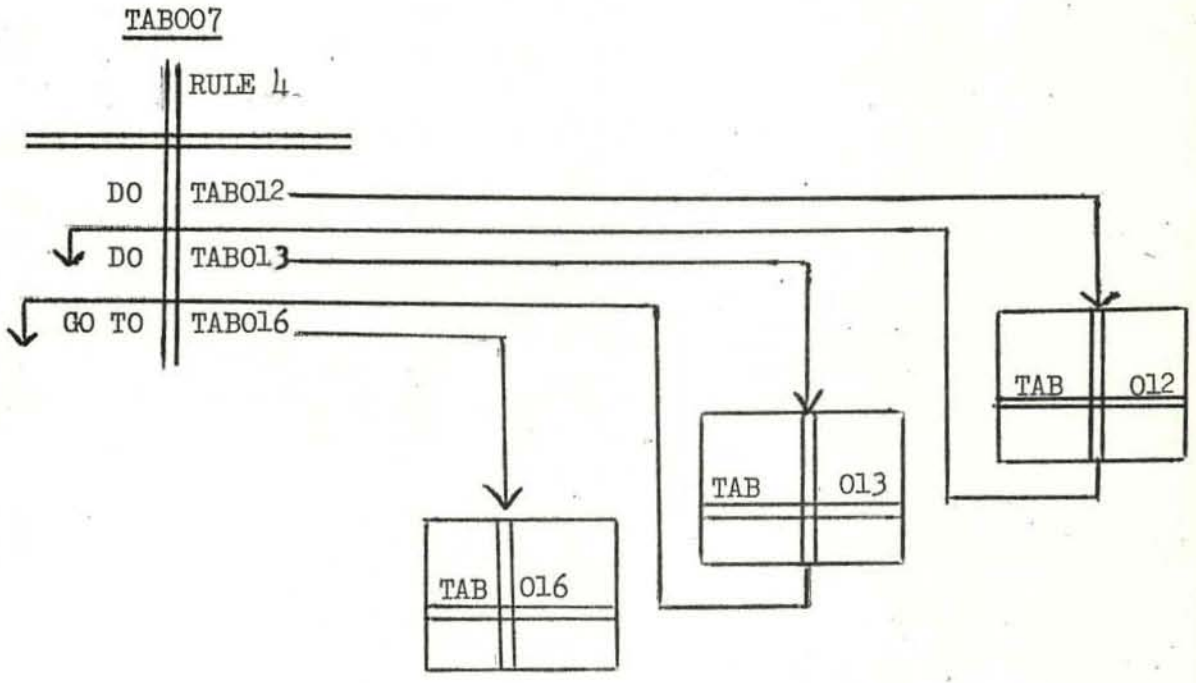
" - (hyphen) or blank The action will not be executed when the rule is satisfied."

IV-41 Answer 21 should read:

"Write EXIT: TAB-F (for example) beneath the table name FIRST-TAB, then sequence control would still pass to TAB-F when Rule 2 was satisfied."

IV-48 b. Condition-Names. Delete word "optional" after "associated data-name".

The Sequence Control Operator DO sets up a transfer to another table, and the return when that table has been executed. Therefore, in Rule 4, if CODE equals "A", TAB012 would be executed, control would then return to the next action rule which is DO TAB013. After control was returned to TAB007, TAB013 would be executed. The Sequence Control Operator GO TO would then transfer control to TAB016. In the absence of a GO TO Operator, Control is transferred to the next table as directed by the table header or executive routine.



QUESTIONS

REVISED 10/62

ANSWERS - EXAMPLE 8

1. Execute the table named in the operand and return to the following action.
2. If the code is S, move READ 1 to NAMENO, move READ4 to STKAMT, move (STKAMT + READ5) to STKAMT, DO Table 013, and GO TO Table 016.
3. The next table as specified.

DECISION TABLE TUTORIAL

USING DETAB-X

REVISED MARCH 1963

ACKNOWLEDGEMENT

The Instruction Task Force of the CODASYL Systems Development Group was composed of a team from the Education & Training, Automatic Programming and Applications Research Groups of Radio Corporation of America. In preparing the text of this Decision Table Tutorial, experience and examples have been drawn upon freely from users of decision tables, especially General Electric, International Business Machines, and Insurance Company of North America.

The members of the Instruction Task Force were

Florence Cheeseman
Thomas Jones
Regina Maas
Bruce Paterson
Mary K. Hawes, Chairman

DECISION TABLES

Introduction

Decision Tables are not new. In fact it would be difficult to find a person who has never used a table in some form. Almost everyone has used tables for insurance rates, calorie counts, or - although it may not be pleasant to think about it - income tax.

Decision tables present a useful means of analyzing and portraying logic because they provide a graphical representation of complex procedures in a way that is easy to visualize and understand. Decision tables are a method of stating conditions which must be met in order to draw conclusions and decide what action to take.

Simple decisions are made by people every day. For example, before leaving the house in the morning a man decides whether or not to wear a topcoat. He probably reasons:

if rain is forecast

or if the weather is cool

or if I will be late getting home

or if my wife insists I wear it

or if Mr. Jones is wearing one (I must keep up with the Joneses)

Another way of expressing the same situation is:

if clear is forecast
and if the weather is warm
and if I'll be home early
and if my wife doesn't insist
and if Mr. Jones isn't wearing one
then I won't wear a topcoat.

In this case all five conditions must be satisfied if he is not to wear his topcoat. A decision table uses the same principle - all conditions must be satisfied before the conclusion is drawn and the action taken. Thus the basic concept of a decision table is the if . . . and if . . . then . . . and then . . . relationship.

A decision table differs from other tables in that it is separated into two parts. The upper part is used to state the conditions (the if's) and the lower section states the actions which are to be taken as a result (the then's).

The easiest way to construct a decision table is to

(1) Define your problem

(2) List all possible conditions

(3) List the actions that are to be taken with each combination of conditions.

	1	2	3	4
Grass needs cutting?	yes	no	yes	no
Weather good?	yes	yes	no	no
Cut the grass	Sat. A.M.	-	-	-
Play golf	Sat. P.M.	Sat. A.M.	-	-

The section above the double line contains the conditions and the section below states the actions. Each set of conditions and actions is called a rule. In this example there are four rules. If this weekend the weather is good and the grass doesn't need cutting, then according to our table, (Rule 2) we can play golf Saturday morning. If, however, the grass needs cutting and the weather is bad, then according to the table (Rule 3) we can't cut the grass and we can't play golf.

Now that you have some idea of what a decision table looks like, let's look at another example and examine it in more detail.

ELEPHANT OR GIRAFFE?

	Rule 1	Rule 2	Rule 3	ELSE
Is it an animal?	Yes	Yes	Yes	
Number of legs =	4	4	4	
Nose	long*	short	long	
Neck	short	long	long	
Then it is	elephant	giraffe	hallucination	
Go to	Table E	Table G	psychiatrist	Table X

* long = ⩾3 feet

First observe that the table has a title. The title is important because it enables you to refer to one table from another by a "Go to (table title)". Also notice that as before, the table is divided into upper and lower sections by the horizontal double line. Again, the conditions are listed in the upper sections and the actions in the lower. In addition to the double horizontal line is a double vertical line which divides the table into two sections: the left side is called the stub, and the right side contains the entries. As stated before, each vertical combination of conditions and actions is called a rule.

Rule 1 is read: If the specimen is an animal, and it has 4 legs, and it has a long nose, and it has a short neck, then it is an elephant and go to the table entitled E. The underlined words are not stated in the table but are implied by the table layout. If the characteristics of the specimen do not coincide with all the conditions of Rule 1, we move to the next rule and examine the conditions listed there. If we satisfy all the conditions of a rule, we then carry out the actions of that rule. Since only one rule can be satisfied in a single pass

ANSWER

1. Rule 2 - If the specimen is an animal with 4 legs, a short nose and a long neck, it is a giraffe and go to Table G.

   Rule 3 - If the specimen is an animal with 4 legs, a long nose and a long neck, it is an hallucination and go to a psychiatrist.

   COMMENT: Since the meaning was not altered in any way, these rules were stated by the shortened version.

   Rule 2 may also be stated:

   If the specimen is an animal, and it has 4 legs, and a short nose, and a long neck, then it is a giraffe and go to Table G.

   Rule 3 may also be stated:

   If the specimen is an animal, and it has 4 legs, and a long nose, and a long neck, then it is an hallucination and go to a psychiatrist.

Decision tables can be written in two forms - limited entry and extended entry. Table CREDIT is an example of a limited entry table.

CREDIT

	1	2	3	4
Credit OK	Y	N	N	N
Pay experience favorable	-	Y	N	N
Special clearance obtained	-	-	Y	N
Approve order	X	X	X	-
Return order to sales	-	-	-	X

Rule 1 reads "If credit is OK then approve the order. Again the underlined words are implied by the format.

In limited entry form, the entire condition or action is written in the stub and the entry portion is limited to a single character. In a limited entry condition the entry contains a Y (Yes), N (No), or - (the condition is not relevant). In a limited entry action, the entry contains an X (execute) or - (do not execute).

In contrast, the extended entry form has part of the condition or action extended directly into the entry.

Both forms may be used within one table, but any one horizontal row (condition or action) must be limited or extended.

You may note from the examples that the basic concept of a single rule

in Rule 1 are not met, then try Rule 2. If no rule succeeds, it is implied that something is wrong. Now if we have considered all significant possibilities and we want to indicate "Go ahead anyway with a special routine" then we put ELSE in the last rule, or we provide for a general purpose table error routine.

Now that you have a basic understanding of the structure of decision tables, consider a more realistic and interesting example - a candy vending machine. This machine dispenses 5 cent and 10 cent candy bars and will take nickels, dimes, and quarters and give the necessary change. The machine is operated by inserting a coin and pushing a button to select the desired candy bar. Table SELECT determines if the item selected is in stock, and, if it is, refers to Table CHANGE which determines the change to be returned.

Let us follow the logic when a quarter is inserted and Item 2, a ten cent bar, is wanted; i.e.,

Item wanted = Item 2

Money Inserted = Quarter

Price = Ten

SELECT

	1	2	3	4	ELSE
Item wanted	Item 1	Item 2	Item 3	Refund	-
Any items in	Stock 1	Stock 2	Stock 3	-	-
Go to CHANGE	X	X	X	-	-
Refund coin(s)	-	-	-	X	-
Light select button	-	-	-	-	X
Stop	-	-	-	X	-
Go to SELECT	-	-	-	-	X

We begin with Table SELECT. Since Item 1 is not the ITEM WANTED, the first condition of Rule 1 is not met and we go to the next rule. Item 2 is the ITEM WANTED. The second condition asks, "Are there any items left in Stock 2?" Fortunately the machine was recently restocked so the answer is yes and the conditions of Rule 2 are satisfied. Therefore, from the action portion we are directed to the CHANGE Table.

QUESTION 2

What would happen if there is no Item 2 left in stock?

ANSWER 2

Rule 5 would be satisfied and the select button would be lighted.

CHANGE

	1	2	3	4	5	6	7	8	ELSE
Money inserted =	5	10	10	25	25	25	25	25	-
Price =	5	5	10	5	5	5	10	10	-
Balance of nickels ≥	-	1	-	-	2	4	1	3	-
Balance of dimes ≥	-	-	-	2	1	-	1	-	-
Return nickels	0	1	0	0	2	4	1	3	-
Return dimes	0	0	0	2	1	0	1	0	-
Dispense item and change	X	X	X	X	X	X	X	X	-
Stop	X	X	X	X	X	X	X	X	-
Light select	-	-	-	-	-	-	-	-	X
Go to select	-	-	-	-	-	-	-	-	X

Starting at the leftmost column of the change table entries we do not satisfy condition one until we reach Rule 4. We check Rule 4 to see if condition two is satisfied. It isn't. Therefore, we must go to the next rule and continue our search until we find a rule in which all the conditions are satisfied. In Rule 7 both the first two conditions are satisfied. Now we must check to see if the other conditions are met; i.e., if there is at least one dime and one nickel left. If there are, we continue down Rule 7 and perform the actions there. If either of the last two conditions in Rule 7 were not satisfied, we would go to the next rule and continue our search. Assuming that Rule 7 conditions are satisfied, 1 nickel and 1 dime are returned and Item 2 is dispensed. The machine stops.

QUESTION 3

Explain what would happen if the machine were out of nickels when we made

ANSWER 3

The procedure in TABLE SELECT would be the same as described on page 8. In TABLE CHANGE we would have searched until we reached the ELSE Rule. Here there are no conditions so we drop through to the action portion, light the select button and go back to TABLE SELECT to await the next item (or refund) to be selected.

There is a manual operation inferred for selecting another item or refund. Normally, the refund button is pressed when you do not receive the item selected.

WHY DECISION TABLES

The need that exists today for a standard method for defining problems has led to experimentation with Decision Tables. To date no approach seems to offer as much success -- particularly in those "sticky" problem areas. As you well know, what causes them to be "sticky" is not so much the conditions which have been defined, but rather those conditions which have not been defined.

In our computer world probably the hardest thing for us to get at has been this precise definition of the problem to be solved. This boils itself down to defining exactly what is to be done under all combinations of circumstances. With Decision Tables we may make a breakthrough. Their tabular format not only forces organized thinking along the line of exact problem definition, but also lends itself to direct input to a computer system for automatic programming.

To indicate the potential benefits to be realized from the use of a tabular form, the following statements paraphrase various user opinions:

DECISION TABLES....

....Spell out the problem clearly, systematically and precisely.

....Encourage completeness by detecting any omission in the

....Eliminate unnecessary verbal description which is prone to misinterpretation.

....Force ambiguous items to be clarified.

....Show graphically the alternatives and exceptions, and point up the meaningful relationships among variables.

....Employ common ground rules in table formation so that all persons familiar with the rules can work through a given logic structure.

....Serve as their own documentation device.

A few problems, patterned after real computer installation experience, are presented to demonstrate further the value of using Decision Tables to define problems. Attempt to work through these problems using this tabular approach.

We believe that the narrative in which these problems are described compares to many problem descriptions that you have met at one time or another in your data processing installation.

We want to warn you, however, that you will encounter difficulty in attempting to solve these problems as presented. This will be due to the

incomplete. You will note that Decision Tables point up these trouble spots as the problem is defined, while a verbal description tends to ignore them.

Some of the glaring examples of ambiguity and incompleteness that you encounter will be spelled out in the problem answers. We are sure you will be able to spot other examples which are not mentioned in the answers. It is our objective to emphasize through these examples that evasive items can be pinpointed with Decision Tables. For our purposes at the present, we are not necessarily concerned with what each and every omission is.

Problem #1

JOB #1235	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
IS COMPUTER RUN SCHEDULED	Y				
TAPES RETURNED ARE	-				
IS COMPUTER RUN COMPLETED					
IS PRINTING COMPLETED					
TAPE FACULTY	-	-	-	B25CD	P25CD
SELECT REELS LABELED					
LABEL BLANKS					
SEND TO COMPUTER REELS LABELED					
SCHEDULE PRINTER					
SEND TO PRINTER REEL LABELED					
FILE REEL LABELED					
RELEASE REEL LABELED					
GO TO JOB					

The entries appearing in the table above have been inserted in order to help you in completing the table.

Problem #1

It is the tape librarian's responsibility to keep track of the disposition of all tapes. For this reason there is a book of instructions in which all actions required of her are specified by Job Number on a pre-printed form like the one on the opposite page. When new instructions are issued, she makes whatever changes are necessary to keep this book up-to-date, in addition to actually logging all reels that have been sent out, filed, or released.

Job #1235 is a simple operation in which an output tape (B25CD) from a previous job (#1230) is further processed for Report (P25CD). Job #1235 is normally run on Tuesday, but in the event of a crowded schedule may be held over to the following day. Reel dates should be Current Date or Current Date -1 day. B25 reels are to be held for 3 cycles; P25 reels are to be released following printing. (CD is used to represent Current Date or Current Date -1 day. Its format (6 digits) is MO-DA-YR.)

Since Run #1235 produces a print tape (P25) as output, the librarian labels a blank tape which she sends along with the input (B25) to the computer room prior to the run. Upon the return to the library of both reels, the input (B25CD) is filed; P25 is scheduled for printing and then sent to the Printer Room.

Answer - Problem #1

JOB #1235	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
IS COMPUTER RUN SCHEDULED	Y	–	–	–	–
TAPES RETURNED ARE	–	B25CD P25CD	P25CD	–	–
IS COMPUTER RUN COMPLETED	N	Y	Y	–	–
IS PRINTING COMPLETED	-	N	Y	–	–
TAPE FACULTY	–	–	–	B25CD	P25CD
SELECT REELS LABELED	B25CD	–	–	–	–
LABEL BLANKS	P25CD	–	–	–	–
SEND TO COMPUTER REELS LABELED	B25CD P25CD	–	–	–	–
SCHEDULE PRINTER	–	X	–	–	–
SEND TO PRINTER REEL LABELED	–	P25CD	–	–	–
FILE REEL LABELED	–	B25CD	–	–	–
RELEASE REEL LABELED	–	B25CD -3Cycles	P25CD	B25CD	P25CD
GO TO JOB	–	–	–	1230	1235

Were there any omissions in the definition of the problem? In completing this table you were probably cognizant of the fact that all possible conditions were not stated in the narrative form. No mention was made as to what action the librarian should take in the event that a rerun of Job #1235 was required. It was merely stated that there could be a rerun. Nor was there mention made of what to do when tapes were faulty. The answer Table was completed assuming that faulty tapes (whether due to machine, manual, or program failure) would be released (or scratched) and the job rerun.

Problem #1 serves to emphasize the importance of insuring completeness in problem definition, and in picking up logic omissions. Notice in using the tabular approach that you are forced to be complete.

Problem #2

VACATION WEEK VERIFICATION ROUTINE	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	ELSE
IS EMPLOYEE NUMBER IN SEQUENCE	Y	Y	Y	N	-	
IS DISCONTINUED DATE PRESENT	N	N	N	N	Y	
EMPLOYMENT YEAR IS	LE CURR YR-5YRS	LR CURR YR	EQ CURR YR	-	-	
EMPLOYMENT YEAR IS	-	GR CURR YR-5YRS	-	-	-	
VACATION WEEKS EARNED UNEQUAL TO	3	2	1	-	-	
DO VACATION CALCULATION						
WRITE ERROR						
GO TO TABLE						

LR: LesseR than (Less Than)

GR: GReater than

EQ: EQual to

Problem #2:

Because of the high activity on one of the Personnel tape files and the need for valid input, it was deemed necessary to incorporate a verification routine in the first phase of the program. Fields to be checked are Discontinued Date, Employee Number, Employment Year and Vacation Weeks Earned.

No record is to be examined further if it is found to contain a Discontinued Date.

The records on this tape are to be in sequence by Employee Number, and there must be an Employment Year in order to check the accuracy of the field Vacation Weeks Earned. The correct number of Vacation Weeks Earned must always appear in that field.

The procedure followed for calculating Vacation Weeks Earned is:

1. If Employment Year is equal to current year, then employee has earned 1 week vacation.
2. If Employment Year is equal to or less than Current Year minus 5 years, then employee has earned 3 weeks vacation.
3. If Current Year minus employment year is greater than zero and less than 5 years, then employee has earned 2 weeks vacation.

When the record is verified, proceed to Process Table.

Construct the Decision Table. Although omissions have been made in the verbal description, complete the table as far as you are able. When finished,

Answer - Problem #2

VACATION WEEK VERIFICATION ROUTINE	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	ELSE
IS EMPLOYEE NUMBER IN SEQUENCE	Y	Y	Y	N	-	-
IS DISCONTINUED DATE PRESENT	N	N	N	N	Y	-
EMPLOYMENT YEAR IS	LE CURR YR-5YRS	LR CURR YR	EQ CURR YR	-	-	-
EMPLOYMENT YEAR IS	-	GR CURR YR-5YRS	-	-	-	-
VACATION WEEKS EARNED UNEQUAL TO	3	2	1	-	-	-
DO VACATION CALCULATION	X	X	X	-	-	-
WRITE ERROR	X	X	X	X	X	-
GO TO TABLE	PROCESS	PROCESS	PROCESS	READ	READ	PROCESS

Did you notice that no mention was made of what to do if there was an error? We assumed that the programmer would want all errors written out for checking purposes, so we added this action to our list.

You will also note that although you are told not to examine records containing a discontinued date, you are not told what to do with them. Again we incorporated the actions we thought would probably be followed.

Normally, a verbal description requires more than one reading to determine just what steps are to be taken at what time, and then, as we have seen, misinterpretation frequently occurs.

The problem, as narrated, did not spell out explicitly that when the vacation weeks field is not valid you are to calculate it. This was again merely implied.

Problem #3

For a smooth-running system, it is very important that OPERATIONS be notified of any and all actions that might have to be taken by them during a computer run.

Programming furnished the operating procedures for Payroll Job #0565. By using a Decision Table see if you can determine if the following instructions were adequate.

Problem #3

JOB #0565
PAYROLL OPERATING PROCEDURES

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
IS HALT AT LOCATION 0267	Y	Y	Y	Y	N
RUN DAY IS	WED	WED	FRI	OTHER	–
IS CONTROL INFORMATION RECEIVED	Y	N	–	–	–
STORE DATE AT LOCATION 0398					
ENTER CONTROL INFORMATION					
CONTINUE PROCESSING					
RERUN					
CONTACT PROGRAMMER					
FOLLOW SPECIAL INSTRUCTIONS					

Problem #3 (cont'd)

JOB #0565

PAYROLL OPERATING PROCUEDURES

Halt 1 (a program control point) should always occur and at location 0267.

1. Wednesday (each week)
   a. Store Friday's date (6 digits) at location 0398 MO-DA-YR
   b. Store Control Information, furnished by Payroll Division, as directed by typewriter tear sheet. If control information is discovered to be missing, investigate and rerun as soon as possible.
   c. Continue Processing - Hit Start Key.

2. Friday (each week)
   a. Store Wednesday's date at location 0398 MO-DA-YR
   b. No Control Information.
   c. Continue Processing - Hit Start Key.

Missing Procedures:

1. If the run was done on any day other than Wednesday or Friday, then what?

   (We assumed that there were SPECIAL instructions to which we could refer.)

2. What should the operator do in the event that a Halt occurred at a location other than 0267?

   (In this case we contacted the Programmer.)

3. Was there any confusion regarding which Friday and which Wednesday?

Our Table looked like the following:

Answer - Problem #3

JOB #0565

PAYROLL OPERATING PROCEDURES	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
IS HALT AT LOCATION 0267	Y	Y	Y	Y	N
RUN DAY IS	WED	WED	FRI	OTHER	–
IS CONTROL INFORMATION RECEIVED	Y	N	–	–	–
STORE DATE AT LOCATION 0398	NEXT FRI DTE		NEXT WED DTE		
ENTER CONTROL INFORMATION	X				
CONTINUE PROCESSING	X		X		
RERUN		X			
CONTACT PROGRAMMER					X
FOLLOW SPECIAL INSTRUCTIONS				X	

In all problems thus far, we have tried to emphasize how Decision Tables point up incomplete definition, for this is the biggest problem with which we are faced in the computer industry.

Also evident is the ready application that may be made of Decision Tables in 3 of the main areas of a computer installation:

1. manual - such as tape handling,
2. programming logic, and
3. operating procedures.

Problem #4

The following example serves to illustrate what might happen if a change is proposed to an existing system. Such a change may or may not have a drastic effect. However, this can be readily determined if originally a Decision Table was used to state the logic flow. Sometimes it pays to evaluate whether or not a change should be made by determining what must be sacrificed to make it. Since a Decision Table is its own documentation device, it serves to point out quickly what areas would be affected.

Problem #4

PENALTY BILLING PROCEDURE	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6	RULE 7	RULE 8	ELSE
MONTHS OVERDUE	GE 3	GE 3	GE 6	GE 6					
MONTHS OVERDUE	LR 6	LR 6							
UNPAID BALANCE GR $500	N	Y	N	Y					
SPECIAL ACCOUNT									
CALCULATE 1% PENALTY CHARGE	X		X						
CALCULATE 2% PENALTY CHARGE		X		X					
POST TO OPEN CHARGE	X	X	X	X					
POST TO INVOICE	X	X	X	X					
GIVE TO COLLECTION AGENCY			X	X					
GO TO INVOICE TOTAL RTN	X	X	X	X					
WRITE ERROR									X
GO TO NEXT ACCOUNT									X

Problem #4 (cont'd)

In addition to regular billings, the Girard Company collects a penalty charge from all accounts whose payment is in arrear 3 or more months. For this reason a PENALTY BILLING PROCEDURE is in existence.

A charge of 1% is made on an upaid balance up to and including $500.00, and for over $500.00, 2% is charged. This charge is added to the OPEN charge record and included on the invoice. After this penalty charge is calculated, the next step is an Invoice Totaling Routine.

When an account is overdue 6 or more months, it is turned over to a collection agency. Company billing continues, however.

This procedure has been in operation for a year. At the present time there is under consideration a change which will permit special billing privileges to be given to specified accounts. These accounts are to be billed every 2 months instead of every month which will delay their penalty charges and listing with a Collection Agency by one month.

What changes must be made to the logic flow to incorporate this plan? Make these changes to the table on the opposite page.

Answer - Problem #4

PENALTY BILLING PROCEDURE	RULE 1	RULE 2	RULE 3	RULE 4	RULE 5	RULE 6	RULE 7	RULE 8	ELSE
MONTHS OVERDUE	GE 3	GE 3	GE 6	GE 6	GE 4	GE 4	GE 7	GE 7	
MONTHS OVERDUE	LR 6	LR 6			LR 7	LR 7			
UNPAID BALANCE GR $500	N	Y	N	Y	N	Y	N	Y	
SPECIAL ACCOUNT	N	N	N	N	Y	Y	Y	Y	
CALCULATE 1% PENALTY CHARGE	X		X		X		X		
CALCULATE 2% PENALTY CHARGE		X		X		X		X	
POST TO OPEN CHARGE	X	X	X	X	X	X	X	X	
POST TO INVOICE	X	X	X	X	X	X	X	X	
GIVE TO COLLECTION AGENCY			X	X			X	X	
GO TO INVOICE TOTAL RTN	X	X	X	X	X	X	X	X	
WRITE ERROR									X
GO TO NEXT ACCOUNT									X

You can see that the modifications to the above program, that are required to handle the proposed change, are minor. Bear in mind, however, that an additional change must be made to a table that logically precedes this one. This is in order to incorporate an additional test to prevent Special Accounts which are overdue only 3 months from being processed by the Penalty Billing Table. Without this change such accounts would be processed by the ELSE column.

It is well to remember that no account should enter this table unless

CONCLUSION

Decision Tables tend to simplify the definition of complex problems by concentrating on one small area at a time. Furthermore, the discipline of the technique forces you to include in a table all possible paths, eliminating ambiguities that tend to predominate in a number of other methods where it is not easy to determine if all possible paths have been considered.

Decision Tables are two-dimensional in nature. They permit us to fully express and consider both the sequential and parallel aspects of logic. This is something that techniques such as flow charts and symbolic logic do not do --- they are primarily sequential in nature.

The ramifications of a change to an existing system can be easily analyzed, and thus logical conclusions can be drawn. The ability to evaluate a possible change and also to effect that change are of the greatest importance.

There is an indication that Decision Tables afford in a system design something that no other approach has yet been able to offer - flexibility. This is apparent not only in the individual table, but also in the ability to link tables together.

DECISION TABLE PROBLEMS

This Chapter is devoted to a series of exercises designed to introduce the concepts of Decision Tables. The material consists of examples of partial tables, a discussion about one or more elements in each table and several questions to test your comprehension of the problem. After completing the questions, you will find the correct answers on the following page.

Failure to answer all the questions correctly could indicate a need to review the previous exercises.

		Rule 1	Rule 2	Rule 3
TYPCOD	EQ	1	2	3
SET MODEL	EQ	"CNVTBL"	"HRDTOP"	"STAWGN"

EXAMPLE 1

The table on the opposite page, if written in English, would appear as follows:

If the type code of the car body is 1, then the model is a convertible.

If the type code of the car body is 2, then the model is a hardtop.

If the type code of the car body is 3, then the model is a station wagon.

This table has a single condition row which appears above the double horizontal line.

(If) TYPCOD EQ 1 (or) 2 (or) 3

It also contains a single action row, which appears below the double horizontal line.

SET MODEL EQ "CNVTBL" (OR) "HRDTOP" (OR) "STAWGN"

There are three decision rules in this table. In this case, the decision rule is made up of one condition entry and one action entry.

Rule 1 (IF) TYPCOD EQ 1 (Then) SET MODEL EQ "CNVTBL".

Rule 2 (IF) TYPCOD EQ 2 (Then) SET MODEL EQ "HRDTOP".

Rule 3 (IF) TYPCOD EQ 3 (Then) SET MODEL EQ "STAWGN".

QUESTIONS

1. The information appearing above the double horizontal line constitutes the _____.

2. The information appearing below the double horizontal line constitutes

ANSWER - EXAMPLE 1

1. condition row.
2. action row.
3. three

		Rule 1	Rule 2	Rule 3
TYPCOD	EQ	1	2	3
SET MODEL	EQ	"CNVTBL"	"HRDTOP"	"STAWGN"

EXAMPLE 2

Condition statements and action statements are made up of operands and operators. An operand is that information upon which the action is to take place. One form that the operand may take is a Data Name. A Data Name is a unique name assigned to a data field. In Example 2, TYPCOD and MODEL are both Data Names. The data field TYPCOD may contain the values 1, 2, or 3. As a result of this table, the data field MODEL will contain "CNVTBL", "HRDTOP" or "STAWGN".

Another type of operand is the literal. A literal is the actual value of the operand to be used in a statement as opposed to a data field that will contain the value. In Example 2, the values 1, 2, 3, "CNVTBL", "HRDTOP" and "STAWGN" are literals. Literals are of two types numeric and non-numeric. A numeric literal may contain only numeric quantities using the digits 0 thru 9. A non-numeric literal may contain any allowable character and is always enclosed in quote marks. In Example 2, the values 1, 2, and 3 are numeric literals; and "CNVTBL", "HRDTOP" and "STAWGN" are non-numeric literals.

QUESTIONS - EXAMPLE 2

1. List the data names in example 2.
2. List the literals.
3. A data field is referred to by name, whereas a _____ is referred to by its actual value.
4. Non-numeric literals are always surrounded by _____ _____.

ANSWERS - EXAMPLE 2

1. TYPCOD, MODEL.
2. 1, 2, 3, "CNVTBL", "HRDTOP", "STAWGN".
3. Literal.
4. Quotation Marks.

		Rule 1	Rule 2	Rule 3*
PAID	EQ	"N"	"G"	
SET PCHNAM	EQ	RDNAME	RDNAME	–
SET PCHAMT	EQ	RDNET	RDGRS	–
WRITE		CDPCH	CDPCH	–
READ		CDREAD	CDREAD	–

EXAMPLE 3

This problem involves a utility billing procedure. The bill stub that a customer returns with his payment has punched in it a net amount and a gross amount. Before the bill stub goes into the computer system, the letter N or G is punched into the card to indicate which amount was paid; it is then unnecessary to punch the amount of the payment.

In English this table reads:

Rule 1: If the data field PAID contains "N" (for net), transfer the customer's name from input card (RDNAME) to output card (PCHNAM), transfer net amount (RDNET) to amount field (PCHAMT), write a card, and read another card.

Rule 2: If the data field PAID contains "G" (for gross), transfer the customer's name from input card (RDNAME) to output card (PCHNAM), transfer gross amount (RDGRS) to amount field (PCHAMT), write a card, and read another card.

An operator describes the operations to be performed on the operands. In this case SET...EQ is a data manipulation operator. The action it describes is the transfer of the contents of data name RDNAME to data name PCHNAM; and the contents of data names RDNET or RDGRS to data name PCHAMT depending on which condition is satisfied.

QUESTIONS

ANSWERS - EXAMPLE 3

1. Transfer

2. "N", "G".

		Rule 1	Rule 2	Rule 3
PAID	EQ	"N"	"G"	
SET PCHNAM	EQ	RDNAME	RDNAME	–
SET PCHAMT	EQ	RDNET	RDGRS	–
WRITE		CDPCH	CDPCH	–
READ		CDREAD	CDREAD	–

EXAMPLE 4

In order to compile any program, there must be a detailed description of the data elements referenced in the program. Each data field must be described. In addition data fields are grouped into units called Records. For example a record could be one time card or one stock number record. A file is the total number of these records associated with an input or output device.

In this example Read and Write are Input/Output Operators. They require file name operands. When the READ operation is executed, a record will be read from the file name CDREAD. This file happens to be associated with the card reader. The write operator will trigger the punching of a card. The operand CDPCH specifies the file name and record to be punched.

QUESTIONS

1. List the Data names in this table.
2. List the file names.
3. The operand in a READ or WRITE action must be a _____.
4. A file is composed of _____.

ANSWER - EXAMPLE 4

1. PAID, PCHNAM, RDNAME, PCHAMT, RDNET, RDGRS.
2. CDPCH, CDREAD.
3. File name.
4. Records

	Rule 1	Rule 2	Rule 3	Rule 4
AGE	LE 25	LE 25	GR 25	GR 25
SEX EQ	"M"	"F"		
ACDNTS			EQ 0	GR 0
SET RATE EQ RATE	+ RSKFAC		- SPRFAC	
SET PURATE EQ	RATE	RATE	RATE	RATE
WRITE RATECD	X	X	X	X

EXAMPLE 5

NOTE: See TV-31 last paragraph regarding the Stub of Action Row 1

This table is an example of multiple condition rows. Rule 1 reads as follows: If Age is less than or equal to 25 and if Sex is Male (M)... The blank in the third condition row indicates that the condition is not relevant to rule 1. (A blank is permissable as well as the hyphen to indicate irrelevant.)

In addition we have introduced three of the conditional operators: EQ, LE, GR. There are six available conditional operators:

Equal	(EQ)	Not Equal to	(NE)
Less than (LesseR than)	(LR)	Less than or Equal to	(LE)
Greater than	(GR)	Greater than or Equal to	(GE)

These operators are to be understood in the sense of a question to which the answer can be only yes or no. The first condition in Rule 1 thus reads: "Is the age of the applicant less than or equal to 25 years?" If the answer is "yes", the condition is satisfied.

QUESTIONS

1. State Rule 3 in English.
2. List the literals.
3. List three data names in the condition area and four in the action area.

ANSWERS - EXAMPLE 5

1. Rule 3: If the applicant is over 25 and has had no accidents, subtract the special rate factor from the previously computed rate, move the new rate to PURATE and Write a record to RATECD file.

2. "M", "F", 0, 25.

3. ACDNTS, SEX, AGE: RATE, PURATE, RSKFAC, SPRFAC.

	Rule 1	Rule 2	Rule 3	Rule 4
AGE	LE 25	LE 25	GR 25	GR 25
SEX EQ	"M"	"F"		
ACDNTS			EQ 0	GR 0
SET RATE EQ RATE	+ RSKFAC		- SPRFAC	
SET PURATE EQ	RATE	RATE	RATE	RATE
WRITE RATECD	X	X	X	X

EXAMPLE 6

Action Row 1 demonstrates the use of SET operator in conjunction with a formula, (SET data-name EQ to formula). In this case the formula (RATE + RSKFAC) and (RATE - SPRFAC) is split between the stub and the entry. In Rule 1 the contents of RSKFAC are to be added to RATE if the conditions are satisfied, in Rule 3 the contents of SPRFAC are to be subtracted from the field RATE.

In this example we introduce a new format in the third action row. Since the same action is to be carried out for each rule, we write the complete action specification in the stub part of the table (to the left of the vertical double line) and place X's in the entry part to indicate in which rules the action is to be executed. This is called a limited entry; the format of the previous tables is called extended entry, since the operand is extended into the entry portion of the table. The second action could also have been written in limited entry fashion, with SET PURATE EQ RATE entirely in the stub, and X's in the "OP" fields of the entries.

QUESTIONS

1. List the file names in this table.
2. Could action 2 have been written in limited entry form.
3. Which condition row is in extended entry form.
4. State Rule 4 in English.

ANSWERS - EXAMPLE 6

1. RATECD.
2. Yes.
3. All three.
4. If Age is greater than 25 and Accidents is greater than 0,

   Move RATE to PURATE and Write a record to RATECD file.

			Rule 1	Rule 2	Rule 3
	STKTYP	EQ	01	02	03
	SHARES	GR 100000	Y		N
SET	TYPE	EQ	"INDIV"	"BANK"	"BROKER"
MOVE	STKVAL	TO	VALUE 1	VALUE 2	
WRITE	REPORT		X	X	X

EXAMPLE 7

This table demonstrates a limited entry in the condition row. Previously we had an example of a limited entry in the action section. In order for Rule 1 to be satisfied, the answer to "Is the number of shares greater than 100000" must be yes. In order to satisfy Rule 3, the answer must be no.

The data manipulation operator MOVE is also introduced in this example. The operator MOVE is very similar to SET except that the data movement is reversed. In the following example the data movement is from operand-2 to operand-1:

SET operand-1 EQ operand-2.

In the next example, the data movement is from operand-1 to operand-2.

MOVE operand-1 to operand-2.

The difference in usefulness of the two operators may be stated as follows:

SET...EQ is valuable when one of several quantities is to be moved to a single data field;

MOVE...TO is valuable when a single quantity is to be moved to one of several data fields.

QUESTIONS

1. What is the difference between SET and MOVE?
2. Must there be a Y or N in each condition entry space of a limited

ANSWERS - EXAMPLE 7

1. SET transfers the values of the second operand to the first;

   MOVE transfers the first to the second.

2. No, it may be blank.

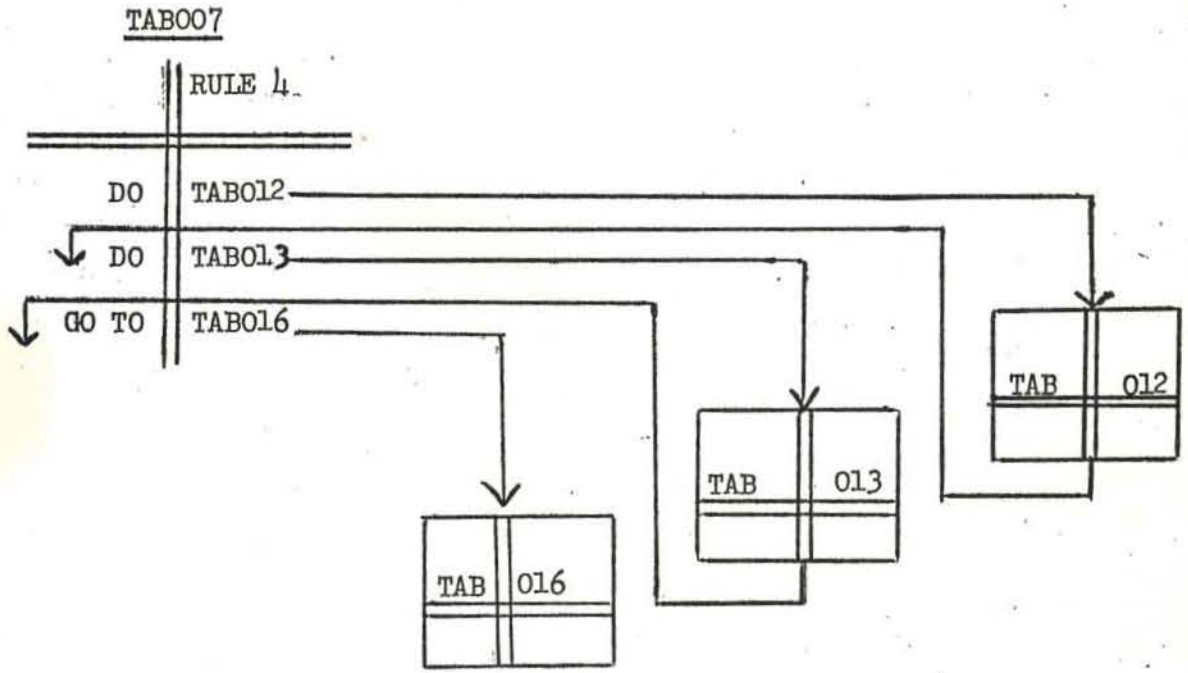
3. Data names.

4. Rule 2: If the stock type is 02, set TYPE equal to "BANK", move stock value to VALUE 2, and print the report names REPORT.

TAB007

CODE EQ	Rule 1	Rule 2	Rule 3	Rule 4
	"N"	"B"	"S"	"A"
SET NAMENO EQ READ1	–	X	X	X
SET BNDAMT EQ READ2	–	X	–	X
SET BNDAMT EQ BNDAMT	–	+ READ	–	+ READ3
SET STKAMT EQ READ4	–	–	X	X
SET STKAMT EQ STKAMT	–	–	+ READ5	+ READ5
DO	–	TAB012	TAB013	TAB012
DO	–	–	–	TAB013
GO TO TAB016	–	X	X	X

The Sequence Control Operator DO sets up a transfer to another table, and the return when that table has been executed. Therefore, in Rule 4, if CODE equals "A", TAB012 would be executed, control would then return to the next action rule which is DO TAB013. After control was returned to TAB007, TAB013 would be executed. The Sequence Control Operator GO TO would then transfer control to TAB016. In the absence of a GO TO Operator, Control is transferred to the next table as directed by the table header or executive routine.



QUESTIONS

1. What action is specified by a DO?
2. State rule 3 in English

ANSWERS - EXAMPLE 8

1. Execute the table named in the operand and return to the following action.
2. If the code is S, move READ 1 to NAMENO, move READ4 to STKAMT, move (STKAMT + READ5) to STKAMT, DO Table 013, and GO TO Table 016.
3. The next table as specified.

TABO01	1	2	3	4	5	6	ELS
STOCK-NR-A EQ STOCK-NR-C							
STOCK-NR-A LR STOCK-NR-C							
STOCK-NR-A GR STOCK-NR-C							
CHANGE-CODE EQ "REC"							
CHANGE-CODE EQ "SHIP"							
CHANGE-CODE EQ "ADJUST"							
CHANGE-CODE EQ "NEW-ITEM"							
QUANTITY GR ON-HAND-A							
MOVE (ON-HAND-A + QUANTITY) TO ON-HAND-A							
MOVE (ON-HAND-A - QUANTITY) TO ON-HAND-A							
SET CHANGE-CODE EQ "BACK-ORDER"							
WRITE SHIP-ORDER FROM CHANGE							
WRITE NEW-MASTER FROM MASTER							
WRITE NEW-MASTER FROM CHANGE							
READ CHANGE							
READ MASTER							
DO ERROR-ROUTINE							
GO TO TABO06							

The purpose of this exercise is to complete the limited entry table on the opposite page. The problem is a simplified file maintenance application. Except for starting and ending procedures, which are taken care of in additional tables, the entire procedure is handled in this single table. The file names and relevant data fields are listed below:

	File Names	Data Fields
Inputs:	1. Master file	1. Stock-Nr-A 2. On-Hand-A
	2. Change file	1. Stock-Nr-C 2. Quantity 3. Change-Code
Outputs:	1. New-Master file	(Same as Master file)
	2. Ship-Order file	(Same as Change file)

When an item from the Master file does not have a Change file record to apply against it, write the Master file record to the New-Master file.

When an item from the Change file does not correspond to a Master file record, it must be a "NEW-ITEM". In this case, create a New-Master file record from the Change file record.

When the Stock Number of the Master file agrees with the Stock Number of the Change file, update the Master file record as follows:

a. If the Change-Code is equal to "REC", adjust quantity on hand.

b. If the Change-Code is equal to "SHIP", and the quantity requested is available, adjust ON-HAND-A and write Shipping Order. If the quantity requested is not available, modify the Change-Code to read "NEW-ERROR".

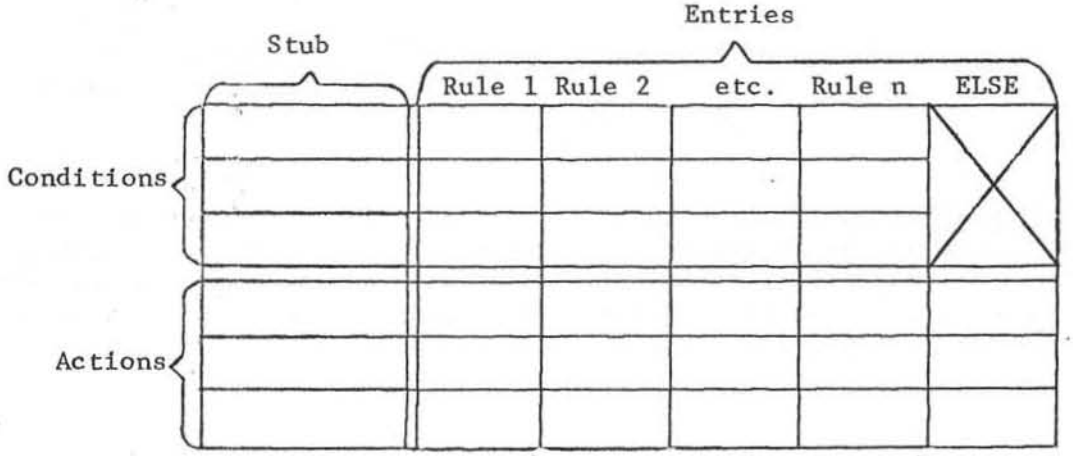
TABOO1	1	2	3	4	5	6	ELSE
STOCK-NR-A EQ STOCK-NR-C	Y	Y	Y	Y	N	N	
STOCK-NR-A LR STOCK-NR-C	N	N	N	N	Y	N	
STOCK-NR-A GR STOCK-NR-C	N	N	N	N	N	Y	
CHANGE-CODE EQ "REC"	N	N	Y	N		N	
CHANGE-CODE EQ "SHIP"	Y	Y	N	N		N	
CHANGE-CODE EQ "ADJUST"	N	N	N	Y		N	
CHANGE-CODE EQ "NEW-ITEM"	N	N	N	N		Y	
QUANTITY GR ON-HAND-A	N	Y					
MOVE (ON-HAND-A + QUANTITY) TO ON-HAND-A			X	X			
MOVE (ON-HAND-A - QUANTITY) TO ON-HAND-A	X						
SET CHANGE-CODE EQ "BACK-ORDER"		X					
WRITE SHIP-ORDER FROM CHANGE	X	X					
WRITE NEW-MASTER FROM MASTER					X		
WRITE NEW-MASTER FROM CHANGE						X	
READ CHANGE	X	X	X	X		X	
READ MASTER					X		
DO ERROR-ROUTINE							X
GO TO TABOO6	X	X	X	X	X	X	X

CONDITIONS AND ACTIONS

BASIC SPECIFICATIONS

This section of the manual will describe in detail a basic set of Condition and Action functions that are appropriate for use in decision tables. These functions are, in fact, a subset of those described in the DETAB-X Manual, with minor modifications. The sample problems in the following section of the manual will only use functions from this basic set, and a condensed reference list of the functions will be found at the end of this manual.

The structure of the decision tables used in the example will be as shown here:



Notes:

1. The combination of conditions that define each rule must not be

3. If none of the rules can be satisfied then the actions associated with the ELSE column will be executed. The ELSE column may be omitted if desired, but if there are any circumstances under which no rule will be satisfied then the user should realize that no actions will be taken under those circumstances.

4. The use of a formula as an operand in conditions or actions allows arithmetic operations to be performed. The normal conventions are used within this manual, as indicated below:

Arithmetic Operators	Meaning	Evaluation Sequence
+	plus	1. **
-	minus	2. * and /
*	multiplied by	3. + and - unless modified by the presence of parentheses (and).
/	divided by	
**	exponentiated by	

(COMMENT: The Evaluation Sequence indicates that in evaluating a formula the normal algebraic rules are followed, e.g., multiplication and division take precedence over addition and subtraction, and portions of the formula within the parentheses are evaluated before the portions outside the parentheses.)

I. CONDITIONS

In the following discussion, three types of conditions will be introduced:

a. Relation Tests

b. Condition-Names

c. End-File Tests

The use of each type will be shown both for limited entry and extended entry condition rows.

a. Relation Tests

A Relation Test is a comparison to determine if the value of an item is equal to, greater than, or less than, the value of another item. Its format is:

operand-1 operator operand-2

The permissible operators in relations, with their equivalent meanings, are:

Relation Operator	Meaning
GR	GReater than
LR	LesseR than
EQ	EQual to
LE	Less than or Equal to
GE	Greater than or Equal to
NE	Not Equal to

Examples:

DEDUCTION GR NET-PAY means "if Deduction is greater than Net-Pay"

(3 * P ** 2 + 5 * Q ** 2) / R LE 45 means "if the sum of 3 times P squared, and 5 times Q squared, all divided by R, is less than or equal to 45"

Note that the word IF is implied because the relation test is written in the Condition part of a decision table.

In a limited entry condition row the whole relation test is written in the condition stub. The condition entry columns may contain:

Entry	Meaning
Y	The condition must be satisfied if the rule (column) is to be satisfied.
N	The condition must not be satisfied if the rule is to be satisfied.
-or blank	The condition is irrelevant for this rule.

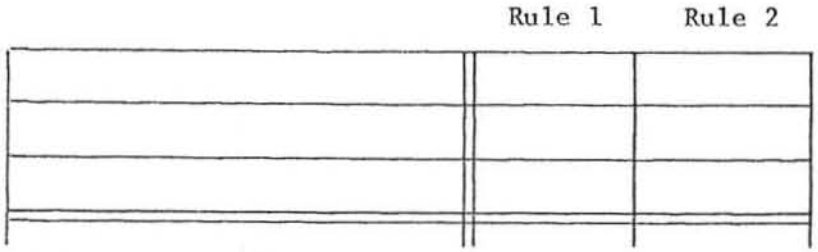
Example:

Condition Stub	Condition Entries	
	Rule 1	Rule 2
DEDUCTION LR NET-PAY	Y	N
PAY-CODE EQ "M"	N	-

In the above table, Rule 1 is satisfied if Deduction is less than Net-Pay, and Pay-Code is not an M; Rule 2 is satisfied if Deduction is not less than Net-Pay, regardless of the value of Pay-Code.

Question 1

Write conditions in the table below so that Rule 1 is satisfied if PART-NO is equal to SOLD-PART, and BALANCE is less than SOLD-AMT, and RES-CODE is not equal to an R. Use a Y in the entry wherever possible.



Answer on page 6

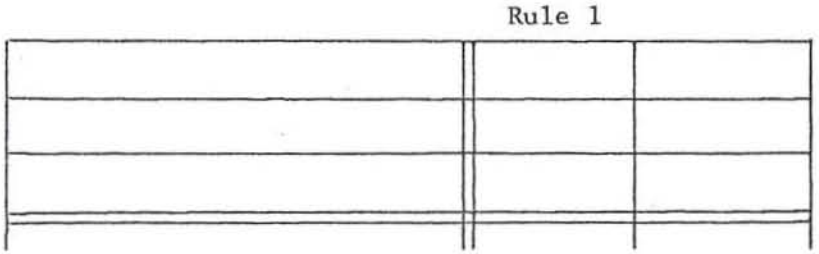
Question 2

Write entries in the preceding table so that Rule 2 is satisfied if PART-NO is equal to SOLD-PART, and BALANCE is not less than SOLD-AMT, and RES-CODE is equal to "R".

Answer on page 7

Question 3

Rewrite in the table below the conditions for question 1 in such a way that the entries under Rule 1 can contain an N wherever possible.



Answer on page 8

Answer 1

	Rule 1	Rule 2
PART-NO EQ SOLD-PRT	Y	
BALANCE LR SOLD-AMT	Y	
RES-CODE NE "R"	Y	

Now turn back to question 2.

Answer 2

	Rule 1	Rule 2
PART-NO EQ SOLD-PART	Y	Y
BALANCE LR SOLD-AMT	Y	N
RES-CODE NE "R"	Y	N

Now turn back to question 3.

Answer 3

	Rule 1	
PART-NO NE SOLD-PART	N	
BALANCE GE SOLD-AMT	N	
RES-CODE EQ "R"	N	

In an extended entry condition row a relation test is split between the condition stub and the condition entry in one of the following ways:

	Condition Entries		
Condition Stub	Rule 1	Rule 2	Rule 3
operand-1 operator	operand-2a	operand-2b	etc.
operand-1	operator-a operand-2a	operator-b operand-2b	etc.

As before, an entry may contain — or blank to indicate that the condition is not relevant to a particular rule.

Example:

	Condition Entries		
Condition Stub	Rule 1	Rule 2	Rule 3
ORDER-CODE EQ	"M"	"S"	—
HP-RATING	EQ MAX-HP	LR MAX-HP	GR MAX-HP

Question 4

Write conditions in the table below in extended form so that Rule 1 is satisfied if PAY-CODE is W and PAY-RATE is less than 65, Rule 2 is satisfied if PAY-CODE is W and PAY-RATE is less than 90 but not less than 65, Rule 3 is satisfied if PAY-CODE is W and PAY-RATE is not less than 90 and Rule 4 is satisfied if PAY-CODE is not equal to W.

	Rule 1	Rule 2	Rule 3	Rule 4

Answer 4

	Rule 1	Rule 2	Rule 3	Rule 4
PAY-CODE	EQ "W"	EQ "W"	EQ "W"	NE "W"
PAY-RATE LR	65	90	—	—
PAY-RATE GE	—	65	90	—

Question 5

Remembering that each rule must be independent of all other rules, so that only one rule can be satisfied at any one time, what is wrong with the following example?

	Rule 1	Rule 2	Rule 3	ELSE
PAY-CODE EQ "H"	Y	Y	Y	
OT-HOURS GR	20	10	0	

Answer on page 12

Question 6

What happens in the above example if PAY-CODE is an H and OT-HOURS equals zero?

Answer on page 13

Question 7

Rewrite the above example to eliminate the errors.

	Rule 1	Rule 2	Rule 3	ELSE

Answer on page 14

Answer 5

If Rule 1 is satisfied, Rules 2 and 3 are also satisfied.

If Rule 2, but not Rule 1, is satisfied, Rule 3 is also satisfied.

Now turn back to question 6.

Answer 6

The actions associated with the ELSE column are executed.

Now turn back to question 7.

Answer 7

	Rule 1	Rule 2	Rule 3	ELSE
PAY-CODE EQ "H"	Y	Y	Y	
OT-HOURS GR	20	10	0	
OT-HOURS LE		20	10	

b. Condition-Names

The second type of condition to be discussed here is the Condition-Name. Suppose that a data item whose data-name is MARITAL-STATUS can possess the values 1, 2, 3, or 4, indicating respectively Single, Married, Widowed, or Divorced. In order to state the condition "If Married", the word MARRIED may be used in the decision table instead of the relation test MARITAL-STATUS EQ 2. This form of condition is known as a Condition-Name; it is more compact and often is more readily understood by users of the table than the equivalent relation test would be.

In the example just mentioned there are actually four condition-names associated with the data-item MARITAL-STATUS:

Value	Condition-Name
1	SINGLE
2	MARRIED
3	WIDOWED
4	DIVORCED

It is not necessary to define in the decision tables themselves the condition-names that are to be associated with a given data-name. This definition would normally be given in that part of the problem-statement in which the data items entering the problem are described, that is to say the Data Descriptions. The entries for the above example might appear in the Data Descriptions as follows:

In a limited entry condition-row the condition-name is written in the condition stub. The condition entry columns may contain Y, N, or blank; the meanings of these entries are exactly as explained previously for relation tests.

Example:

Suppose the following data-names and condition-names have been defined:

MARITAL-STATUS		SEX-CODE	
SINGLE	= 1	MALE	= "M"
MARRIED	= 2	FEMALE	= "F"
WIDOWED	= 3		
DIVORCED	= 4		

	Rule 1	Rule 2	Rule 3
SINGLE	Y	Y	N
MALE	Y	N	

In the above table Rule 1 is satisfied if MARITAL-STATUS equals 1 and SEX-CODE is an M; Rule 2 is satisfied if MARITAL-STATUS equals 1 and SEX-CODE is not an M; Rule 3 is satisfied if MARITAL-STATUS is not equal to 1, regardless of the value of SEX-CODE.

Question 8

Using the condition-names defined above, write conditions in the limited entry table below so that Rule 1 is satisfied for Single Males whose AGE is less than 25; Rule 2 is satisfied for Single Males whose AGE is not less than

Answer 8

	Rule 1	Rule 2	Rule 3	Rule 4
SINGLE	Y	Y	Y	N
MALE	Y	Y	N	
AGE LR 25	Y	N		

Question 9

In the following table, which categories of persons will cause the ELSE actions to be executed?

	Rule 1	Rule 2	Rule 3	Rule 4	ELSE
SINGLE	Y		N		X
DIVORCED		Y	N	N	
FEMALE	Y	Y	Y	N	

Answer on page 18

Answer 9

Divorced Males. The first three rules exhaust all the possibilities of MARITAL-STATUS for Females, and the 4th Rule is satisfied for all Males who are not Divorced.

In an extended entry condition-row any of the condition-names associated with a particular data-name may appear in the successive entries of a row. A blank may be left if the value of the data-name is not relevant to a particular rule. The condition stub may not be left blank.

Example:

	Rule 1	Rule 2	Rule 3	ELSE
MARITAL-STATUS	SINGLE	SINGLE	MARRIED	X
SEX-CODE	MALE	FEMALE		

Note that within a single condition row all condition-names must be associated with the same data-name.

Question 10

Find the error in the following example:

	Rule 1	Rule 2	Rule 3	Rule 4
MARITAL-STATUS	SINGLE	SINGLE		DIVORCED
SEX-CODE	MALE	FEMALE	WIDOWED	

Answer 10

WIDOWED is not associated with the same data-name as MALE and FEMALE, and should not appear in row 2. It would be legitimate in row 1, however.

Question 11

Assuming the following definitions have been made:

BODY-TYPE	AGE-YRS	STATUS
SEDAN = 1	THIS-YEAR = 0	NEW = "N"
HRDTP = 2	LAST-YEAR = 1	USED = "U"
CNVTBL = 3	ANCIENT = 2	
STNWGN = 4		

and that no other values of the data-names are permissible, rewrite the following table using condition-names wherever possible:

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
BODY-TYPE EQ	1	1	1	3	4
AGE-YRS	LR 1	EQ 1	GR 1		
STATUS EQ "N"			Y	N	N

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5

Answer 11

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
BODY-TYPE	SEDAN	SEDAN	SEDAN	CNVTBL	STNWGN
AGE-YRS	THIS-YEAR	LAST-YEAR	ANCIENT		
STATUS			NEW	USED	USED

Note that no relation-operators should appear in the condition stub since all rows have been converted to condition-name rows.

c. End-File Tests

The third, and simplest, type of condition to be discussed here is the End-File Test. While this test is related more to a specific technique of processing data than to the inherent characteristics of the data itself, it is included here as an aid in practical applications of decision tables. The format of the test is:

EOF ON file-name

File-name is the name assigned to a file which contains input data to be processed in a serial fashion. Each time that a READ action is executed for the file to obtain the next information unit or record, an end-file test should be made before that information is examined or processed in any way, in case it should happen that the READ action failed to find any further information on the file.

In a limited entry condition-row the end-file test is written in the condition stub. The condition entry columns may contain Y, N, or blank, just as before.

Example:

	Rule 1	Rule 2
EOF ON ORDERS-RECEIVED	N	Y

End-file tests may not be written in extended entry form since only one file may be tested for an end-condition in a single condition-row.

Question 12

Remembering that rules should not be defined such that more than one of them can be satisfied simultaneously, why would it be improper to allow end-file tests in extended entry form as shown below?

	Rule 1	Rule 2	ELSE
EOF ON	TRANSACTIONS	MASTER	

Answer 12

If an end-file condition had been reached on both of the files TRANSACTIONS and MASTER, then Rule 1 and Rule 2 would be satisfied simultaneously.

Question 13

Write end-file tests in the following table (in limited entry form) such that Rule 1 is satisfied if no end-file conditions have been reached, Rule 2 is satisfied if an end-file condition has been reached on TRANSACTIONS but not on MASTER, and Rule 3 is satisfied if an end-file condition has been reached on MASTER but not on TRANSACTIONS.

	Rule 1	Rule 2	Rule 3	ELSE

Answer on next page.

Question 14

What happens in the preceding case if an end-file condition has been reached on both files?

Answer 13

	Rule 1	Rule 2	Rule 3	ELSE
EOF ON TRANSACTIONS	N	Y	N	
EOF ON MASTER	N	N	Y	

Now turn back to question 14.

Answer 14

The action in the ELSE column will be executed.

II. ACTIONS

In the following discussion three general types of Actions will be discussed. Each general type of Action includes a number of specific action operators, as shown below:

	Type	Action Operators
a.	Arithmetic and Data Manipulation	MOVE, SET, ADD, SUBTRACT
b.	Sequence Control	DO, GO TO
c.	Input-Output	READ, WRITE

The use of each type will be shown both for limited entry and extended entry action rows.

a. Arithmetic and Data Manipulation

MOVE

The format of the MOVE action statement is:

MOVE operand-1 TO data-name

The operand-1 which immediately follows the operator MOVE may be any one of the following:

Operand	Example
data-name	MOVE NEW-AMT TO POLICY-AMT
literal	MOVE 2 TO MARITAL-STATUS
formula	MOVE (AMT + EXCESS) * RATE TO CHARGES

In the first example the contents of the item NEW-AMT is transferred to the item POLICY-AMT; in the second example the value 2 is trans-

In a limited entry action row the whole MOVE action statement is written in the action stub. The action entry may contain:

Entry	Meaning
X	The action will be executed when the rule is satisfied;
(hyphen) or blank	The action will not be executed when the rule is satisfied.

Example:

Action Stub	Action Entries	
	Rule 1	Rule 2
MOVE NEW-AMT TO POLICY-AMT	X	X
MOVE "B" TO BONUS-CODE	X	—

In an extended entry action row the data-name that follows the word TO is written in the action entry, while the rest of the action statement is written in the action stub. As before, the action entry may contain — or blank to indicate that the action is not to be executed for a particular rule.

Example:

Action Stub	Action Entries		
	Rule 1	Rule 2	ELSE
POLICY-TYPE EQ	"A"	"B"	
MOVE NEW-AMT TO	POLICY-AMT	POLICY-AMT	ERROR-AMT
MOVE NEW-AMT * RATE TO	A-CHARGE	B-CHARGE	—

Question 15

At some point in the processing of factory orders possessing a JOB-CODE item which may take the values A, B, C, or D, the item PRICE must be placed in the item NET-A, NET-B, NET-C, or NET-D respectively, depending on JOB-CODE. If JOB-CODE is A or B, the PRICE multiplied by 1.1 must be placed in GROSS-A or GROSS-B respectively, while if the JOB-CODE is C or D, the PRICE multiplied by 1.2 must be placed in GROSS-C or GROSS-D respectively. In addition, if JOB-CODE is A or D then the value R71 should be placed in the item SHOP-ORDER; if B, then R82 should be placed there, and if C, then T34 should be placed there. Using the MOVE action, describe the above operations in the following table. Use extended entry rows wherever this will reduce the number of rows that must be written in the table. Do not omit quote marks from non-numeric literals.

Answer 15

	Rule 1	Rule 2	Rule 3	Rule 4
JOB-CODE EQ	"A"	"B"	"C"	"D"
MOVE PRICE TO	NET-A	NET-B	NET-C	NET-D
MOVE PRICE * 1.1 TO	GROSS-A	GROSS-B		
MOVE PRICE * 1.2 TO			GROSS-C	GROSS-D
MOVE "R71" TO SHOP-ORDER	X			X
MOVE "R82" TO SHOP-ORDER		X		
MOVE "T34" TO SHOP-ORDER			X	

SET

In the question just completed you will notice that in the action row MOVE PRICE TO ....., it was possible to write the various receiving items NET-A, NET-B, NET-C, and NET-D in the action entry columns for the corresponding rules, since PRICE was a common sending item. However, this was not possible when the various sending items "R71", "R82", and "T34" had to be moved to a common receiving item SHOP-ORDER, and thus several rows were required in the table to express the actions instead of only one row. To eliminate this restriction the SET action is introduced. It's format is:

SET data-name EQ operand-2

The operand-2 which follows the word EQ may be a data-name, literal, or formula, similar to operand-1 of the MOVE action. However, the action takes place from right to left in this case, instead of from left to right, i.e., SET causes the value of operand-2 to be placed in the data-name that follows the operator SET.

In a limited entry action row the whole SET statement is written in the action stub, while in an extended entry row operand-2, if it is a data-name or a literal, is written in the entry. When a formula is used as the operand-2 of a SET in an extended entry row the last data-name or literal that appears in the formula, and also the preceding arithmetic operator if desired, is written in the entry, and the remainder of the formula is written in the

Examples:

	Rule 1	Rule 2	Rule 3
SET POLICY-AMT EQ NEW-AMT	X	X	
SET PREMIUM EQ NEW-AMT * RATE	X	X	
SET AGENCY EQ	NEW-AGENCY	OLD-AGENCY	
SET BONUS-CODE EQ	"B"		"C"
SET AGENT-FEE EQ (NEW-AMT + BONUS)*	RATE-A	RATE-B	
SET BILLING-RATE EQ RATE	+ .02		- .01

Question 16

Show in the first action row of Table 2 below how the last three action rows of the example in question 15 can be expressed in a single action row; for convenience, that part of the example is repeated here as Table 1. Show in the second action row how to add 5, 10, 15 or 20 to the item SURPLUS, for JOB-CODE equal to A, B, C, or D respectively.

Table 1.

JOB-CODE EQ	Rule 1 "A"	Rule 2 "B"	Rule 3 "C"	Rule 4 "D"
MOVE "R71" TO SHOP-ORDER	X			X
MOVE "R82" TO SHOP-ORDER		X		
MOVE "T34" TO SHOP-ORDER			X	

Table 2.

JOB-CODE EQ	Rule 1 "A"	Rule 2 "B"	Rule 3 "C"	Rule 4 "D"

Answer 16

	Rule 1	Rule 2	Rule 3	Rule 4
JOB-CODE EQ	"A"	"B"	"C"	"D"
SET SHOP-ORDER EQ	"R71"	"R82"	"T34"	"R71"
SET SURPLUS EQ SURPLUS +	5	10	15	20

Question 17

The following table was established to accumulate counts of each type of job processed.

	Rule 1	Rule 2	Rule 3
JOB-CODE EQ	"A"	"B"	"C"
SET A-COUNT EQ A-COUNT + 1	X		
SET B-COUNT EQ B-COUNT + 1		X	
SET C-COUNT EQ C-COUNT + 1			X

Rewrite the first action, using the MOVE operator in a limited entry form.

MOVE	X		

Answer on page 35

Question 18

Is there a way of expressing all three action rows from question 17 as a single extended entry action row, using either a MOVE or SET operator as described in this manual? If not, invent your own action operator and format to solve the problem.

Answer on page 36

Answer 17

MOVE A-COUNT + 1 TO A-COUNT	X		

Now turn back to question 18.

Answer 18

No; when an arithmetic operand is the same item that is used for storage of the result of the operation, and this item is not common from one rule to another, MOVE and SET statements as described here will not accommodate expansion into extended entry form. We recommend the following:

ADD 1 TO	A-COUNT	B-COUNT	C-COUNT

ADD and SUBTRACT

The question just completed provided a demonstration of the utility of an ADD action. A corresponding SUBTRACT action is included here. Since it is relatively less likely that the result item of a multiply or divide operation would have to be stored in one of the operand items, it does not seem to be necessary to introduce MULTIPLY and DIVIDE actions. The options available through the use of formulae will normally be adequate.

The formats for ADD and SUBTRACT actions are:

```
ADD       operand-1   TO     data-name
SUBTRACT  operand-1   FROM   data-name
```

Operand-1 may be a literal or data-name.

In limited entry action rows the whole action statement is written in the action stub. In extended entry action rows the data-name following the word TO or FROM is written in the entry.

Example:

ADD TAX TO NET	X		X
SUBTRACT AMOUNT FROM	TOTAL-A	TOTAL-B	

b. Sequence Control

GO TO

A decision table specifies a set of rules each of which consists of a set of conditions to be tested, together with a corresponding sequence of actions. The action sequence corresponding to a particular rule is executed if the rule is satisfied. The whole process of evaluating the conditions and executing the appropriate actions is initiated when sequence control is passed to the table. After completion of the appropriate actions it will normally be necessary for processing to continue under control of another table. Accordingly the last action associated with a rule may be to pass sequence control to another table. This is achieved with a GO action statement, which has the format:

GO TO table-name

Every table should possess a name to enable references to be made to it by GO actions. The name is written above the top left-hand corner of the table.

It is not possible to pass control to a portion or subdivision of a table. If this type of action is required the table must be rewritten as two or more tables each with its own name.

If the last action statement associated with a rule is not a GO TO statement, then sequence control passes to the next table.

In a limited entry action row the whole statement is written in the stub.

In an extended entry action row the table-name which appears in the GO statement may be written in the entry.

Example:

The name of the following table is CODE-TEST.

CODE-TEST	Rule 1	Rule 2	Rule 3	ELSE
JOB-CODE EQ	"A"	"B"	"C"	
ADD 1 TO	A-COUNT	B-COUNT	C-COUNT	ERROR-COUNT
GO TO	TAB-A	TAB-B	TAB-X	–
GO TO ERROR	–	–	–	X

The functions in the second and third action rows are separated purely for illustrative purposes.

Question 19

What is wrong with the following table?

FIRST-TAB	Rule 1	Rule 2	Rule 3	Rule 4	ELSE
MARITAL-STATUS	SINGLE	SINGLE	MARRIED	MARRIED	
MALE	Y	N	Y	N	
GO TO	TAB-S2	–	TAB-M2	–	TAB-G
GO TO TAB-F	–	X	–	X	X

Answer:

Answer 19

The ELSE column specifies two tables to which sequence control is to be transferred.

FIRST-TAB	Rule 1	Rule 2	Rule 3	Rule 4	ELSE
MARITAL-STATUS	SINGLE	SINGLE	MARRIED	MARRIED	
MALE	Y	N	Y	N	
GO TO	TAB-S2	TAB-F	TAB-M2	TAB-F	TAB-G

Answer questions 20 and 21 before turning the page.

Question 20

If no ELSE column had been written for the above table, and MARITAL-STATUS was neither SINGLE nor MARRIED, what would happen to the sequence control?

Answer:

Question 21

If the entry TAB-F were omitted from Rule 2 in the above table, how else could you prevent sequence control from being undefined when the conditions of Rule 2 were satisfied?

Answer:

Answer 20

Sequence control would be undefined under these circumstances. This is a serious error in the preparation of the table.

Answer 21

Write EXIT: TAB-F (for example) beneath the table name FIRST-TAB, then sequence control would still pass to TAB-F when Rule 2 was satisfied.

DO

The DO action statement has the format

DO table-name

Like the GO TO action, it causes sequence control to be passed to another table, but, in addition, it causes control to be returned to the normal sequence of actions in the original table after the actions associated with any rule of the other table have been executed. Notice that the sets of actions in this other table should not terminate with GO TO actions since they would then contradict the return of control that is indicated by the DO action.

Again, like to GO TO action, the table-name may appear in the entry of an extended entry row.

Example:

CODE-TEST	Rule 1	Rule 2	Rule 3	ELSE
JOB-CODE EQ	"A"	"B"	"C"	
ADD 1 TO	A-COUNT	B-COUNT	C-COUNT	–
DO AB-PROCESS	X	X	–	–
DO AC-PROCESS	X	–	X	–
DO	A-EDIT	B-EDIT	C-EDIT	ERRORS
GO TO NEXT-JOB	X	X	X	X

Answer 22

Go to table AC-PROCESS

Return to Action 4 of table CODE-TEST

Go to table C-EDIT

Return to action 5 of table CODE-TEST

Go to table NEXT-JOB

Question 23

Could the 2nd and 3rd action rows be combined into a single extended entry action row? Write your solution or reasoning here.

Answer 23

No. In Rule 1 the actions specified in the 2nd and 3rd rows must both be executed. This could not be expressed in a single action row.

c. Input-Output

WRITE

The format of a WRITE action is:

WRITE file-name

File-name is the name of some file which has been designated to receive output data from the processing operation. The action causes the current unit of information, or record, that is associated with the file to be placed on the file. The record is then no longer accessible and cannot be processed by condition tests or action statements. However, it may be possible to retrieve the record for further processing by means of the READ action, if the file is of a type which permits direct Read-back.

READ

The format of a READ action is:

READ file-name

File-name is the name of some file which contains input data to be processed. The action causes a unit of information, or record, to be brought from the file to an access area where it can be processed by condition tests or action statements. After a READ action, the information brought in by the previous READ is not accessible.

MAIN-TABLE	Rule 1	Rule 2	ELSE
JOB-CODE EQ	"A"	"B"	
QTY-ORDERED LE 1000	Y	Y	
DO	A-EDIT	B-EDIT	ERRORS
WRITE	EDITED	EDITED	REJECTS
READ INPUT-JOBS	X	X	X
GO TO MAIN-TABLE	X	X	X

Question 24

In the above example, what kinds of actions would you expect to find in table A-EDIT, bearing in mind that the data access areas associated with the two files EDITED and INPUT-JOBS are separate and distinct?

Answer:

Answer 24

The expected actions would include MOVE and/or SET statements to transfer data from the access area associated with the INPUT-JOBS file to the access area associated with the EDITED file. These statements could involve some arithmetic computations also.

REFERENCE LIST OF BASIC FUNCTIONS

FOR DECISION TABLES

I. CONDITIONS

a. Relation Tests

	Rule n
operand-1 operator operand-2	{ Y / N / blank or – }
operand-1 operator	{ operand-2 / blank or – }
operand-1	{ operator operand-2 / blank or – }

Operands	Operators	Meaning
Data-names	GR	GReater than
Literals	LR	LesseR than
Formulas	EQ	EQual to
	LE	Less than or Equal to
	GE	Greater than or Equal to
	NE	Not Equal to

b. Condition-Names

	Rule n
condition-name	{ Y / N / blank or – }
associated data-name	{ condition-name / blank or – }

II. ACTIONS

a. Arithmetic and Data Manipulation

Must appear in Stub	May appear in Stub or Entry
MOVE { data-name / literal / formula } TO	data-name
SET data-name EQ { [formula (except last operand)] }	{ data-name / literal / [formula (last operand)] }
ADD { literal / data-name } TO	data-name
SUBTRACT { literal / data-name } FROM	data-name

b. Sequence Control

Must appear in Stub	May appear in Stub or Entry
GO TO	table-name
DO	table-name

c. Input-Output

Must appear in Stub	May appear in Stub or Entry
READ	file-name
WRITE	file-name