# CHAPTER TWO

# PROVING THE CORRECTNESS OF FLOWCHART PROGRAMS

## 2 1 INTRODUCTION

When we write a computer program, our intention is that it carry out some particular computation However, as all programmers are painfully aware, most of our programs have errors (or bugs) in them Thus we normally spend a large percentage of our programming time testing and debugging the (incorrect) programs we have written Even when we have finished testing and debugging a program, we cannot be certain that it is completely correct All we can be sure of is that he program gives the correct results for the particular data we used to test it Later, when the program is used with different data, still another bug may appear Every experienced programmer knows of programs that had run correctly over a long period of time and thus appeared almost certainly to be correct, when suddenly another bug mysteriously appeared in the program Hence testing, no matter how extensive, can never really assure us that a program is correct

Ideally, we should prove that the program is correct rather than depending solely on testing it Of course, even if we manage to give a proof that the program is correct, we cannot be absolutely certain that it is For proofs can have errors (bugs) in them just as programs do Nonetheless, the process of attempting to prove that a program is correct is very helpful in forcing one to thoroughly comprehend the program One can only prove a program correct by understanding it very thoroughly Experience has shown that for this reason, correctness proofs are helpful in uncovering bugs one might otherwise overlook

If we could formalize correctness proofs and have them checked by an absolutely reliable source (an automatic proof checker), then we could have complete confidence in them This may be possible in the future (see Chapter 5) but is impractical at the present time We will be dealing with informally expressed correctness proofs Experience with such informal proofs indicates that they do increase ones confidence in and understanding of programs and are thus worthwhile, even though they do not bring absolute certainty Such informal crrectness proofs can be viewed as a very systematic form of desk checking a program -- something all good programmers do anyway This chapter introduces the basic ideas of such correctness proofs for flowchart programs

## 2 2 BASIC PRINCIPLES OF PROVING FLOWCHART PROGRAMS CORRECT

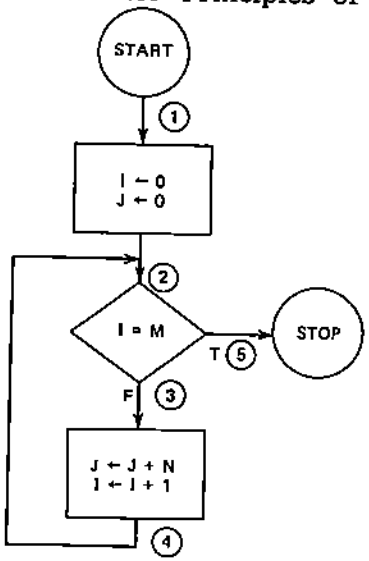
If we wish to prove that some program is correct, of course we must be able to write down a description of what the program is supposed to accomplish Let us refer to such a description as the correctness statement or assertion A proof that the program is correct then consists of a proof that the program, when executed, will eventually terminate (stop) and, when it does terminate, the correctness assertion will be true Often a program is only required to work correctly for certain values of the input data If this is the case, the correctness proof would consist of a proof that, whenever the program is executed with appropriate input data, it will eventually terminate and, when it does terminate, the correctness assertion will be true

Let us illustrate these ideas with the correctness proof for a very simple flowchart program

### EXAMPLE 2 2 1

Suppose we wish to compute the product of any two integers M,N such that $M \geq 0$ without using the operation of multiplication One way to do this is to add N to itself M times The result will be $M \cdot N$ Consider the following flowchart program, which implements this computation






We wish to prove that this program correctly computes the product of any two integers M and N provided that $M \geq 0$, that is, whenever the program is executed with M and N already having integer values and $M \geq 0$, it will eventually terminate (reach point 5) with J = M N

In order to check that the flowchart works correctly, we could test it with some specific data for example, let us do a hand simulation of the execution of the program with the data M = 3 and N = 5

The following table shows the values of the variables I and J each time execution reaches point 2 (the loop point) in the flowchart

For M = 3, N = 5

| The Number of Times, n, That Execution Has Reached Point 2 | Value of I | Value of J |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 1 | 5 |
| 3 | 2 | 10 |
| 4 | 3 | 15 |

The fourth time execution reaches point 2, the value of I is 3, which is equal to M, and therefore execution will leave the loop and proceed to point 5

When execution reaches point 5 , J = 15 = 3·5 = M N Thus we see that the flowchart works correctly for the specific data, M = 3, N = 5 Even though this tends to convince us that the program works correctly, it certainly does not prove that the program works correctly for all possible values of M and N We can now continue to test the program with different data for M and N If it ever computes a wrong answer, we would of course know that it is incorrect However, if it continues to compute correct answers, we may gain confidence in the program, but we will never be certain that it computes correct answers for all possible values of M and N (there will always be an infinite number of possible values for which the program has not been tested) Of course, on any real computer, the values of M and N are restricted to some finite range of integer numbers Thus in principle the program could be exhaustively tested on this range of numbers However for most computers this range is so large as to make exhaustive testing completely impractical Furthermore, we normally design such programs to work correctly, regardless of the size of M and N Hence we should only be satisfied if we can demonstrate that the program works correctly regardless of the size of M and N This we can never accomplish by testing

| The Number of Times, n, That Execution Has Reached Point 2 | Value of I | Value of J |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 1 | N |
| 3 | 2 | 2·N |
| 4 | 3 | 3·N |
| . | . | |
| . | | . |
| | | . |
| M+1 | M | M N |



Suppose that instead of executing the program with specific data for M and N, we execute it with the symbolic data M, N Then we would obtain the previous trace of the values of I, J each time execution reaches point 2

Thus we see that when execution has proceeded around the loop M times (so that execution has reached point 2 for the M+1th time), the value of I is M and the value of J is M N At this point I's value is equal to M, and hence execution will leave the loop and proceed to point 5 Thus, when execution reaches point 5 and terminates, we will have J = M·N This shows that the program works correctly for any values of M and N Nevertheless, notice that the trace assumes that M ≧ 0 so that I's value (which starts at 0 and increases by 1 each time around the loop) will eventually equal M If M < 0, then execution would proceed around the loop forever without I's value ever being equal to M, and hence execution would fail to terminate Thus the flowchart really works correctly only for M ≧ 0 For any integer values of M, N such that M ≧ 0, the above trace shows that the flowchart eventually terminates with J = M N

However, notice that there is a gap in the table represented by three dots How did we figure out what I and J's values were after the gap in the table? And how can we be sure that they are as shown? The answer to the first question is that by tracing the flowchart and filling in the first part of the table, we came to understand how I and J's values change each time around the loop (we saw a pattern to these changes) and used this understanding (inductively) to figure out what the values of I and J would be the M+1th time execution reaches point 2 The answer to the second question is that to be really sure we should prove it

We can see from the table that each time execution reaches point 2, we have J = I·N Let us prove this by induction on the number of times n that execution has reached point 2 (see the last paragraph of Section 1 2 for a thorough explanation of such inductive proofs)

(1) The first time (n=1) execution reaches point 2 (coming from point 1), we have I = 0 and J = 0 and thus J = I N = 0 N = 0 is true

(ii) Suppose that J = I N is true the nth time execution reaches point 2 We need to show that if execution returns to point 2 for an n+1th time, then J = I N will again be true Let us denote the values of I and J the nth time execution reaches point 2 by $I_n$, $J_n$ Thus the induction hypothesis would be that $J_n = I_n \cdot N$ The only way that execution can return to point 2 for an n+1th time is to proceed around the loop (to points 3, 4 and back to point 2) If it does so, then we can see by tracing execution around the loop once that when execution returns to point 2, I's new value is its old value +1 and J's new value is its old value +N That is,

$$I_{n+1} = I_n + 1 \text{ and } J_{n+1} = J_n + N$$

$$J_{n+1} = I_n \quad N + N \text{ by the induction hypothesis}$$

$$= (I_n + 1) \cdot N$$

$$= I_{n+1} \; N$$

Thus we see that if execution returns to point 2 for a n+1th time, again, J = I N This completes the induction proof and shows that each time execution reaches point 2, the statement J = I·N is true

The only way for execution to terminate (stop going around the loop) is to reach point 2 with the test I = M true At that point execution would be at point 2 with I = M and J = I N = M·N Execution would then proceed to point 5 and terminate with J = M N However, please note that we have not yet shown that execution does in fact terminate All we have shown is that each time execution reaches point 2, J = I·N and, thus, if execution does ever terminate, J = M·N To see this more clearly, note that the proof that J = I N each time execution reaches point 2 did not depend on whether M ≥ 0 or not Hence, even if M < 0, that proof would still be correct and would show that each time execution reaches point 2, J = I·N However, if M < 0, execution will proceed around the loop forever without I's value ever being equal to M and thus execution will not terminate and will not correctly compute J = M·N It will be true, however, even in this case (M < 0), that each time execution reaches point 2, J = I·N (this must be the case or the previous proof was incorrect) Check this carefully for yourself by tracing execution of the flowchart with M having some value < 0 You will see that for such an M, execution never terminates, but it is still true that each time execution reaches point 2, J=I·N



Thus we still need to prove that the flowchart terminates To do this we need the assumption about the data, namely that M ≥ 0 We want to prove that if M ≥ 0, then eventually execution will reach point 2 with I = M This is quite apparent from the flowchart, since I's value is initialized to 0 and is increased by 1 each time around the loop Thus it must eventually increase to be equal to M, provided that M is an integer and M ≥ 0 However, if we want to be very rigorous about this, we can also prove this fact by induction

If M ≥ 0, let us prove by upward induction on m that for each value of m, such that 0 ≤ m ≤ M, execution eventually reaches point 2 with I = m

(i) The first time execution reaches point 2 we have I = 0 Thus the statement is true for m = 0 If M = 0, then this proves the result

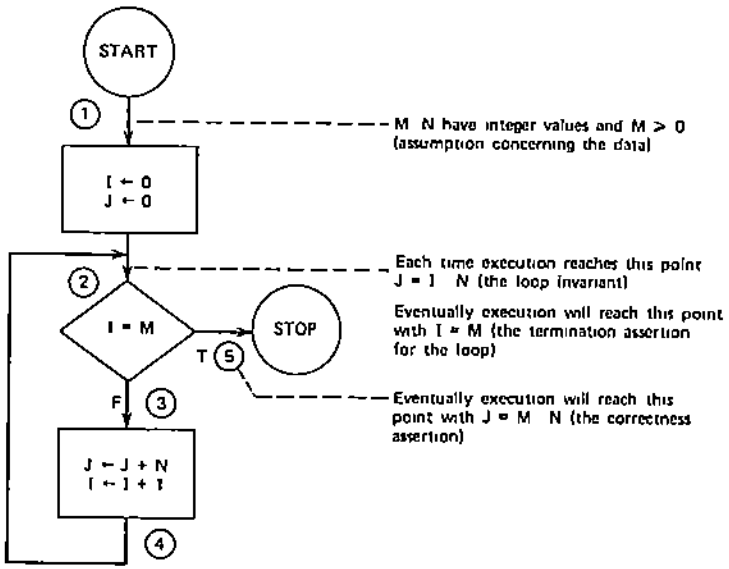
(ii) Otherwise suppose that execution does eventually reach point 2 with I = m and 0 ≤ m < M We need to show that execution will also reach point 2 with I = m + 1 When execution is at point 2 with I = m and 0 ≤ m < M, we will have that the test I = M is false, since m < M and hence execution will proceed around the loop and return to point 2 When it returns to point 2, I's value will have been increased by 1, and hence we see that execution will eventually reach point 2 with I = m+1 This completes the proof by upward induction

From the proceeding, we may conclude that if M ≥ 0, then execution will eventually reach point 2 with I = M At that point, the test I = M will be true and execution will proceed to point 5 and terminate

We have now proven in (excessive) detail that if this flowchart is executed with M and N being any two integer numbers and M ≥ 0, then execution will eventually terminate and when it does, we will have J = M N

## EXAMPLE 2 2 2

We have given the preceeding proof in more detail than is really necessary Let us now repeat this proof leaving out some of the detail and formality of the induction proofs In particular, in proving that a statement is true each time execution reaches some point inside a simple loop like the one in the previous flowchart, all we need to show is that (i) the statement is true the first time execution reaches that point, and (ii) if execution is at the point and the statement is true (the induction hypothesis), and execution proceeds around the loop and returns to the point, then the statement will again be true If we can prove (i) and (ii) then, by filling in the details, we can prove (by induction on the number of times that execution has reached the point) that each time execution reaches the point the statement is true In giving the second version of this correctness proof, we also attach the key assertions that we wish to prove directly to the points in the flowchart to which they refer This helps us to keep clearly in mind the key steps of the correctness proof






Now let us prove that the above flowchart is correct, that is, if it is executed with M, N having integer values with $M \geq 0$, then eventually execution will terminate with $J = M \cdot N$

First let us prove that each time execution reaches point 2, J = I N

(i) The first time execution reaches point 2 (by proceeding from point 1 to point 2), I = 0 and J = 0 Thus it will be true that J = I N = 0 N = 0

(ii) Suppose execution is at point 2 and $J = I \cdot N$ is true Let us call I and J's values at this point $I_n$ and $J_n$ so that we have $J_n = I_n N$ Now suppose execution proceeds around the loop (from point 2 to 3, 4 and back to 2) When execution returns to point 2, the new values of I and J, which we denote as $I_{n+1}$ and $J_{n+1}$, will be

$$I_{n+1} = I_n + 1$$

$$J_{n+1} = I_n \cdot N + N \text{ (since } J_n = I_n \cdot N)$$

$$= (I_n + 1) \cdot N$$

$$= I_{n+1} \cdot N$$

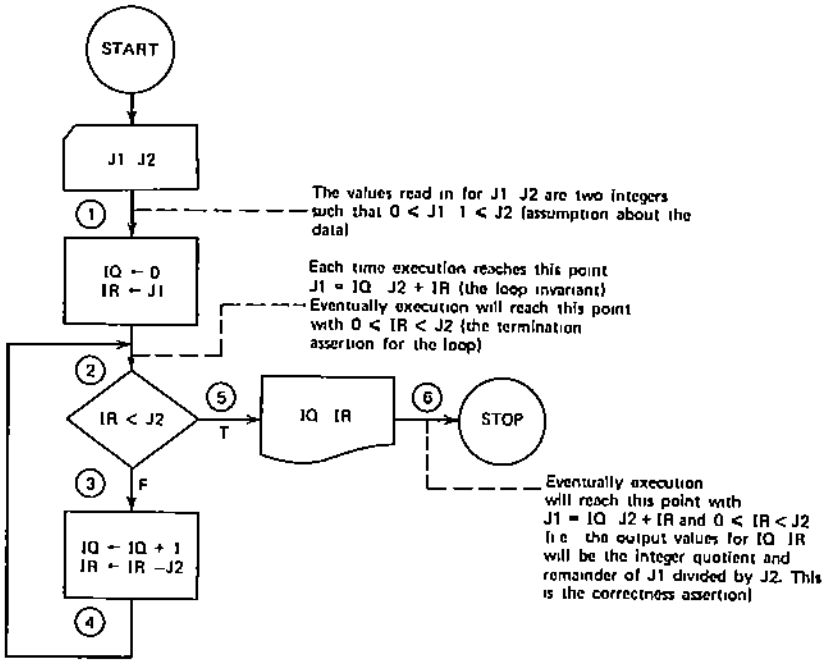
Thus, upon returning to point 2, we again have that $J = I \cdot N$ is true This completes the proof that each time execution reaches point 2, J = I N

Next let us prove that eventually execution will reach point 2 with I = M The first time execution reaches point 2, I = 0 Each succeeding time (if any) that execution reaches point 2, I's value is increased by 1 Since M's value is never changed by the flowchart and since we are assuming that $M \geq 0$, it is apparent that I's value will eventually increase to equal M

When execution reaches point 2 with I = M, we will also have $J = I \cdot N = M \cdot N$ At that point the test I = M will be true, and execution will follow the true arrow to point 5 Thus execution will eventually reach point 5 with $J = M \cdot N$ This completes the correctness proof

## EXAMPLE 2 2 3

The next program we wish to prove correct is one for computing the integer quotient, IQ, and remainder, IR, of J1 divided by J2, where J1 and J2 are any two integers with $0 \leq J1$ and $1 \leq J2$ The flowchart computes IQ and IR without using division One way to express what the flowchart is supposed to compute is to say that it computes two integer numbers IQ, IR so that $J1/J2 = IQ + IR/J2$ and $0 \leq IR < J2$ Rewriting this, $J1 = IQ \cdot J2 + IR$ and $0 \leq IR < J2$ As in the previous example, we attach directly to the flowchart any assumptions about the data and the facts that we wish to prove about the flowchart




Note that we have attached the assumption about the values that are read in for J1 and J2 to the point in the flowchart immediately below the input box This is to indicate that we are only trying to prove that the program works correctly for input data that satisfy these assumptions We have attached the assertion that expresses the correctness of the flowchart to the terminal point of the flowchart In addition, we have attached two key facts that we need to prove about the loop to the beginning point of the loop Now let us do a trace of the flowchart execution with symbolic data for J1 and J2 and see how we arrived at the key loop assertion (the loop invariant)



| The Number of Times, n, Execution Has Reached Point 2 | IQ | IR |
|---|---|---|
| 1 | 0 | J1 |
| 2 | 1 | J1 - J2 |
| 3 | 2 | J1 - 2·J2 |
| 4 | 3 | J1 - 3 J2 |
| . | | |
| | | Until IR<J2 |
| | . | |

Note that each time execution reaches point 2, we have

$IR = J1 - IQ \quad J2$

or, rewriting this,

$J1 = IQ \quad J2 + IR$

This is identical to the correctness assertion except that it is also necessary that $0 \leq IR < J2$ Thus, we also need to show that the loop eventually terminates and when it does, $0 \leq IR < J2$ The student can see that the statements attached to point 2 in the flowchart essentially express these facts about the loop

In order to prove the flowchart is correct, let us first prove that each time execution reaches point 2 in the flowchart, $J1 = IQ \cdot J2 + IR$

(1) The first time execution reaches point 2 (by proceeding from START to 2), we have IQ = 0 and IR = J1
Thus the assertion

$$J1 = IQ \cdot J2 + IR$$
$$= 0 \cdot J2 + J1$$
$$= 0 + J1$$
$$= J1$$

will be true

(ii) Suppose execution is at point 2 and J1 = IQ J2 + IR is true Let us call IQ and IR's values at this point $IQ_n$ and $IR_n$ We assume the assertion is true at this point, that is, J1 = $IQ_n$ J2 + $IR_n$ (the induction hypothesis) Now suppose that execution proceeds around the loop (from point 2 to 3, 4 and back to 2) When execution returns to point 2, IQ and IR's new values are $IQ_{n+1} = IQ_n+1$ and $IR_{n+1} = IR_n-J2$ Thus we will have

$$IQ_{n+1} \cdot J2 + IR_{n+1} = (IQ_n + 1) \cdot J2 + (IR_n - J2)$$
$$= IQ_n \cdot J2 + J2 + IR_n - J2$$
$$= IQ_n \cdot J2 + IR_n$$
$$= J1 \qquad \text{(by the induction hypothesis)}$$

This completes the proof that each time execution reaches point 2, J1 = IQ · J2 + IR

Next let us prove that eventually execution will reach point 2 with $0 \leq IR < J2$ To do this, let us first prove that each time execution reaches point 2, $0 \leq IR$

(i) The first time execution reaches point 2 we have IR = J1 and, by assumption, $0 \leq J1$ (Note that this is the only place where this assumption is used in the proof Thus, if we



didn't require that $IR \geq 0$, the flowchart would work correctly, even for negative values of J1)

(ii) Suppose execution is at point 2 and $0 \leq IR$ Let us call its value at this point $IR_n$ If execution proceeds from point 2 around the loop, when execution returns to point 2, IR's new value is $IR_{n+1} = IR_n - J2$ But execution will proceed around the loop only if the test $IR_n < J2$ is false, that is, only if $J2 \leq IR_n$ But if $J2 \leq IR_n$, then $0 \leq IR_n - J2 = IR_{n+1}$ is true This completes the proof that each time execution reaches point 2 $0 \leq IR$

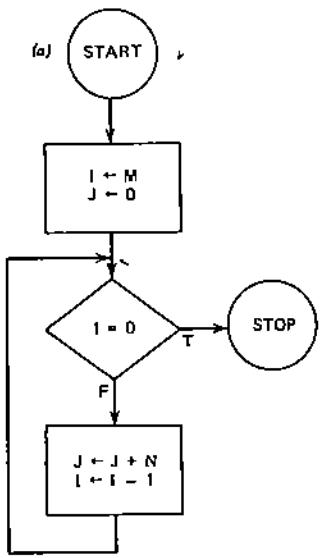
Next let us prove that eventually execution reaches point 2 with IR < J2 (so that, at that point, we would have $0 \leq IR < J2$) Note that each time around the loop, IR's value is decreased by subtracting J2's value from it Since J2's value is never changed by the flowchart and since, by assumption, $1 \leq J2$, we see that this decreases IR's value by at least 1 each time around the loop Hence IR's value must eventually decrease to be less than J2 (as usual, this informal proof could be made rigorous by induction -- see Exercise 7)

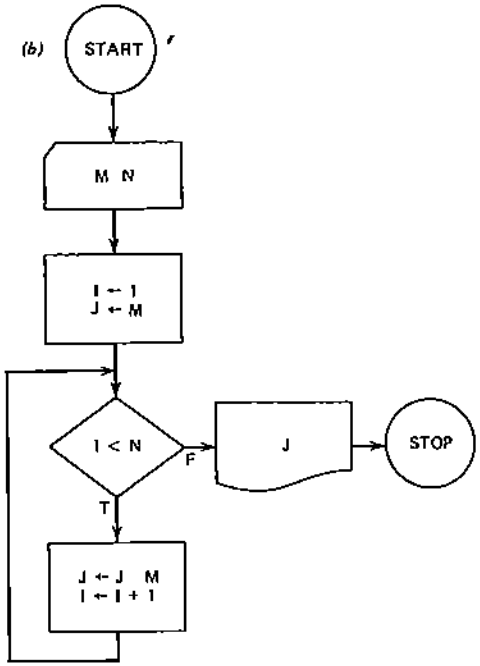
Thus eventually execution will reach point 2 with $0 \leq IR < J2$ and J1 = IQ J2 + IR At that point the test IR < J2 will be true and execution will pass from point 2 to point 5 and then to 6 Since none of the variables have their values changed between point 2 and point 6, it is obvious that the correctness assertion will be true when execution reaches point 6 Thus we have proven that any time this flowchart is executed with the values read in for J1 and J2 being two integers such that $0 \leq J1$ and $1 \leq J2$, execution will eventually terminate with J1 = IQ · J2 + IR and $0 \leq IR < J2$, that is, with IQ and IR being the integer quotient and remainder of J1 divided by J2

## EXERCISES

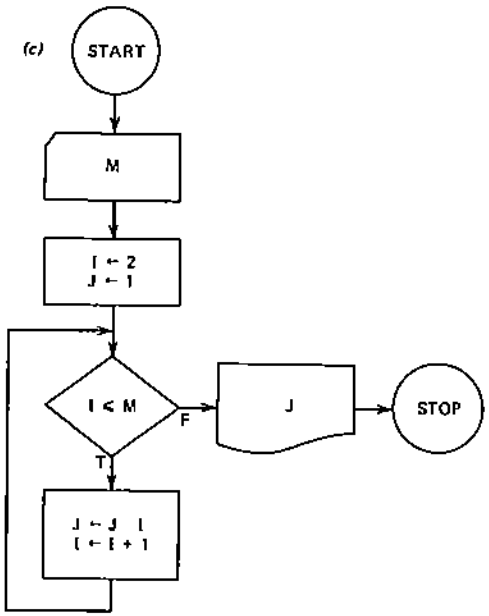
1 Prove that flowchart a) computes the product of M and N, provided that M and N already have integer values and $M \geq 0$ What happens if $M < 0$?

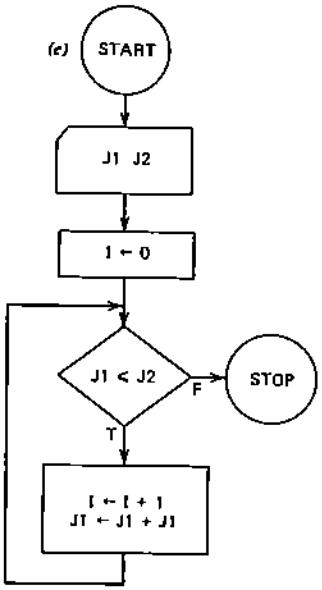
2 Suppose the test in flowchart a) had mistakenly been I = 1 rather than I = 0 Which steps(s) of the correctness proof would fail? Suppose J had mistakenly been initialized to 1 rather than 0 Which step(s) of the correctness proof would fail? Suppose J was initialized to N rather than 0 and the flowchart test was changed to I = 1 For what values of the data would the flowchart correctly compute M · N? How would this show up in the correctness proof?

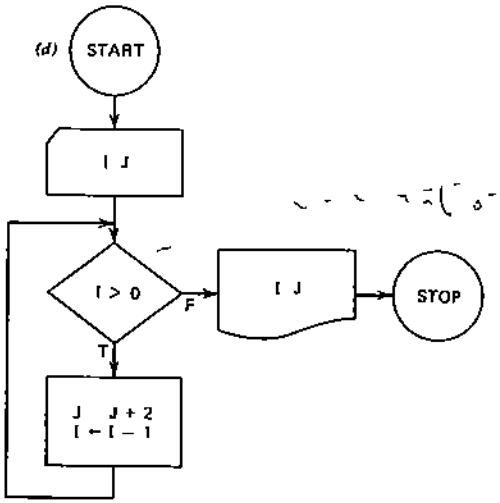





3 Prove that flowchart b) computes and prints out the value of $M^N$ provided the values read in for M and

N are integers such that $1 \leq M$ and $1 \leq N$ Does this flowchart work correctly if the value inputted for N is 0? Change the flowchart so that it does work correctly for N equal 0

4 Prove that flowchart c) computes and prints out the value of $M! = 1 \cdot 2 \cdot 3 \cdots (M - 1) \cdot M$, provided that the value read in for M is a positive integer

5 Prove that if the values read in for I, J in flowchart d) are $I_0$, $J_0$, respectively, and $0 \leq I_0$, then the values printed out for I and J will be 0 and $J_0 + 2 \cdot I_0$, respectively

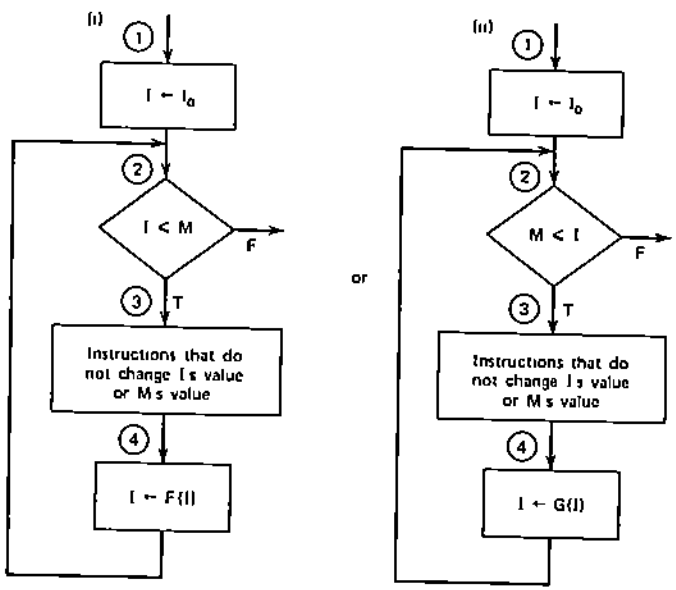
6 For flowchart e) prove the following If the values read in for J1, J2 are integers $J1_0$, $J2_0$ such that $1 \leq J1_0$, then execution of the flowchart will eventually terminate and when it does I will be the smallest nonnegative integer such that $J2_0 < J1_0 \cdot 2^I$ Hint -- Prove that each time execution reaches the loop point, $J1 = J1_0 \cdot 2^I$ and, for all integers i such that $0 \leq i < I$, $J1_0 \cdot 2^i \leq J2_0$

7 (a) Show that for any flowchart of the basic form (i), if for all I, $I + 1 \leq F(I)$, then the flowchart loop eventually terminates Prove this rigorously by induction on the number of times that execution has reached point 2 In particular, show that if execution has reached point 2 for the nth time, then $I \geq I_0 + n - 1$ Thus we will know for $n_0 = M - I_0 + 2$ (or $n_0 = 1$ if $M < I_0$) that after execution reaches point 2, at most $n_0$ times, $I \geq I_0 + (M - I_0 + 2) - 1 = M + 1$ And hence, eventually (after at most $n_0$ times), execution will reach point 2 with the test $I \leq M$ being false and will exit from the loop



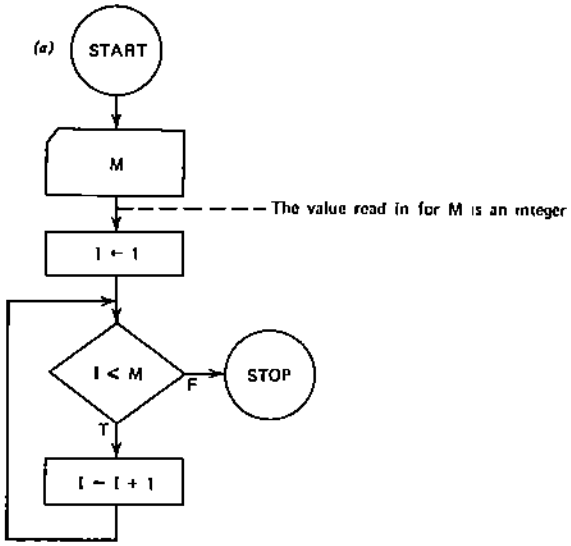
(b) Show that for any flowchart of the basic form (ii), if for all I, $G(I) \leq I - 1$, then the flowchart eventually terminates Prove this rigorously by induction

In most of the correctness proofs we will give rather informal proofs of the termination of loops This problem is designed to show that these could be made more rigorous



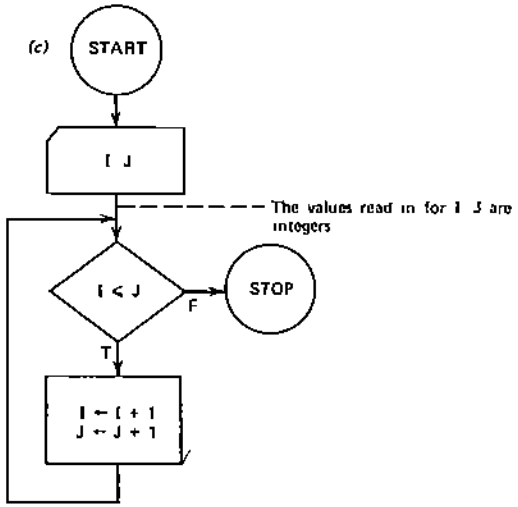

8 For each of the following flowcharts determine for what values of the data the flowchart terminates

(a)




(b) Same as (a) except the test I < M is replaced by the test I = M

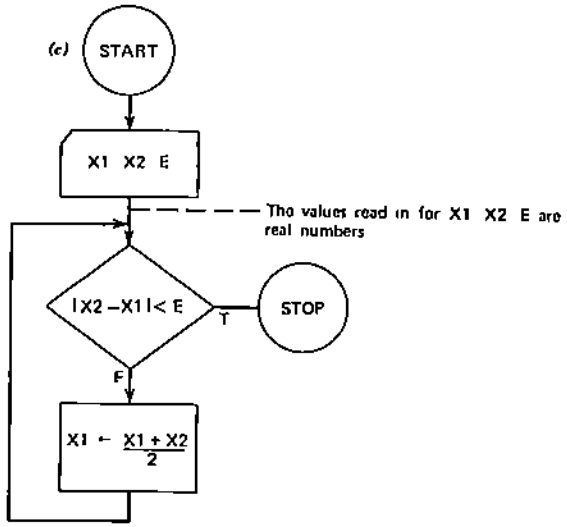
(c)



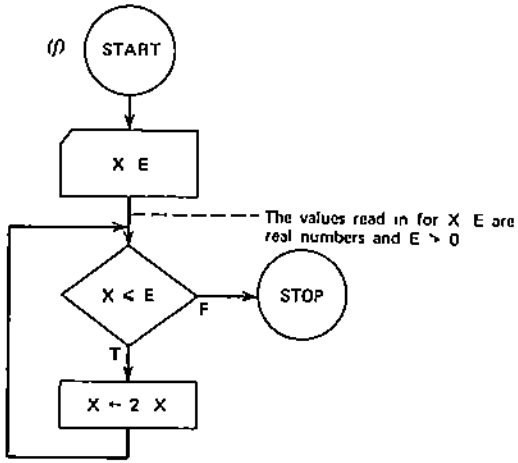

(d) Same as (c), except the instruction I ← I + 1 is replaced by I ← I + 2

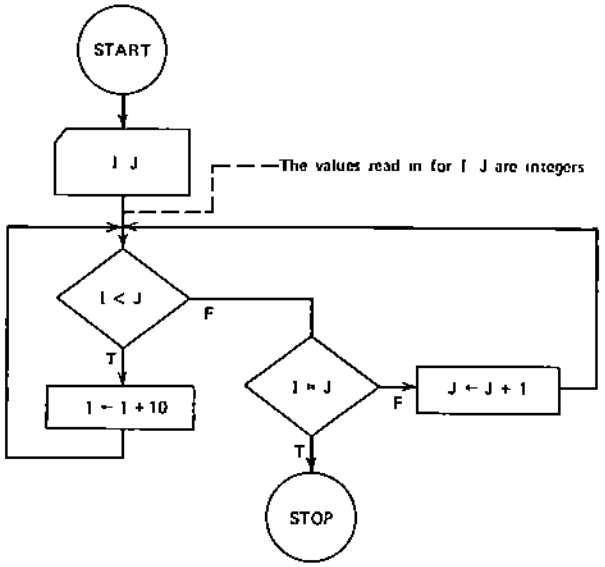


(e)




(f)






(h) Same as (g), except that the instruction J ← J + 1 is replaced by J ← J + 2

9 For flowcharts 8(e) and (f), give formal induction proofs of termination For example, 8(f) terminates for any value of X>0 Prove this by induction on the number of times n that execution has reached the loop point In particular, show that if the value read in for X is $X_0$>0, then the nth time execution reaches the loop point, $X = 2^{n-1} X_0$

## 2 3 ADDITIONAL EXAMPLES OF CORRECTNESS PROOFS FOR FLOWCHARTS

In this section we will present some additional examples of correctness proofs for flowchart programs The proofs will be presented in an informal style with some of the detail of the induction proofs deleted You should discover for yourself the key assertions attached to the loops in the flowcharts by tracing the execution of the flowchart with symbolic data and keeping track of what is true each time execution reaches the loop point