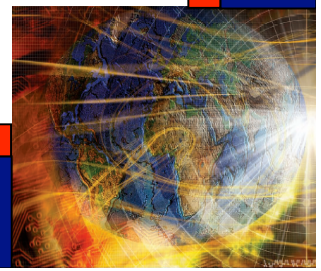
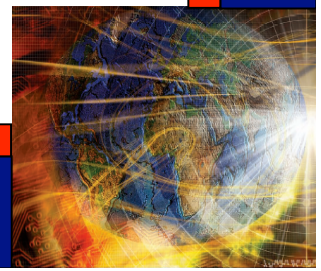
Chapter 9

Path Testing–Part 2



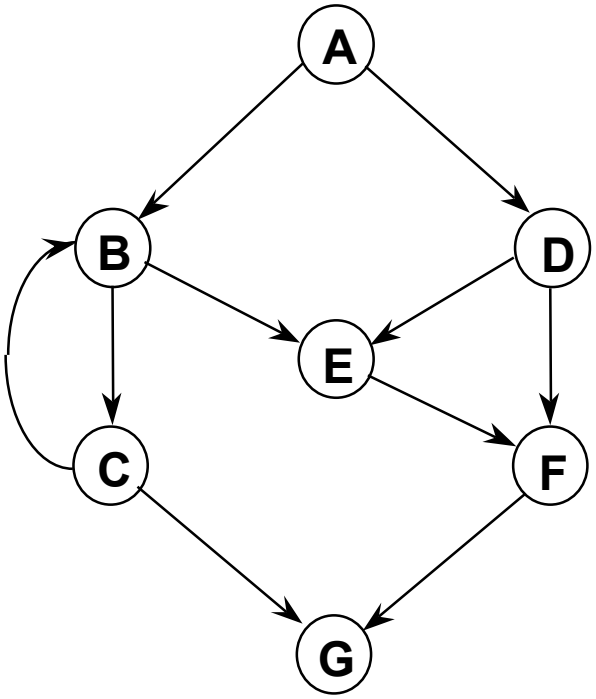
(McCabe) Basis Path Testing

- in math, a basis "spans" an entire space, such that everything in the space can be derived from the basis elements.
- the cyclomatic number of a strongly connected directed graph is the number of linearly independent cycles.
- given a program graph, we can always add an edge from the sink node to the source node to create a strongly connected graph. (assuming single entry, single exit)
- computing V(G) = e - n + p from the modified program graph yields the number of independent paths that must be tested.
- since all other program execution paths are linear combinations of the basis path, it is necessary to test the basis paths. (Some say this is sufficient; but that is problematic.)
- the next few slides follow McCabe's original example.



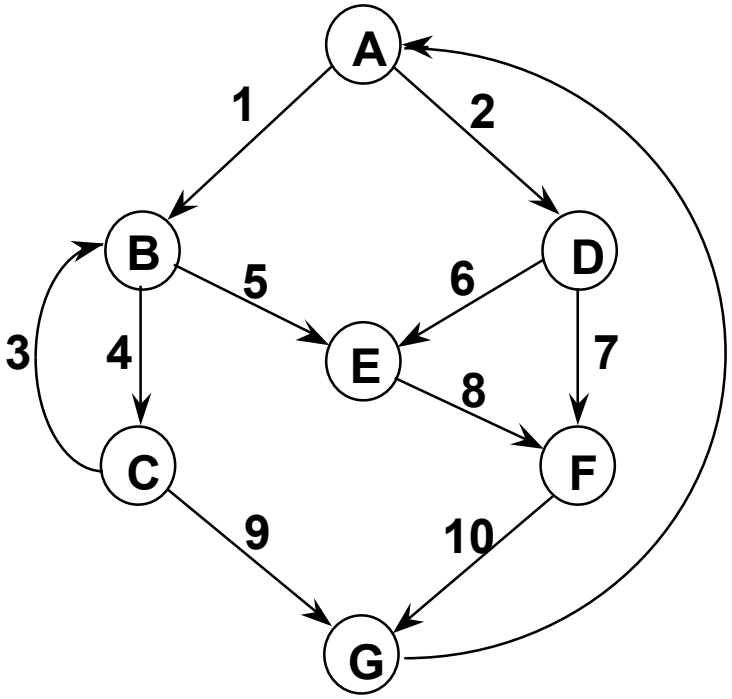
McCabe's Example

McCabe's Original Graph



V(G) = 10 - 7 + 2(1)
= 5

Derived, Strongly Connected Graph



V(G) = 11 - 7 + 1
= 5

McCabe's Baseline Method

To determine a set of basis paths,

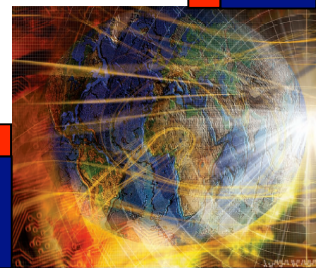
1. Pick a "baseline" path that corresponds to normal execution. (The baseline should have as many decisions as possible.)

2. To get succeeding basis paths, retrace the baseline until you reach a decision node. "Flip" the decision (take another alternative) and continue as much of the baseline as possible.

3. Repeat this until all decisions have been flipped. When you reach V(G) basis paths, you're done.

4. If there aren't enough decisions in the first baseline path, find a second baseline and repeat steps 2 and 3.

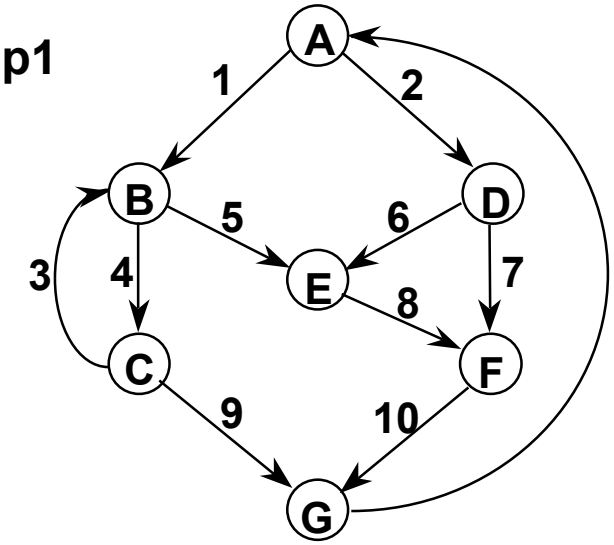
Following this algorithm, we get basis paths for McCabe's example.



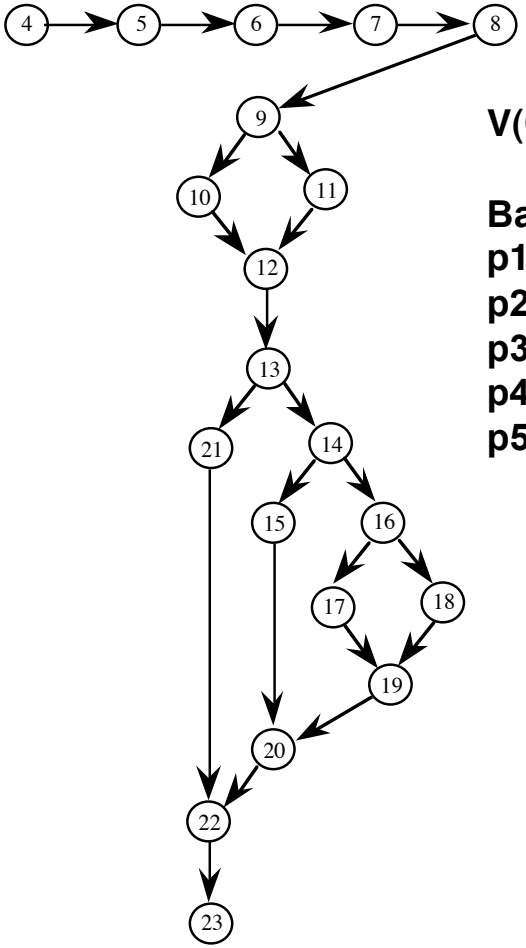
Basis Paths

path \ edges traversed	1	2	3	4	5	6	7	8	9	10
p1: A, B, C, G	1	0	0	1	0	0	0	0	1	0
p2: A, B, C, B, C, G	1	0	1	2	0	0	0	0	1	0
p3: A, B, E, F, G	1	0	0	0	1	0	0	1	0	1
p4: A, D, E, F, G	0	1	0	0	0	1	0	1	0	1
p5: A, D, F, G	0	1	0	0	0	0	1	0	0	1
ex1: A, B, C, B, E, F, G	1	0	1	1	1	0	0	1	0	1
ex2: A, B, C, B, C, B, C, G	1	0	2	3	0	0	0	0	1	0

ex1 = p2 + p3 - p1
ex2 = 2p2 - p1



McCabe Basis Paths in the Triangle Program



V(G) = 23 - 20 + 2(1) = 5

Basis Path Set B1

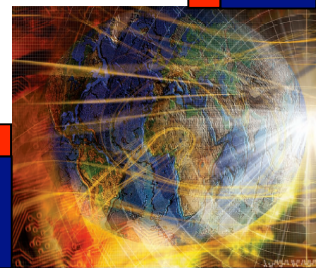
p1: 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 18, 19, 20, 22, 23	**(mainline)**
p2: 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 16, 18, 19, 20, 22, 23	**(flipped at 9)**
p3: 4, 5, 6, 7, 8, 9, 11, 12, 13, 21, 22, 23	**(flipped at 13)**
p4: 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 20, 22, 23	**(flipped at 14)**
p5: 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 19, 20, 22, 23	**(flipped at 16)**

There are 8 topologically possible paths.
4 are feasible, and 4 are infeasible.

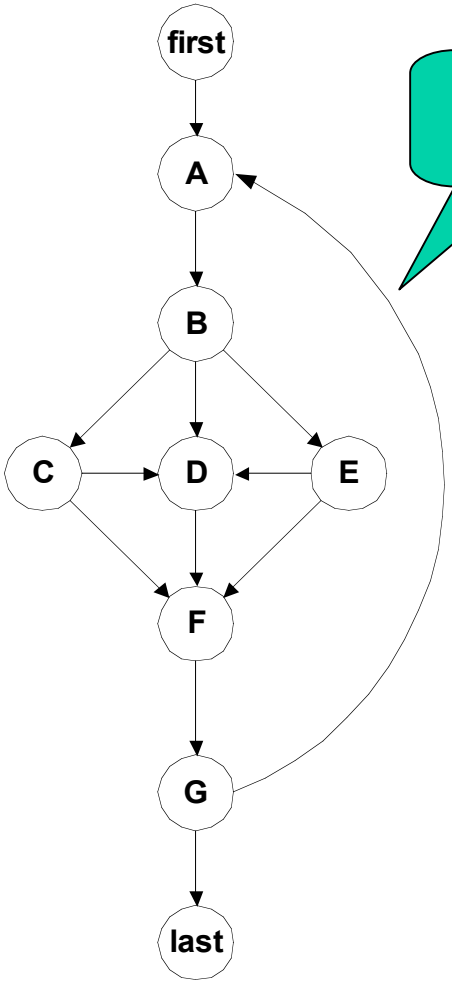
Exercise: Is every basis path feasible?

Essential Complexity

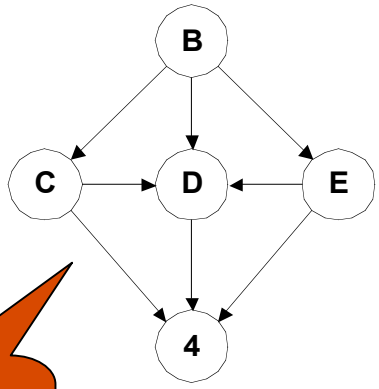
- McCabe's notion of Essential Complexity deals with the extent to which a program violates the precepts of Structured Programming.
- To find Essential Complexity of a program graph,
  - Identify a group of source statements that corresponds to one of the basic Structured Programming constructs.
  - Condense that group of statements into a separate node (with a new name)
  - Continue until no more Structured Programming constructs can be found.
  - The Essential Complexityof the original program is the cyclomatic complexity of the resulting program graph.
- The essential complexity of a Structured Program is 1.
- Violations of the precepts of Structured Programming increase the essential complexity.



Essential Complexity of Schach's Program Graph

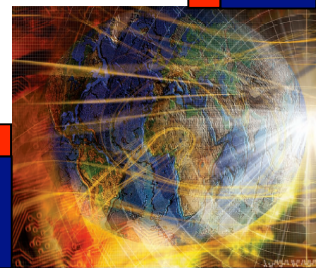


This original

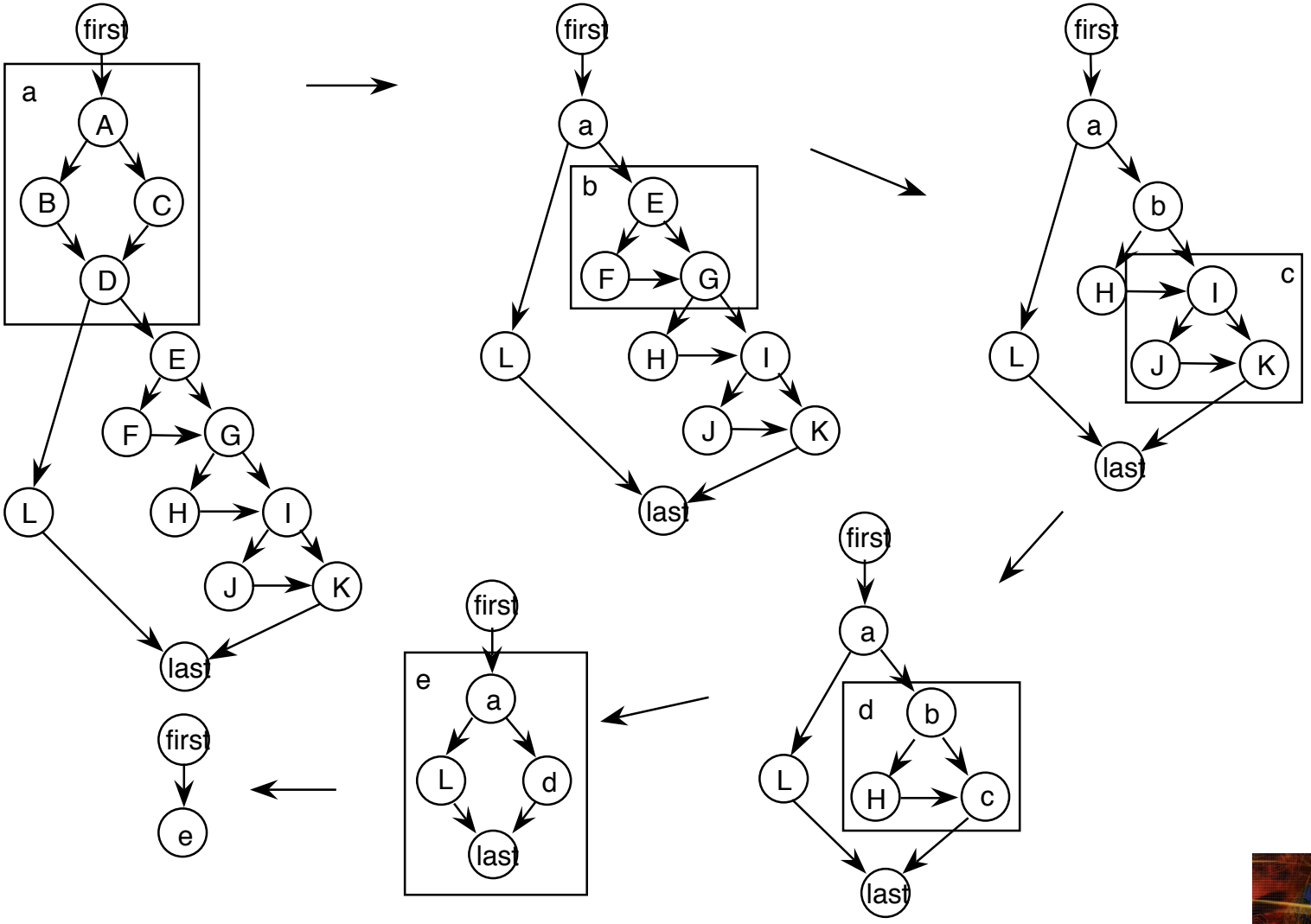


Reduces to this

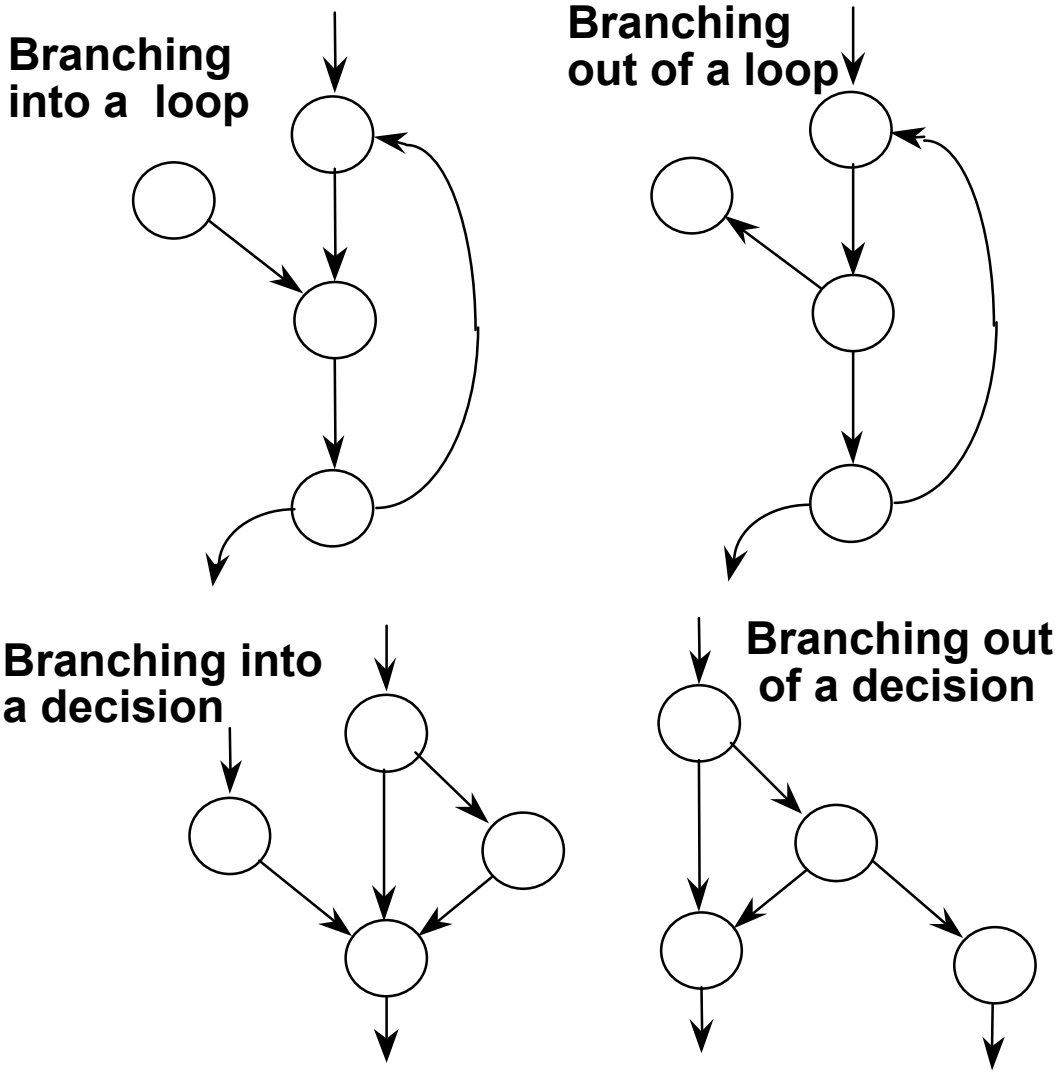
V(G) = 8 – 5 + 2(1)
= 5
Essential complexity is 5



Condensation with Structured Programming Constructs

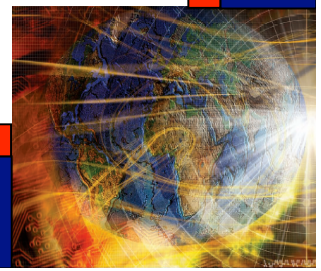


Violations of Structured Programming Precepts

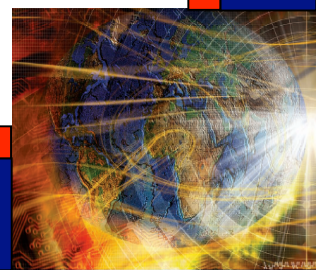


Cons and Pros

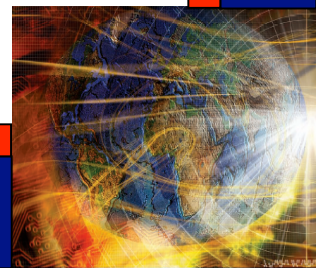
- **Issues**
  - Linear combinations of execution paths are counter-intuitive. What does 2p2 – p1 really mean?
  - How does the baseline method guarantee feasible basis paths?
  - Given a set of feasible basis paths, is this a sufficient test?
- **Advantages**
  - McCabe's approach does address both gaps and redundancies.
  - Essential complexity leads to better programming practices.
  - McCabe proved that violations of the structured programming constructs increase cyclomatic complexity, and violations cannot occur singly.

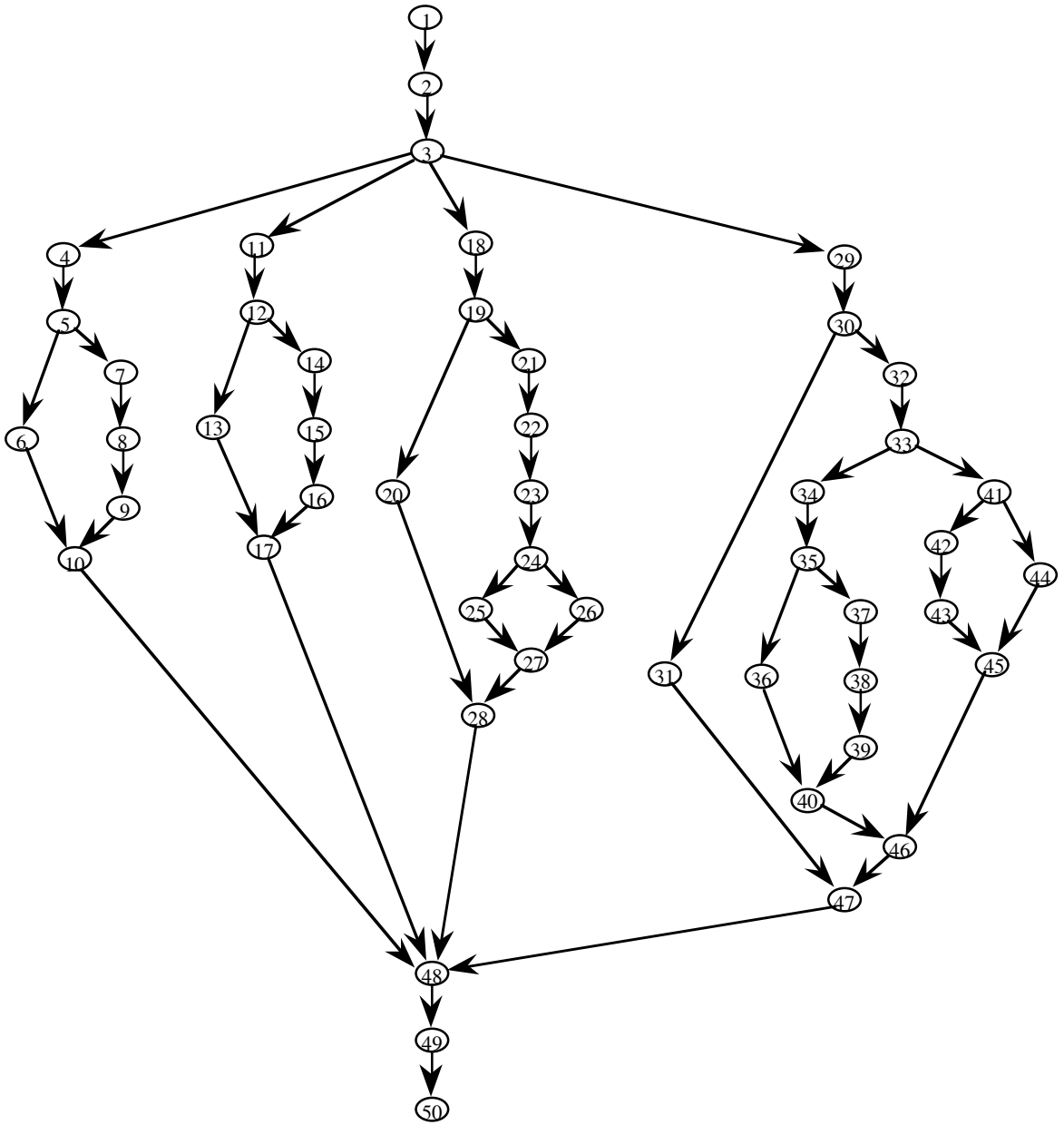


```
Program NextDate
Dim tomorrowDay,tomorrowMonth,tomorrowYear As Integer
Dim day,month,year As Integer
1.  Output ("Enter today's date in the form MM DD YYYY")
2.  Input (month,day,year)
3.  Case month Of
4.  Case 1: month Is 1,3,5,7,8,Or 10: '31 day months (except Dec.)
5.      If day < 31
6.              Then tomorrowDay = day + 1
7.              Else
8.                      tomorrowDay = 1
9.                      tomorrowMonth = month + 1
10.   EndIf
11. Case 2: month Is 4,6,9,Or 11 '30 day months
12.   If day < 30
13.             Then tomorrowDay = day + 1
14.             Else
15.                     tomorrowDay = 1
16.                     tomorrowMonth = month + 1
17.   EndIf
18. Case 3: month Is 12: 'December
19.   If day < 31
20.             Then tomorrowDay = day + 1
21.             Else
22.                     tomorrowDay = 1
23.                     tomorrowMonth = 1
24.                     If year = 2012
25.                             Then Output ("2012 is over")
26.                             Else tomorrow.year = year + 1
27.                     EndIf
28.   EndIf
```



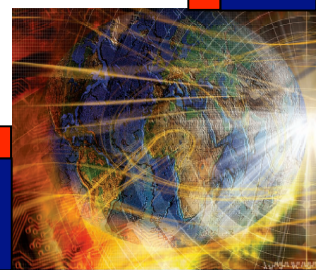
```
29. Case 4: month is 2: 'February
30.    If day < 28
31.           Then tomorrowDay = day + 1
32.           Else
33.                  If day = 28
34.                         Then
35.                                If ((year MOD 4)=0)AND((year MOD 400)π0)
36.                                       Then tomorrowDay = 29 'leap year
37.                                       Else          'not a leap year
38.                                              tomorrowDay = 1
39.                                              tomorrowMonth = 3
40.                                EndIf
41.                         Else  If day = 29
42.                                              Then tomorrowDay = 1
43.                                                     tomorrowMonth = 3
44.                                              Else  Output("Cannot have Feb.", day)
45.                                       EndIf
46.                  EndIf
47.    EndIf
48. EndCase
49. Output ("Tomorrow's date is", tomorrowMonth,
                         tomorrowDay, tomorrowYear)
50. End NextDate
```

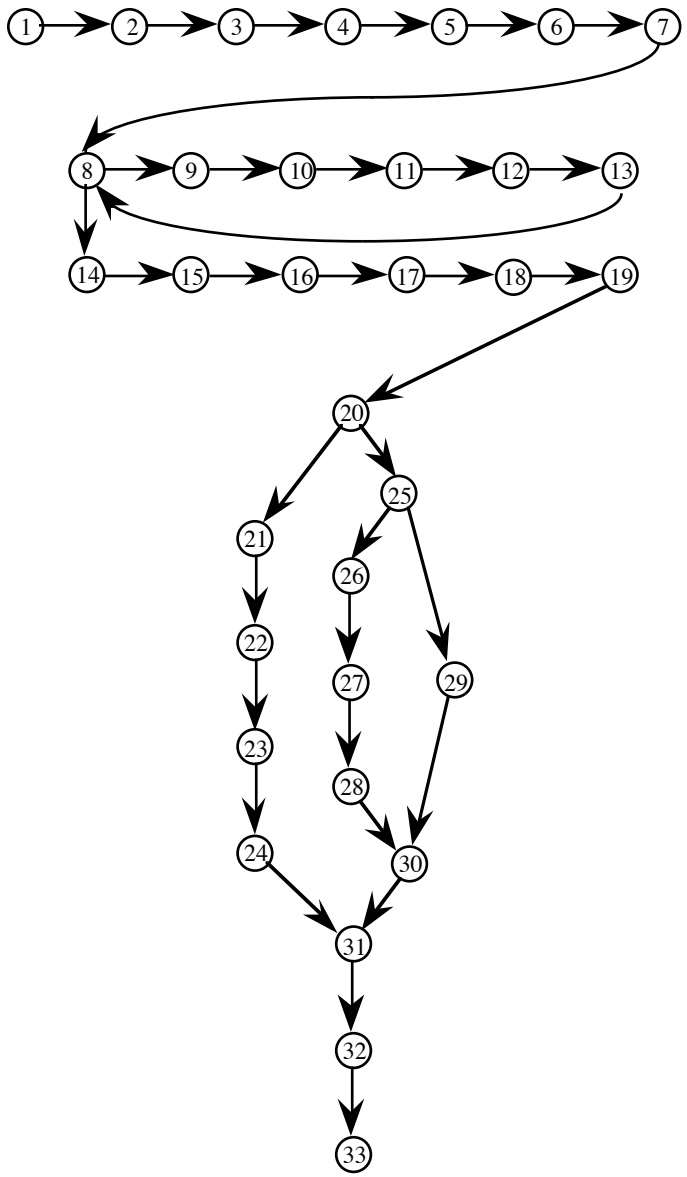



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

Commission Program Pseudo-Code

```
   Program Commission
   Dim lockPrice, stockPrice, barrelPrice As Real
   Dim locks, stocks, barrels As Integer
   Dim totalLocks, totalStocks As Integer
   Dim totalBarrels As Integer
   Dim lockSales, stockSales As Real
   Dim barrelSales As Real
   Dim sales, commission As Real
1  lockPrice = 45.0
2  stockPrice = 30.0
3  barrelPrice = 25.0
4  totalLocks = 0
5  totalStocks = 0
6  totalBarrels = 0
7  Input(locks)
8  While NOT(locks = -1)
9     Input(stocks, barrels)
10    totalLocks = totalLocks + locks
11    totalStocks = totalStocks + stocks
12    totalBarrels = totalBarrels + barrels
13    Input(locks)
14 EndWhile
15 Output("Locks sold: ", totalLocks)
16 Output("Stocks sold: ", totalStocks)
17 Output("Barrels sold: ", totalBarrels)
18 sales = lockPrice*totalLocks +
   stockPrice*totalStocks  + barrelPrice * totalBarrels
19 Output("Total sales: ", sales)
20 If (sales > 1800.0)
21    Then
22       commission = 0.10 * 1000.0
23       commission = commission + 0.15 * 800.0
24        commission = commission + 0.20*(sales-1800.0)
25  Else If (sales > 1000.0)
26     Then
27       commission = 0.10 * 1000.0
28       commission = commission + 0.15*(sales-1000.0)
29    Else commission = 0.10 * sales
30    EndIf
31 EndIf
32 Output("Commission is $", commission)
33 End Commission
```




1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33