Branching

Unit: Coding

Problem Area: Programming Basics

Lesson: Branching

Student Learning Objectives.

Instruction in this lesson should result in students achieving the following objectives:

1. **Determine the nature of a condition.**
2. **Explain Boolean logic.**
3. **Use branching statements.**

Resources.

The following resources may be useful in teaching this lesson:

E-units(s) corresponding to this lesson plan. CAERT, Inc, http://www.mycaert.com.

Introduction to Creating Flowcharts, YouTube. Accessed October 29, 2019. https://youtu.be/SWRDqTx8d4k.

Microsoft product screenshot(s) reprinted with permission from Microsoft Systems Incorporated.

Equipment, Tools, Supplies, and Facilities

- ✓ Overhead or PowerPoint projector
- ✓ Visual(s) from accompanying master(s)
- ✓ Copies of sample test, lab sheet(s), and/or other items designed for duplication
- ✓ Materials listed on duplicated items
- ✓ Computers with printers and Internet access
- ✓ Classroom resource and reference materials
- ✓ PC with Internet Browser Software
- ✓ Internet access

Key Terms.

The following terms are introduced in this lesson (shown in bold italics):

- Boolean expression
- Comparison operators
- Condition
- Decision Structure
- Dual alternative decision structure
- Flowcharts
- Logical operators
- Multi-alternative decision structure
- Nested Decision structure
- Pseudocode
- Single alternative decision structure

Interest Approach.

Use an interest approach that will prepare the students for the lesson. Teachers often develop approaches for their unique class and student situations. A possible approach is included here.

> *Computer programs contain many statements that need to be executed some of the time. For example, a program may contain code to display "Good morning" when the time is between 8 A.M. and 11:59 A.M. in the morning and "Good evening" when time is between 12:00 P.M. and 11:59 P.M. This logic is accomplished in a computer program using branching (or decision) statements, where each branch executes a different set of code. This lesson examines the process of coding statements that evaluate decision statements and implement branching logic in computer programs.*

CONTENT SUMMARY AND TEACHING STRATEGIES

Objective 1: Determine the nature of a condition.

Anticipated Problem: What is a condition?

I. In a computer program, not all statements are executed each time the program runs. Certain statements only execute when a specific condition is met. This section discusses constructing conditions and their effect of program logic.

A. USING FLOWCHARTS. Developing code is a complex venture. It requires knowledge of the programming language used to create the program, and an understanding of the logic to be implemented in code. Therefore, early in the days of programming, a system of depicting logic using symbols called Flowcharting gained prevalence. ***Flowcharts*** depict logic using symbols and shapes and are language independent. Since they are graphical, it is easier to understand program logic than it is to read code. Flowcharts, is a sense, are a language all by themselves with different shapes used to represent different operations. Flowcharts are developed before writing code. Once logic for a problem has been finalized, the flowchart is translated into a programming language. See VM–A for some of the commonly used flowcharting symbols. For example, a rectangle is used to depict variable declaration and computations, and a parallelogram is used to represent input and output operations. In some schools of flowcharting, declaration statements are shown with lines inside. When a program needs to display a message to the user, the text inside the parallelogram can say “Print” or “Display”. The action is better represented by the “Display”, but “Print” is used even though it sounds like an action to print on a printer. This is because languages, such as Python, use a command called “print” to display messages to the user. This lesson uses “Display” as the action instead of “Print”.

B. CONDITIONS AND DECISION STRUCTURES. A ***Condition*** is an expression in a program that evaluates to a True or a False. Different sets of instructions execute when a condition returns a value of True or a False. See VM–B for examples of conditions that a program may need to handle. Conditional logic is implemented in a programming language using a ***Decision Structure***. Decision structures are also referred to as Selection structures or Branching structures. Decision structures are depicted in a Flowcharts using a diamond shape. A decision structure starts with a condition. The condition/question is placed inside the diamond shape. A condition returns a True or a False, and the flowchart shows instructions that execute when the condition returns a True or a False. See VM–C for an examples of decision structures that execute code when the condition returns a true.

C. TYPES OF DECISION STRUCTURES. There are three types of Decision structures and these are discussed here. Single alternative and dual alternative decision structures make use of conditions that return true and false. A multi-alternative decision structure contains multiple paths depending upon the value in a variable.

1. *Single alternative decision structure* : Consider a decision structure that wishes to perform additional steps when a condition returns a true. For example, when an employee sells more than $100,000 worth of goods, a bonus is awarded. This is depicted in VM–D. Since there are statements that need to only execute when the condition is true, this type of decision structure is called a ***Single alternative decision structure*** and has only a single branch. A Single Alternative Decision structure cannot be written so that code executes when the condition returns False.
2. *Dual alternative decision structure*: A ***Dual alternative decision structure*** contains statements that need to be executed when a condition returns a True and when a condition returns False. It contains 2 branches - one that executes on True and one that executes on a False. Suppose that code is used to set price for a product. Its regular price is $4.99 and when it is on sale, its rice is 2.99. This logic can be implemented using a Dual alternative decision structure as shown in VM–E. Different sets of actions are performed when the condition returns True and when it returns False. The problem shown in VM–E can also be coded as shown in VM–F. The question asked in VM–F is the opposite of that asked in VM–E. Both sets of code are correct. Note that in both examples, the assumption is that the user enters a “Y” or “N”. If neither of these is entered, the program will not work as expected. Code should be written to validate used input.
3. *Multi-alternative decision structure*: ***A Multi-alternative decision structure*** is one that contains multiple logical paths based upon the value of a variable. Consider the example of code that displays messages based upon the letter grade achieved by students. If Decision structures are used, code will need to contain 5 decision structures to display the message associated with each of the letter grades. Some languages offer a Multi-alternative decision structure where a question yields more than a True/False answer. See VM–G. The decision shape, the diamond, asks a question about the value of the letter_grade variable. Right after the decision, there is a 5-way branch, with each branch corresponding to a letter grade. Multi-alternative decision structures allow for logic to be coded in a very succinct fashion. However, this type of decision structure cannot execute directly at the machine level, where only single alternative and dual alternative decisions are implementable. A Multi-alternative decision structure is eventually converted into multiple simple Decision structures before execution.
4. *Nested Decision Structures*. A ***Nested Decision structure*** is a structure where there is a decision structure underneath the “Yes” portion or the “No” portion of a decision structure. See VM–H for an example. The problem states that price per unit depends upon the number of units purchased. When number of units is <= 100, price is $3.00. When number of units is more than 100, another decision structure is created to check if it is <= 300. Since there is a

decision structure inside the "Else" portion of the original decision, the larger decision structure is called a Nested Decision structure. Note that the "inner" Decision structure is completely nested inside the "outer" Decision structure.

Teaching Strategy: *Many techniques can be used to help students master this objective. Use VM–A through VM–H to understand conditions and Decision structures.*

Objective 2: Explain Boolean logic.

Anticipated Problem: What is Boolean logic?

II. Boolean logic is the study of Boolean expressions. A Simple Boolean expression consists of 2 operands and a relational operator. Simple Boolean expressions may be combined to create complex Boolean expressions and their result is explained by Boolean algebra.

A. SIMPLE BOOLEAN EXPRESSIONS. When 2 values are compared, the result of the comparison operation is either a yes or a no. For example, when a person's age is checked to see if it is greater than or equal to 21, the answer is either a yes or a no. In Boolean terms, the answer is either a True or a False. A simple ***Boolean expression*** compares 2 operands of the same datatype and the comparison is performed using a ***Relational operator***. The 2 operands can be literals or variables. See VM–I for the list of relational operators used in computer programming languages. Some of the relational operators, such as the greater-than-equal-to operator, are composed of 2 symbols from the keyboard. This is because in the early days of programming, the keyboard used to key in programs was the same one used by typewriters, and it did not have the ≠?, ≥?, or the ≤? symbols on it. Therefore, a combination of symbols found on the keyboard was used to represent certain operations, such as the >= operator. No space is left between the 2 symbols and the order of the 2 symbols is significant. For example, the >= operator cannot be replaced by the =< operator. Some languages do not use the = symbol is not used to check equality of 2 operands because the = operator is used as an assignment operator and is used to assign a value to a variable. Instead the == operator is used to check equality of 2 operands.

B. COMPLEX BOOLEAN EXPRESSIONS. Many real-world computer-based operations require that a value be compared with more than one other value. For example, there may be a requirement to ensure that a test score fall within a range. In such a case, the test score must be compared with the lower boundary value and the upper boundary value. Computers, however, can only compare two values at a time. When a value, such as a test score, must be checked to see if it is within a range, multiple Boolean expressions must be coded and tested. ***Logical operators*** are used to combine simple Boolean expressions to create a complex Boolean expression. There are 3 logical operators used in programming and these are AND, OR, and NOT.

1. *The AND Logical Operator:* When Boolean expressions are combined using AND, each individual Boolean expression must evaluate to True for the combined expression to return a True. The AND logical operator is frequently used to ensure that a value lies within a range. See VM–J. The user is requested to enter a test score in the range 0-100. A condition is built to check for data validity. 2 Boolean expressions are coded. These are test_score >= 0 and test_score <= 100. Both Boolean expressions must be true and are connected using the AND Logical operator. Note that the Boolean expression cannot be coded as test_score >= 0 AND <=100. When the AND logical operator is used, it must be placed between 2 complete Boolean expressions, even if it means repeating the name of the variable in each of the 2 Boolean expressions.
2. *The OR Logical Operator:* When Boolean expressions are combined using OR, one of the 2 individual Boolean expression must evaluate to True, or they both may return True for the combined expression to return a True. The OR logical operator is shown in VM–K. It uses the OR logical operator to check if user input is invalid. Once again, the user is requested to enter a test score in the range 0-100. A condition is built to check for invalid data. The 2 Boolean expressions, test_score < 0 and test_score > 100 are connected using the logical operator OR. Only 1 of these 2 expressions needs to return True for the condition to be True and indicate that the user entered value is invalid.
3. *The NOT Logical Operator:* The NOT logical operator is used to invert a condition. Consider a program that asks the user to enter a letter grade ("A", "B", "C", "D", or "F"), and displays a message when it is invalid. While it can be written in several different ways, VM–L demonstrates this logic using the NOT operator. The condition for good data is built and then placed in parentheses, with the NOT placed before the (symbol. The "good" condition is inverted using the NOT operator to obtain the condition for "bad" or invalid data.

Teaching Strategy: *Many techniques can be used to help students master this objective. Use VM–I through VM–L to understand Boolean Algebra and Logical operators.*

Objective 3: Use branching statements.

Anticipated Problem: What are branching statements and how are they used in code?

III. Decision structures allow programs to pose questions and perform different sets of code when the answer is a True and when the answer is a False. This logic was demonstrated with Flowcharts in previous sections, and this section converts the logic into pseudocode.

A. PSEUDOCODE: ***Pseudocode*** is an English-like language that is used to depict logic. It is much closer to computer programming language code than a flowchart. Pseudocode was used in previous lessons to declare variables and perform actions as shown in VM–M. Pseudocode cannot directly execute on any computer.

It appears to be in a "computer language-like" format but is not computer code. Pseudocode is written in statements that, in many cases, can be converted, line by line, into program code. While drawing flowcharts is a more free-flowing exercise in creating logic structures, pseudocode creation is more structured and serves as a medium to start the process of converting logic structures from graphical representations to stringent program code.

B. DECISIONS IN PSEUDOCODE. Decisions are coded in Pseudocode using the "If" and the "Then" keywords. It may include the "Else" keyword as well.

1. *Single-alternative decision structure*: In the previous section, single-alternative decision structures were discussed. See VM–N for pseudocode for the single-alternative decision structure that was discussed in the previous section. A Decision structure is created to Award bonus when sales is more than $100,000. In the pseudocode, the word "If" is followed by the question asked in the condition. When it returns a True, statements that need to execute are displayed at an indent. The "End If" word represents the end of the decision structure. The "End If" must be placed on the same level of indent as the "If" statement.
2. *Dual-alternative decision structure*: See VM–O. It shows the logic to price an item. Its regular price is $4.99 and when it is on sale, its rice is 2.99. This logic is implemented using a Dual alternative decision structure. In the Pseudocode, the decision structure, once again, starts with the word "If". After the statements that need to execute when the condition returns True, the "Else" word is coded. Statements that execute when the condition returns False are placed between "Else" and "End If". Note that no condition or question is placed after the "Else" word. Control in the flowchart transfers to the "Else" portion of the code when the condition in the "If" statement returns a False. The "If", "Else", and "End If" are placed at the same level of indent. The problem shown in VM–O can also be coded as shown in VM–P. The question asked in VM–P is the opposite of that asked in VM–O. Both sets of code are correct.
3. *Multi-alternative decision structure*: A Multi-alternative decision structure is created to provide alternative paths based upon the value in a single variable. See VM–Q for an example of a multi-alternative decision structure. The value of the variable for each branch is placed after the word "Case". The last branch uses "Other" which is analogous to the "Else" statement. Control passes here when none of the other branch's "Case" value applies to the variable. In most languages, the "Case" statements look for exact matches, and cannot be used to check for ranges.
4. *Nested Decision Structures*. A ***Nested Decision structure*** is a structure where there is a decision structure underneath one of its 2 branches. Each Decision structure is coded with "If", "Else" and "End If". Nesting of Decision structures is complete; the inner Decision structure is completely nested inside the outer Decision structure. See VM–R for an example of a nested decision structure. In this example, the price of a product is based upon the number of units purchased. The first decision determines if the number of units purchased is less

than or equal to 100. If this condition is true, price is set to 3.99. If this condition is false, it means that the number of units purchased is larger than 100. The next decision checks to see if the number of units purchased is less than equal to 300, and if the condition is true, the price is set to 2.99. If the condition is false, it means that the number of units is larger than 300 and price is set to 1.99. There are multiple ways to accomplish this pricing logic. It can be performed using the logic shown in VM–S. Here, 3 separate stand-alone decisions are developed using logical operators. Both sets of logic are valid, and the one chosen is based upon programmer preference.

Teaching Strategy: *Many techniques can be used to help students master this objective. Use VM–M through VM–S to understand how branching is implemented in pseudocode.*

- **Review/Summary.** Use the student learning objectives to summarize the lesson. Have students explain the content associated with each objective. Student responses can be used in determining which objectives need to be reviewed or taught from a different angle. Questions at the ends of chapters in the textbook may also be used in the Review/Summary.

- **Application.** Use the included visual master(s) and lab sheet(s) to apply the information presented in the lesson.

- **Evaluation.** Evaluation should focus on student achievement of the objectives for the lesson. Various techniques can be used, such as student performance on the application activities. A sample written test is provided.

- **Answers to Sample Test:**

Part One: Completion

1. Flowcharts
2. Rectangle
3. Condition
4. Diamond
5. Single alternative decision
6. If

Part Two: True/False

1. T
2. F
3. F
4. F

5. T
6. T

Part Three: Short Answer

1. Flowcharts depict logic using symbols and shapes. They are easier to understand that program code.
2. Pseudocode is written in simple text and be easily converted into program code.
3. A nested decision structure is a structure where there is a decision structure underneath one of the 2 branches of a decision structure.

Sample Test

Name _____

Branching

Part One: Completion

Instructions: **Provide the word or words to complete the following statements.**

1. _____ represent program logic using graphics and symbols.
2. A _____ shape represents a variable declaration.
3. In a computer program a _____ returns a True or false value.
4. Branching is depicted with a _____ shape.
5. A Decision structure that only contains a single branch is called a _____ structure.
6. In pseudocode, decisions are coded using the word _____.

Part Two: True/False

Instructions: **Write *T* for True or *F* for False.**

_____1. Flowcharts use symbols to depict logic.

_____2. Pseudocode uses symbols to depict logic.

_____3. A Boolean Expression returns 3 values — True, False, Neutral.

_____4. Two Boolean expressions may be combined using a Relational operator.

_____5. Decision structures begin with the "If" word in Pseudocode.

_____6. In Pseudocode, dual-alternative decision structures make use of the "Else" word.

Part Three: Short Answer

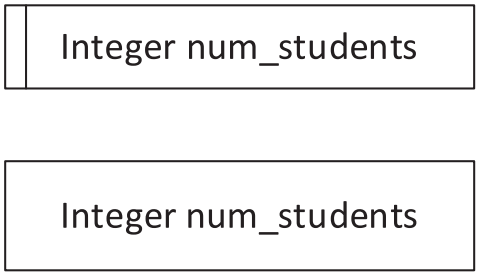
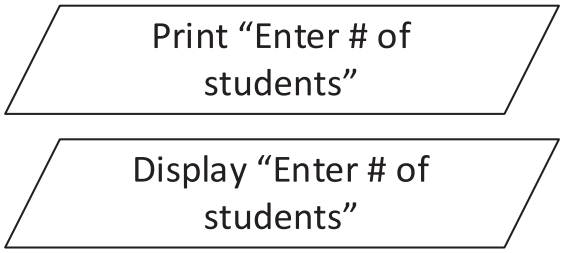
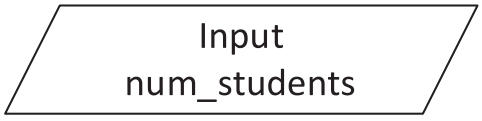
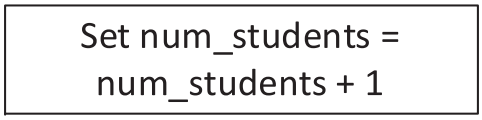
Instructions: **Answer the following.**

1. Why are Flowcharts used?

2. Why is Pseudocode used?

3. What is a nested decision structure?

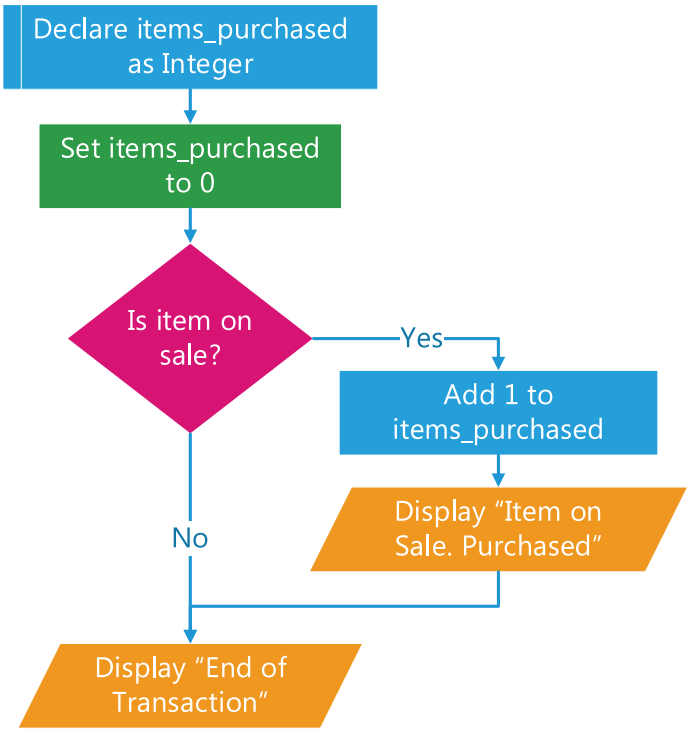
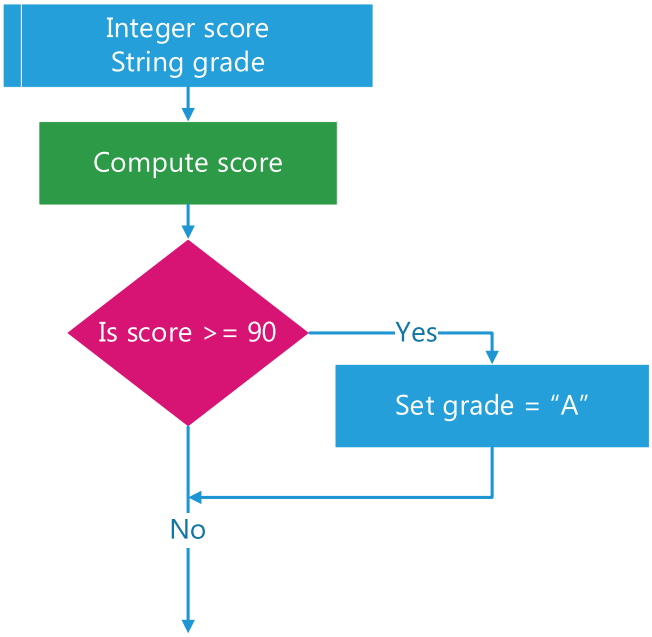
FLOWCHART SYMBOLS

Description	Flowchart Symbol
Declare an Integer variable called **numStudents**	
Print "Enter number of students" Display " Enter number of students"	
Ask the user to enter a value to be placed in the **numStudents** variable	
Add 1 to **numStudents**	

CONDITIONS IN A COMPUTER PROGRAM

Example #	Description	Condition
Example 1	Purchase item when it is on sale	Is item on sale?
Example 2	Assign an "A" grade when score is 90 or more	Is score >= 90?
Example 3	Display item when its quantity is 0	Is quantity == 0?
Example 4	Award fulltime employees a bonus	Is employee a fulltime employee?

DECISIONS IN FLOWCHARTS

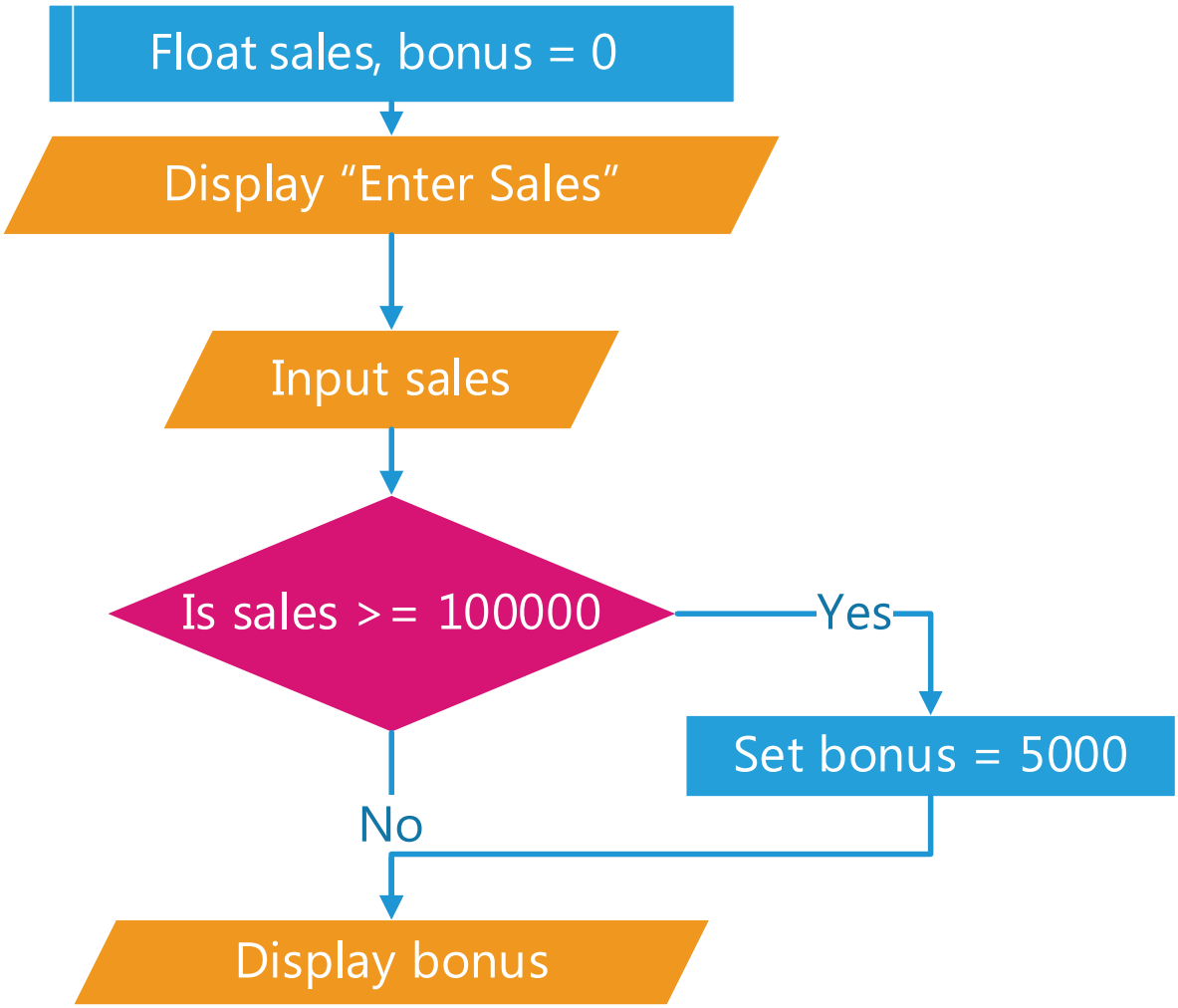
Description	Flowchart
Purchase only those items that are on sale	
Assign an "A" grade when score is 90 or more	

SINGLE ALTERNATIVE DECISION STRUCTURE

Problem

Award bonus when sales is more than $100,000

Flowchart

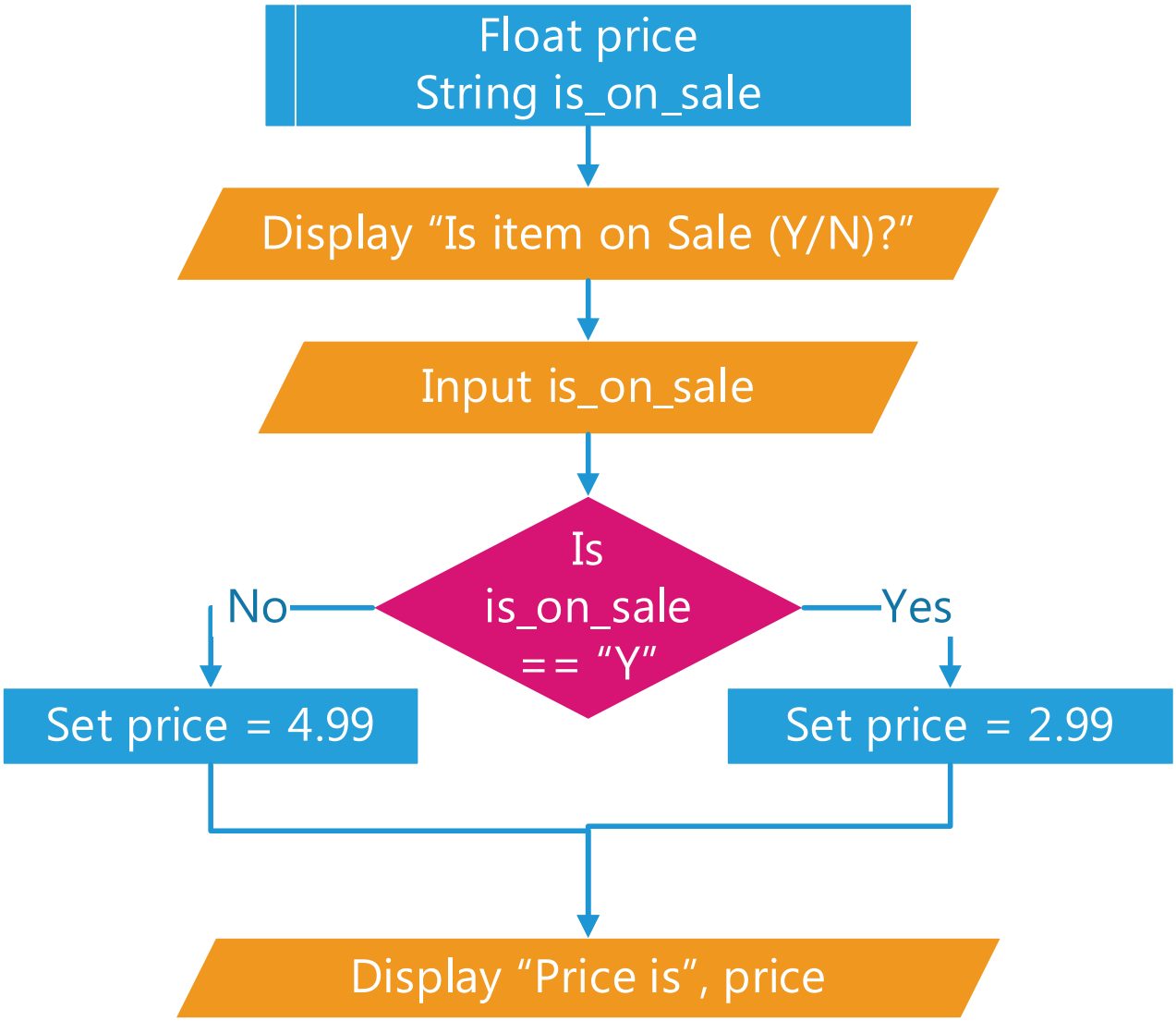


DUAL ALTERNATIVE DECISION STRUCTURE — I

Problem

Regular price is 4.99. When it is on sale, price is 2.99

Flowchart



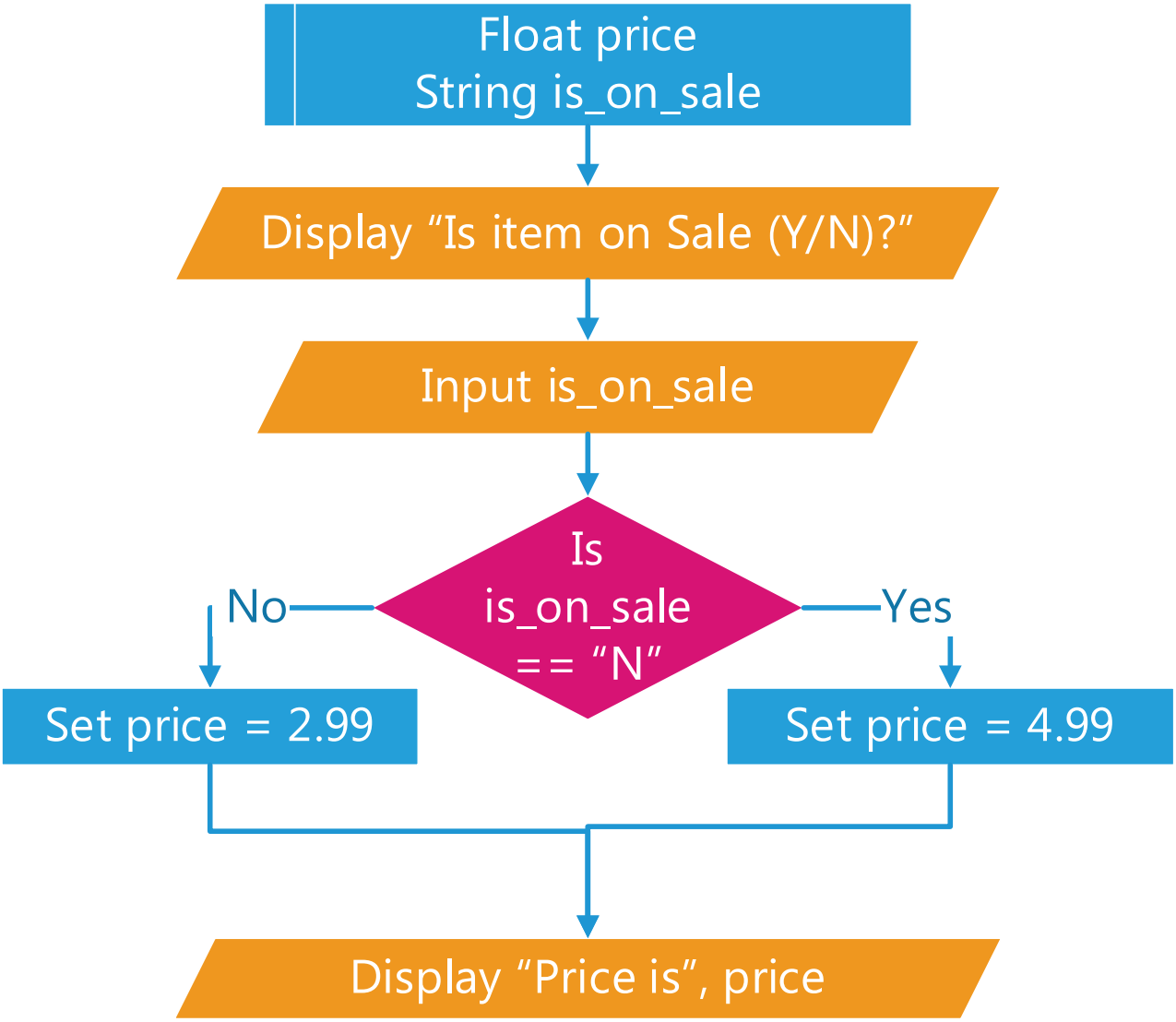
VM–F

DUAL ALTERNATIVE DECISION STRUCTURE — II

Problem

Regular price is 4.99. When it is on sale, price is 2.99

Flowchart

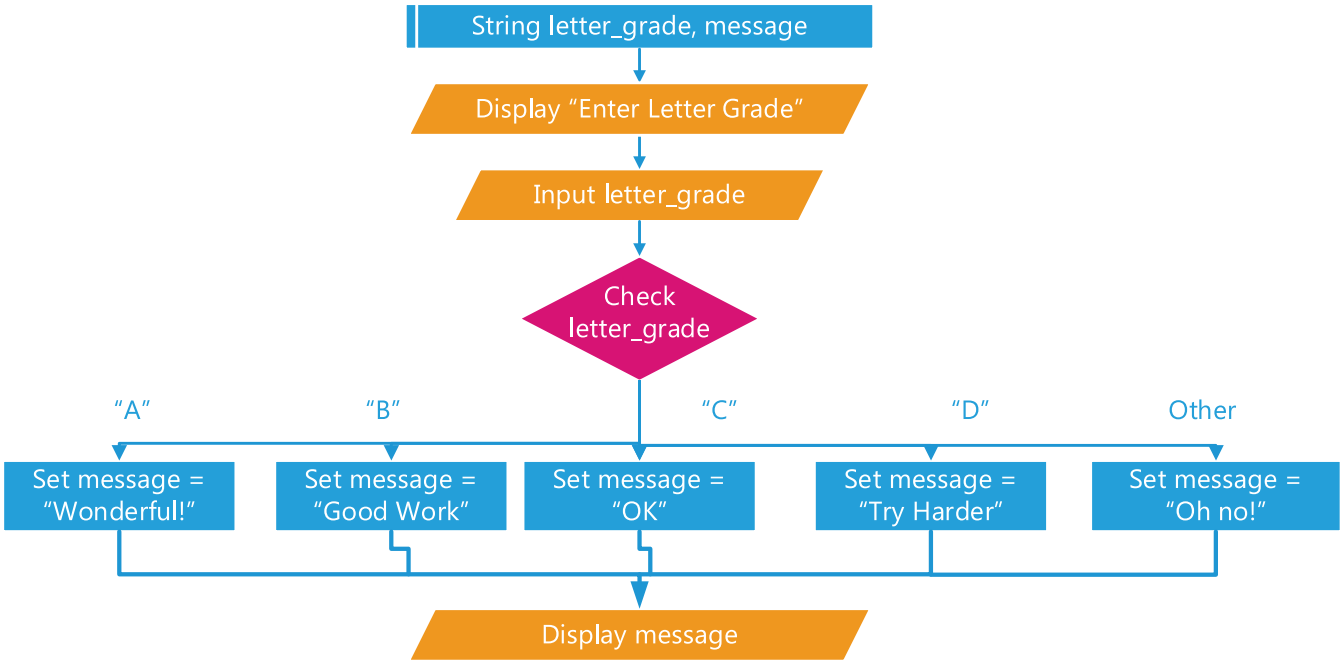


MULTI-ALTERNATIVE DECISION STRUCTURE

Problem

Display message based upon letter grade

Flowchart

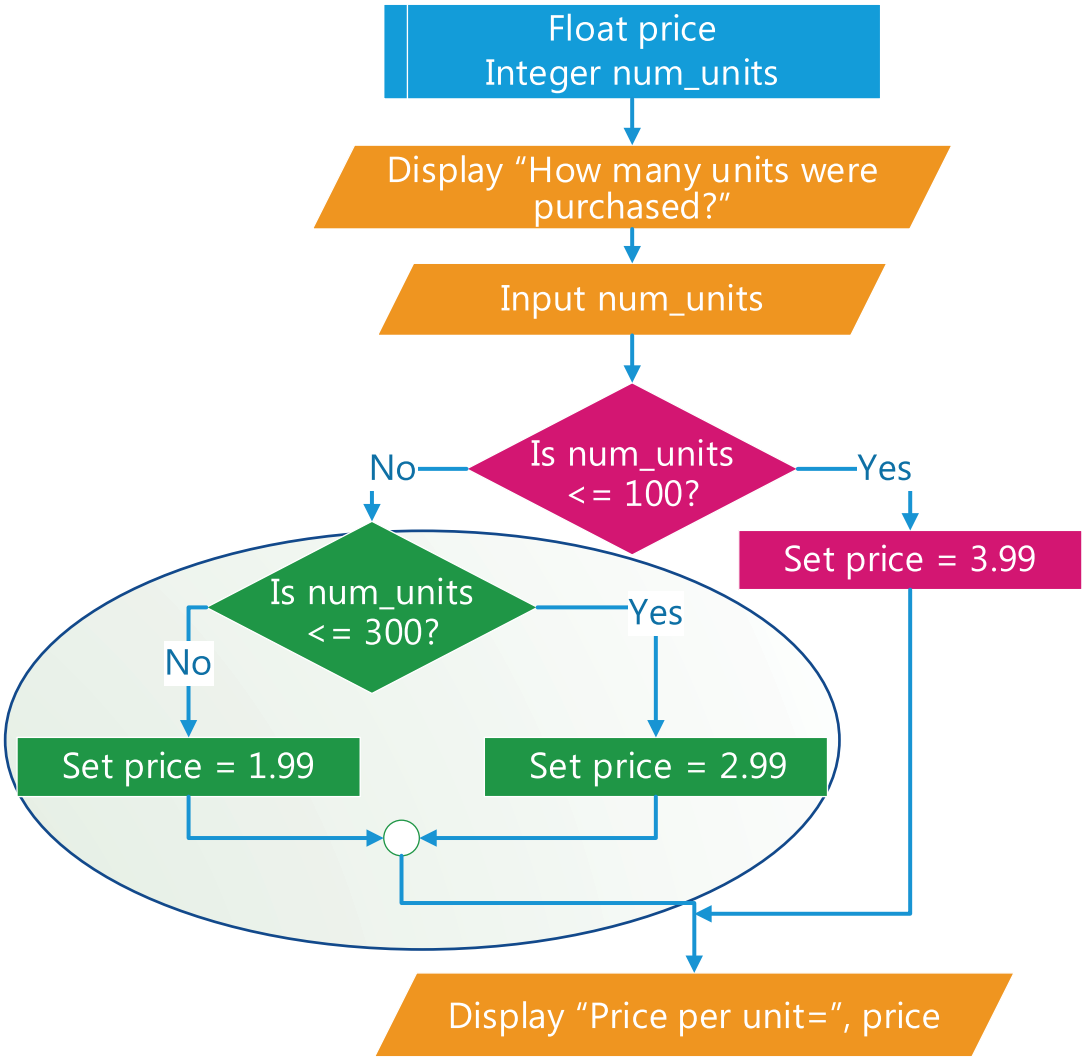


NESTED DECISION STRUCTURE FLOWCHART

Problem

Number of units ordered	Price per unit
0-100	$3.99
101 - 300	$2.99
301 units or more	$1.99

Flowchart



RELATIONAL OPERATORS

Relational Operator	Description	Operator Used in Math
>	Greater than	>
<	Less than	<
>=	Greater than equal to	≥
<=	Less than equal to	≤
==	Equals	=
!=	Not equal to	≠

USING THE AND LOGICAL OPERATOR

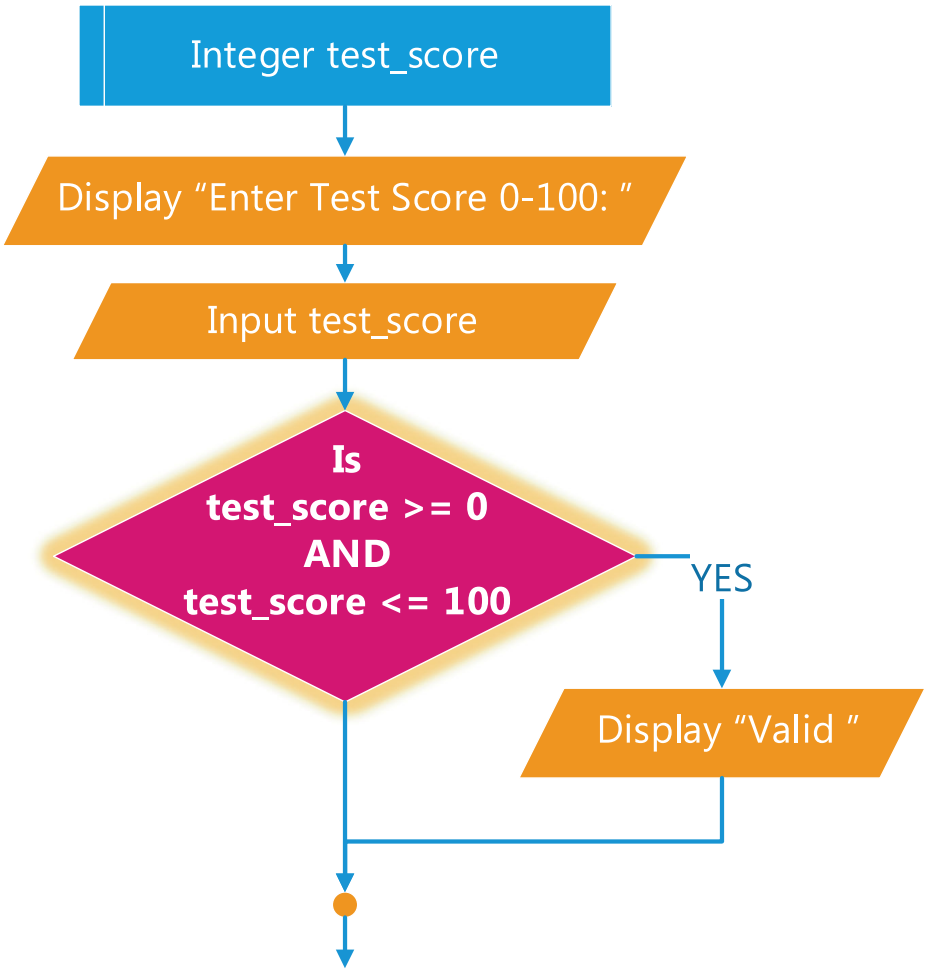
Problem

User must enter a value 0-100

Flowchart

Boolean Expressions used in Decision

- `test_score >= 0`
- `test_score <= 100`



USING THE OR LOGICAL OPERATOR

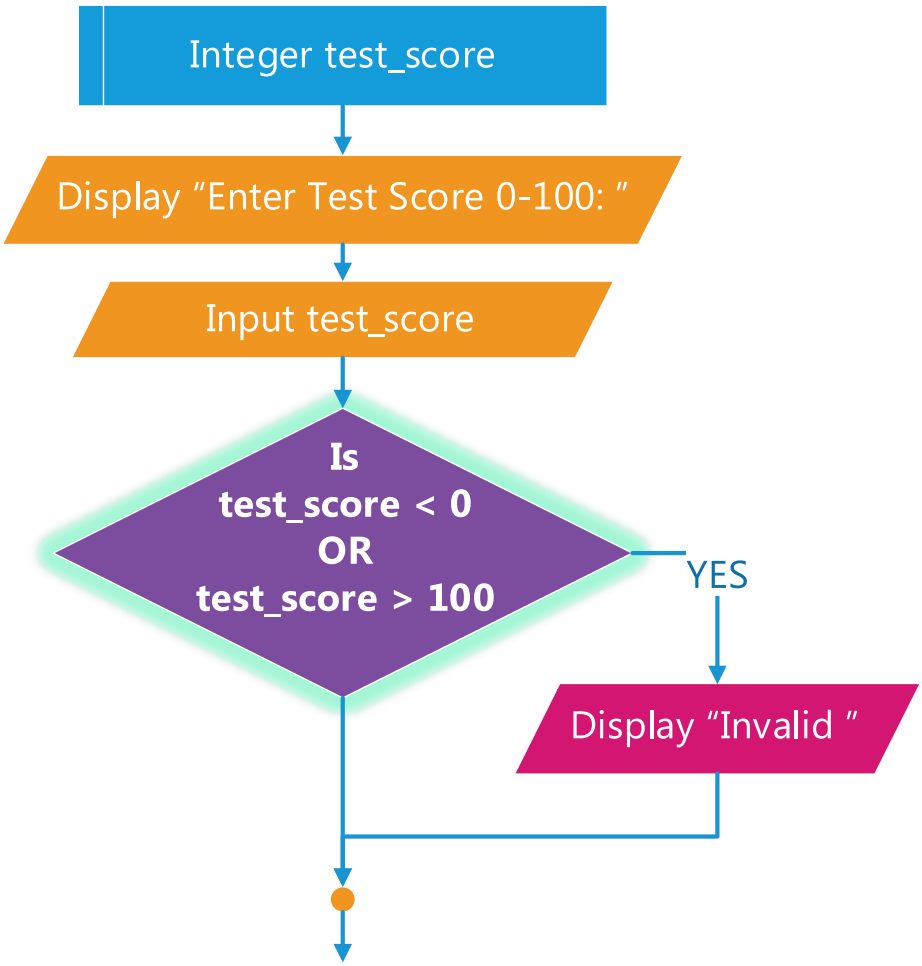
Problem

User must enter a value 0-100

Flowchart

Boolean Expressions used in Decision

- `test_score < 0`
- `test_score > 100`



USING THE NOT LOGICAL OPERATOR

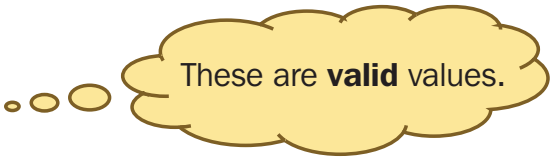
Problem

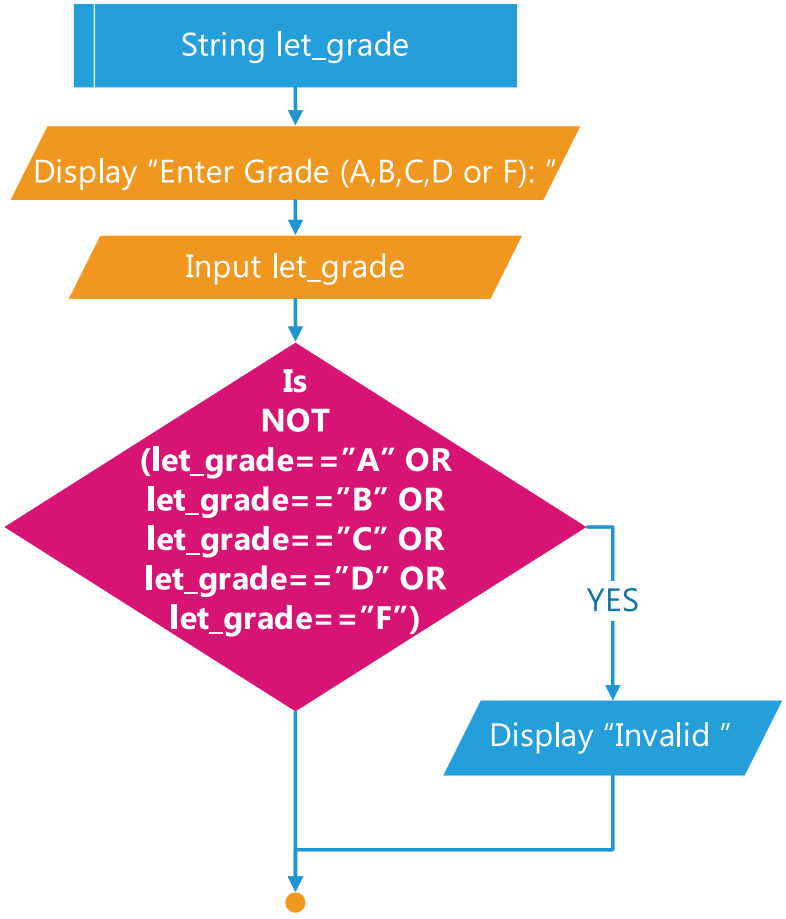
User must enter a value "A", "B", "C", "D" or "F"

Flowchart

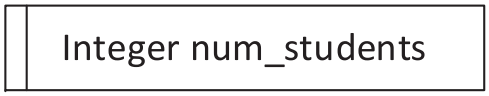
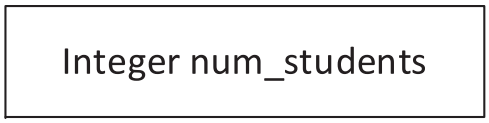
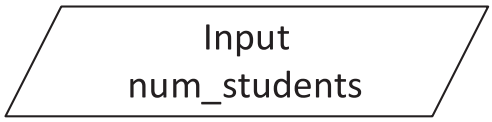
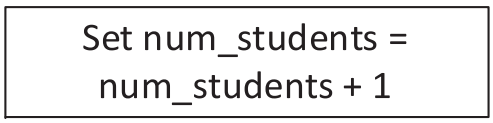
Boolean Expressions used in Decision

- `let_grade == "A"`
- `let_grade == "B"`
- `let_grade == "C"`
- `let_grade == "D"`
- `let_grade == "F"`





FLOWCHART SYMBOLS & PSEUDOCODE

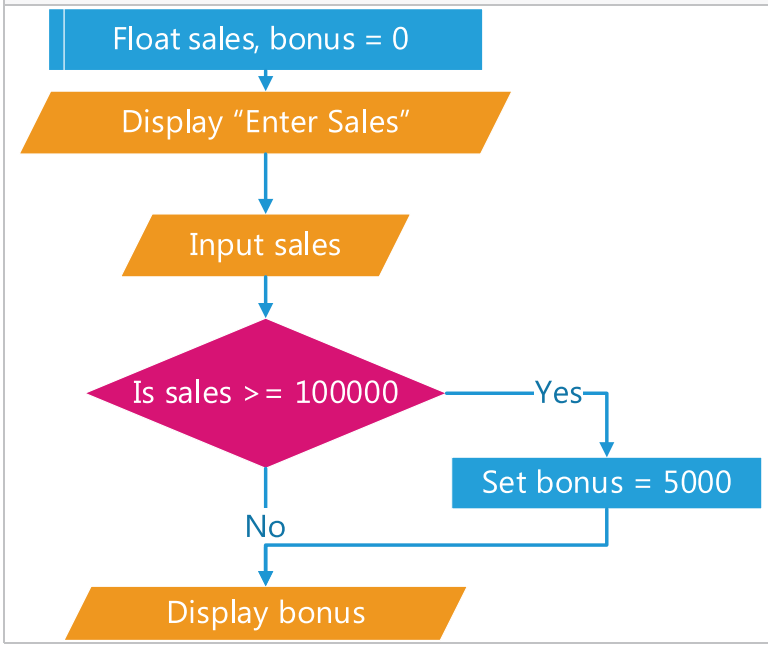
Flowchart Symbol	Pseudocode
 	Declare numStudents as Integer
 	Print "Enter number of students" Display " Enter number of students"
	Input numStudents
	Set numStudents = numStudents + 1

SINGLE ALTERNATIVE DECISION STRUCTURE PSEUDOCODE

Problem

Award bonus when sales is more than $100,000

Flowchart	Pseudocode

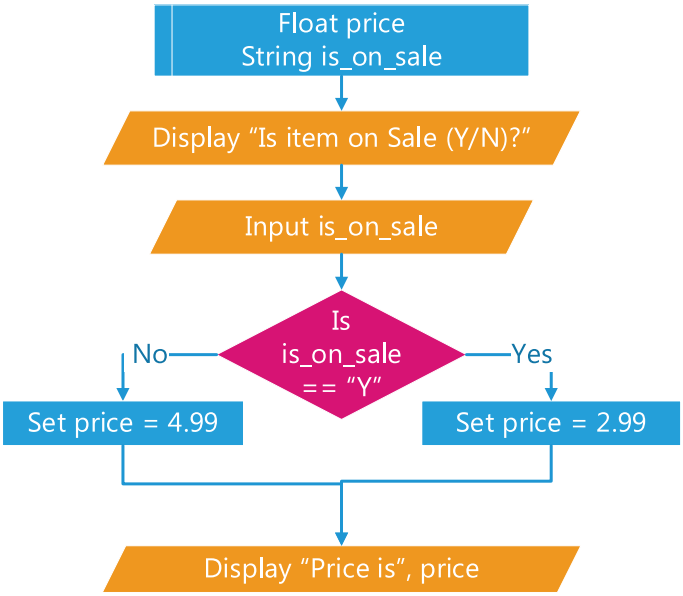
	`Declare sales, bonus = 0 as Float`
	
	`Display "Enter sales"`
	`Input sales`
	
	`If sales > 100000 Then`
	`  Set bonus = 5000`
	`End If`
	
	`Display bonus`

DUAL ALTERNATIVE DECISION STRUCTURE PSEUDOCODE — I

Problem

Regular price is 4.99. When it is on sale, price is 2.99

Flowchart



Pseudocode

```
Declare price as Float
Declare is_on_sale as String

Display "Is Item in Sale (Y/N)?"
Input is_on_sale

If is_on_sale == "Y" Then
     Set price = 2.99
Else
     Set price = 4.99
End If

Display "Price is ", price
```

No condition or question is placed after "Else"

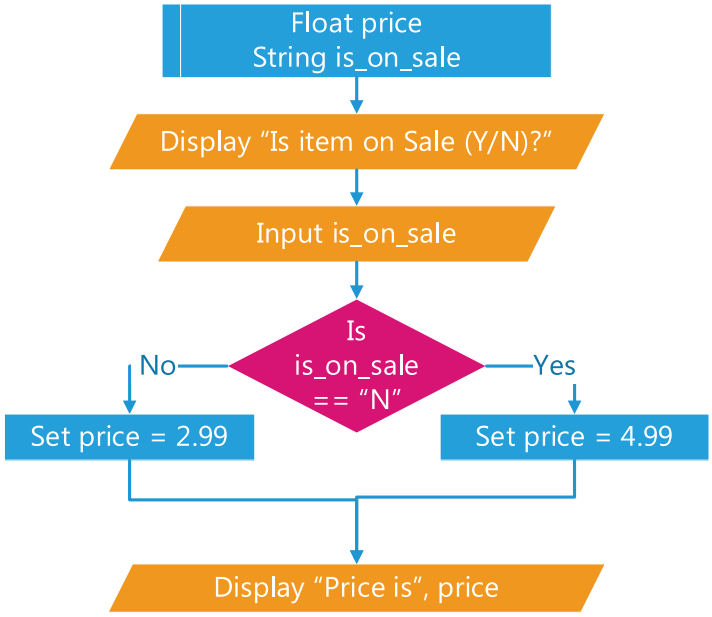
VM–P

DUAL ALTERNATIVE DECISION STRUCTURE
DUAL PSEUDOCODE—II

Problem

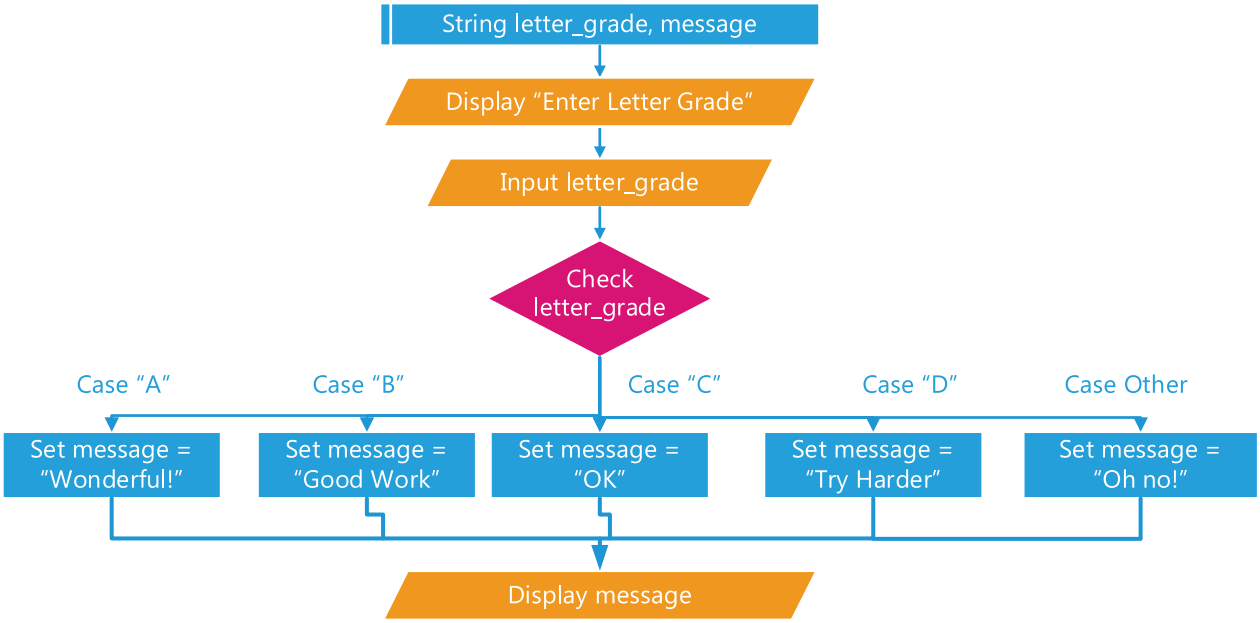
Regular price is 4.99. When it is on sale, price is 2.99

Flowchart	Pseudocode

	`Declare price as Float`
	`Declare is_on_sale as String`
	
	`Display "Is Item in Sale (Y/N)?"`
	`Input is_on_sale`
	
	`If is_on_sale == "N" Then`
	`  Set price = 4.99`
	`Else`
	`  Set price = 3.99`
	`End If`
	
	`Display "Price is ", price`

MULTI-ALTERNATIVE DECISION STRUCTURE PSEUDOCODE

Flowchart



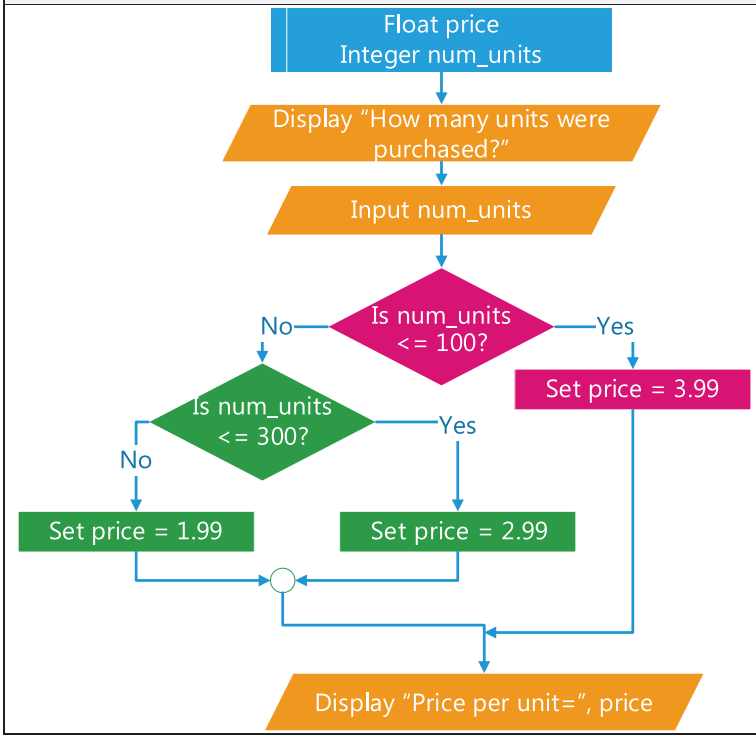

```
Declare letter_grade, message as String

Display "Enter Letter Grade"
Input letter_grade

Case of letter_grade
     Case "A": Set message = "Wonderful!"
     Case "B": Set message = "Good Work"
     Case "C": Set message = "OK"
     Case "D": Set message = "Try Harder"
     Case "F": Set message = "Oh no!"
End Case

Display message
```

NESTED DECISION STRUCTURE PSEUDOCODE

Problem

Number of units ordered	Price per unit
0-100	$3.99
101 - 300	$2.99
301 units or more	$1.99

Flowchart



Pseudocode

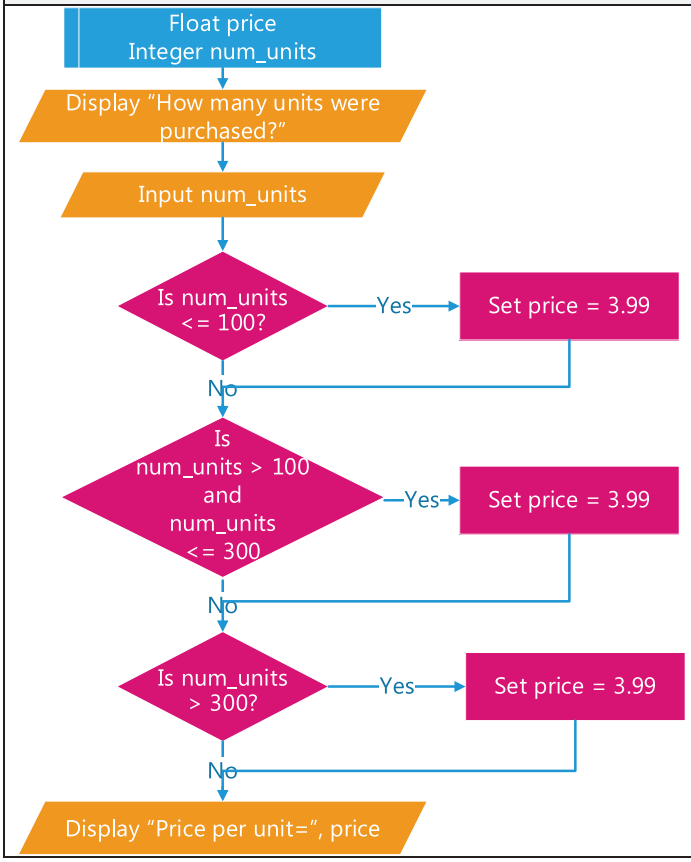
```
Declare price as Float
Declare num_units as Integer

Display "How many units were purchased?"
Input num_units

If num_units <= 100 Then
      Set price = 3.99
Else
      If num_units <= 300 Then
            Set price = 2.99
      Else
            Set price = 1.99
      End If
End If

Display "Price per unit =", price
```

NESTED DECISION STRUCTURE PSEUDOCODE

Problem

Number of units ordered	Price per unit
0-100	$3.99
101 - 300	$2.99
301 units or more	$1.99

Flowchart



Pseudocode

```
Declare price as Float
Declare num_units as Integer

Display "How many units were purchased?"
Input num_units

If num_units <= 100 Then
     Set price = 3.99
End If

If num_units > 100 AND num_units <= 300 Then
     Set price = 2.99
End If

If num_units > 100 Then
     Set price = 1.99
End If

Display "Price per unit =", price
```

Name _____

Create Flowcharts to Depict Decisions

Purpose

Understand decision structures and flowcharts

Objective

Create decision structures in a flowchart.

Materials

- Device with Internet
- Lab sheet
- Pen or pencil

Procedure

1. Read the following instructions and create a flowchart by hand or by using Visio, or going to https://online.visual-paradigm.com/diagrams/solutions/free-flowchart-maker-online/
2. Consider the table that shows letter grades for various points earned in a test.

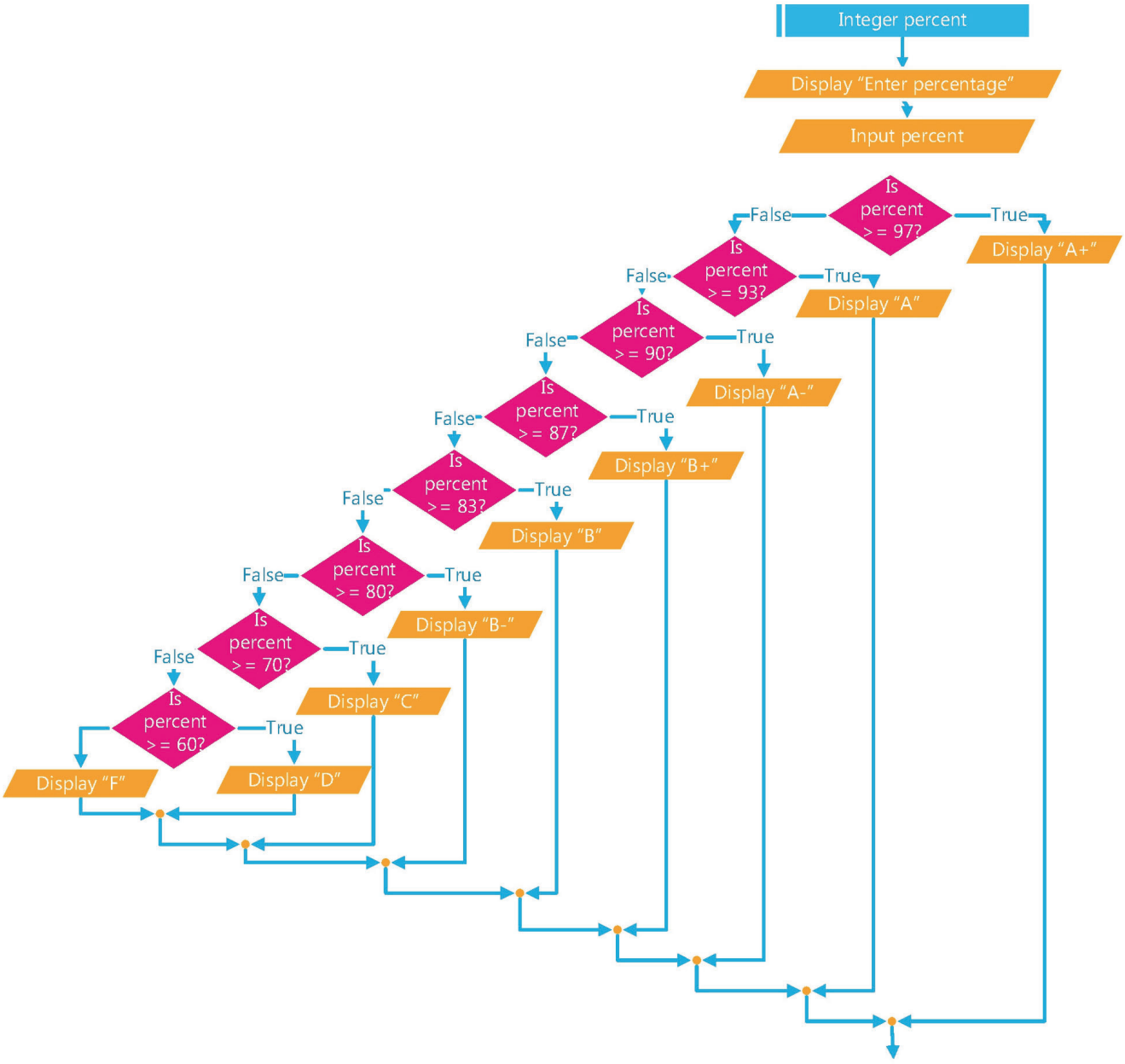
Percent	Letter Grade
97-100	A+
93-96	A
90-92	A-
87-89	B+
83-86	B
80-82	B-
70-79	C
60-69	D
Below 60	F

3. Draw a flowchart that asks user for a percent. The logic should display the letter grade depending upon the percent entered by the user.

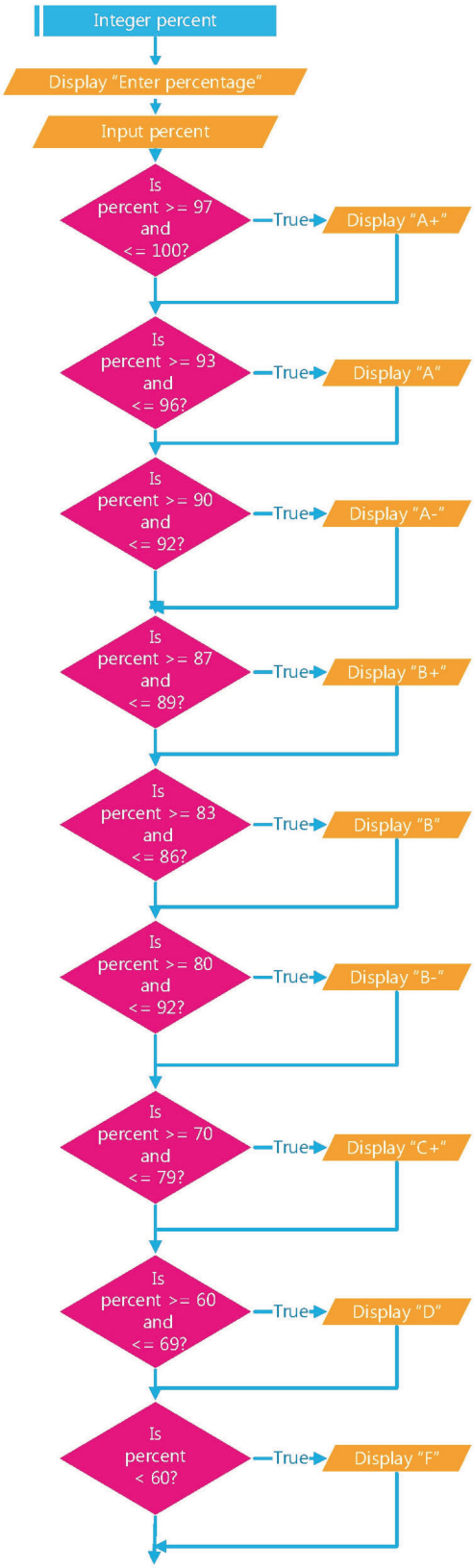
Create Flowcharts to Depict Decisions

There are multiple possible solutions. Two alternatives are shown below:

Alternative 1



Alternative 2



Name _____

Code Decision Structure in Pseudocode

Purpose

Understand the process of creating pseudocode

Objective

Create pseudocode for a decision.

Materials

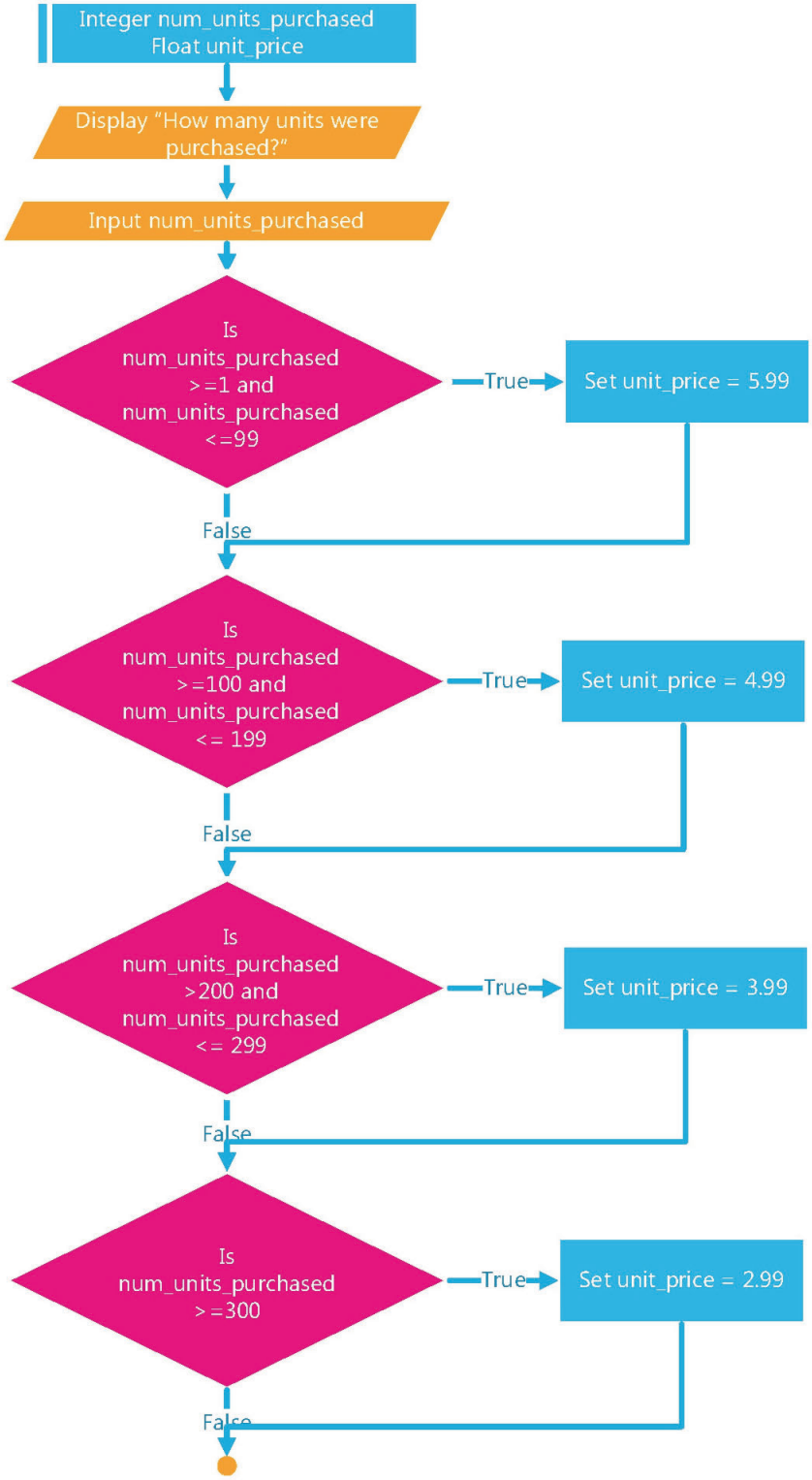
- Device with Internet
- Lab sheet
- Pen or pencil

Procedure

1. Consider the following pricing table that bases unit price of a product based upon the number of units purchased.

Number of Units Purchased	Price per Unit
1-99	5.99
100-199	4.99
200-299	3.99
More than 300	2.99

2. Consider the following flowchart that depicts this logic to determine unit price based upon number of units purchased.



3. Write the pseudocode that corresponds to the logic shown in the flowchart.

Code Decision Structure in Pseudocode

Pseudocode for the problem is as follows:

```
Declare num_units_purchased as Integer
Declare unit_price as Float
Display “How many units were purchased?”
Input num_units_purchased

If num_units_purchased >= 1 and num_units_purchased < = 99 Then
  Set unit_price = 5.99
End If

If num_units_purchased >= 100 and num_units_purchased < = 199 Then
  Set unit_price = 4.99
End If

If num_units_purchased >= 200 and num_units_purchased < = 299 Then
  Set unit_price = 3.99
End If

If num_units_purchased >= 300 Then
  Set unit_price = 2.99
End If
```