**Chapter Four Algorithm Workbench:**

1. Design an If-Then statement (or flowchart with a single alternative decision structure) that assigns 20 to the variable y and assigns 40 to the variable z if the variable x is greater than 100.

If (x > 100)

{

Set y = 20

Set z = 40

}

1. Design an If-Then statement (or flowchart with a single alternative decision structure) that assigns 0 to the variable b and assigns 1 to the variable c if the variable a is less than 10.

If (a < 10)

{

Set b = 0

Set c = 1

}

1. Design an If-Then statement (or flowchart with a single alternative decision structure) that assigns 0 to the variable b if the variable is less than 10. Otherwise it should assign 99 to the variable b.

If (a < 10)  
{

Set b = 0 Else

Set b = 99

}

1. The following pseudocode contains several nested If-Then-Else statements. Unfortunately, it is written without proper alignment and indentation. Re write the code and use the proper conventions of alignment and indentation.

If (score < 60)

Display (“Your grade is an F.”);

Else

If (score < 70)

Display “Your grade is an D.”);

Else

If (score < 80)

Display (“Your grade is an C.”);

Else

If (score < 90)

Display (“Your grade is an B.”);

Else

Display (“Your grade is an A.”);

If (score < 60)

Display (“Your grade is an F.”);

Else If score < 70

Display (“Your grade is an D.”);

Else If (score < 80

Display (“Your grade is an C.”);

Else If score < 90

Display (“Your grade is an B.”);

Else

Display (“Your grade is an A.”);

5. Design nested decision structures that perform the following: If amount1 is greater than

10 and amount2 is less than 100, display the greater of amount1 and amount2.

If amount1 > 10 Then

If amount2 < 100 Then

Display (”amount1”);

Else

If (amount2 > amount1)

Display (“amount2”);

Else

Display (“amount1, amount2 are equal”);

6. Re write the following If-Then-Else If statement as a select case statement.

If (selection == 1)

Display (“You selected A.”);

Else

If (selection == 2)

Display (“You selected 2.”);

Else

If (selection == 3)

(Display “You selected 3.”);

Else

If (selection == 4)

Display (“You selected 4.”);

Else

Display (“Not good with numbers, eh?”);

Select selection:

Case 1:

Display (“You selected A.”);

Case 2:

Display (“You selected 2.”);

Case 3:

Display (“You selected 3.”);

Case 4:

Display (“You selected 4.”);

Default:

Display (“Not good with numbers, eh?”);

7. Design an If-Then-Else statement (or a flowchart with a dual alternative decision structure)

that displays “Speed is normal” if the speed variable is within the range of 24 to 56, If speed

holds a value outside this range, display “Speed is abnormal.”

If (speed >= 24 AND speed <= 56);

Display ("Speed is normal");

Else

Display ("Speed is abnormal.");

8. Design an If-Then-Else statement (or a flowchart with a dual alternative decision structure)

that determines whether the points variable is outside the range of 9 to 51. If the variable

holds a value outside the range it should display “Invalid points.” Otherwise, it should display “Valid points”

If points (< 9 AND points > 51)   
 Display (“Invalid points.”);  
 Else  
 Display (“Valid points.”);

9. Design a case structure that tests the month variable and does the following:

* If the month is set to 1, it displays “January has 31 days.”
* If the month is set to 2, it displays “February has 28 days.”
* If the month is set to 3, It displays “March has 31 days.”
* If the month variable is set to anything else, it displays “Invalid selection.”

Select month  
Case 1:  
Display (“January has 31 days.”);  
Case 2:  
Display (“February has 28 days.”);  
Case 3:  
Display (“March has 31 days.”);  
Default:  
Display (“Invalid selection.”);

10. Write an If-Then statement that sets the variable hours to 10 when the flag variable minimum is set.

If (minimum == minimum)

{  
 Set hours = 10

}

**Chapter Five Algorithm Workbench:**

1. Design a while loop that lets the user enter a number. The number should be multiplied by 10, and the result stored in a variable named product. The loop should iterate as long as product contains a value less than 100.

Module main()  
 Declare Integer product = 0  
 Declare Integer num = 0

While (product < 100)

Display (“Enter a number:“);

product = (num \* 1);

Display (“Product”, product);

1. Design a do-while loop that asks the user to enter two numbers. The numbers should be added and the sum displayed. The loop should ask the user whether he or she wishes to perform the operation again. If so, the loop should repeat; otherwise is should terminate.

Module main()

Declare Integer num1 = 0

Declare Integer num2 = 0

Declare Integer sum = 0

Declare String response

Do

Display ("Enter first number: ");

Input num1

Display ("Enter second number);

Input num2

sum = num1 + num2;

Display ("The sum is: " , sum);

Display ("Enter Y to continue: ");

Input response

While response == "Y"

1. Design a for loop that displays the following set of numbers: 0,10,20,30,40,50…….1000

Module main()

Declare Integer counter = 0

For(counter = 0; counter <=1000; counter++)

Display (“”, counter);

}

1. Design a loop that asks the user to enter a number. The loop should iterate 10 times and keep a running total of the numbers entered.

Module main()

Declare Integer counter = 0

Declare Integer num = 0

Declare Integer total = 0

For(counter = 0; counter <= 10; counter++)

{

Display ("Enter #: ");

Input num total = total + num

}

Display ("Total: " , total);

1. Design a for loop that calculates the total of the following series of numbers: 1/30 + 2/29 + 3/28…...30/1

Module main()

Declare Integer num = 1

Declare Integer den = 30

Declare Integer total = 0

For (num = 1; counter <= 30; counter+)

{

total = total + num/den

den = den -1

Display ("Total: " , total);

}

1. Design a nested loop that displays 10 rows of # characters. There should be 15 # characters in each row.

Declare Integer rowCount = 0

Declare Integer itemCount = 0

Declare String line

For (rowCount = 0; rowCount <=10; rowCount++)

{

Display (rowCount);

}

For itemCount = 0 To 15

{

Display (itemCount);

}

1. Convert the while loop in the following code to a do-while loop:

Declare Integer x = 1

While (x > 0)

{

Display (“Enter a number”)

Input x

}

Declare Integer x = 1

Do

Display (“Enter a number”);

Input x

While (x > 0);

8. Convert the do-while loop in the following code to a while loop:

Declare String sure

Do

Display (“Are you sure you want to quit?”);

Input sure

While (sure ! = “Y” AND sure ! = “y”);

{

Declare String sure  
 Display (“Are you sure you want to quit?”);  
 Input sure

}

While (sure != “Y” && sure != “y”)  
 {

Display (“Are you sure you want to quit?”);  
 Input sure  
 }

9. Convert the while loop in the following code to a for loop:

Declare Integer count = 0

While (count < 50)

Display (“The count is”, count);

Set count = count + 1;

Declare Integer count = 0

For (count = 0 To 50)

Display ("The count is ", count);

10. Convert the for loop in the following code to a while loop:

Declare Integer count

For (count = 1 to 50)

{

Display (count);

}

Declare Integer count = 0

While (count < 50 )

Display (count);

Set count = count + 1