

# CS 162: Computer Science II

## Algorithm Design Document

Make a copy before you begin (File -> Make a copy). Add the Assignment # above and complete the sections below BEFORE you begin to code. The sections will expand as you type. When you are finished, download this document as a PDF (File -> Download -> PDF) and submit to D2L.

This document contains an interactive checklist. To mark an item as complete, click on the box (the entire list will be highlighted), then right click (the clicked box will only be highlighted), and choose the checkmark.

Planning your program before you start coding is part of the development process. In this document you will:

- ☒ Paste a screenshot of your zyBooks Challenge and Participation %
- ☒ Paste a screenshot of your assigned zyLabs completion
- ☒ Write a detailed description of your program, at least two complete sentences
- ☒ If applicable, design a sample run with test input and output
- ☒ Identify the program inputs and their data types
- ☒ Identify the program outputs and their data types
- ☒ Identify any calculations or formulas needed
- ☒ Write the algorithmic steps as pseudocode

### 1. zyBooks

Add your zyBooks screenshots for the % and assigned zyLabs completions below. Required percentages: all **assigned** zyLabs, Challenge Activity with at least 70%, and Participation Activity with at least 80%.

#### Challenge and Participation % screenshot:

22. CS 162 Linked Lists Part I

 100%  100%  100% ▼

#### Assigned zyLabs completion screenshot:

22. CS 162 Linked Lists Part I

 100%  100%  100% ▼

## 2. Program Description

---

In the box below, describe the purpose of the program. You must include a detailed description with at least two complete sentences.

### Program description:

Program designed to keep track of activities. Users can view the list of activities or search for activities based on name, type, or location. They can also add and remove activities from the list. The app will load the list of activities from an external .txt file and modify the .txt file based on user input.

## 3. Sample Run

---

If you are designing your own program, you will start with a sample run. **Imagine** a user is running your program - what will they see? What inputs do you expect, and what will be the outputs from the given inputs? Choose test data you will use to test your program. Calculate and show the expected outputs. Use the sample run to test your program.

**Do not simply copy the sample run from the assignment instructions!**

### Sample run:

```
Welcome!
This program will help you manage your activities.

Pick an option from below:

(a)Add a new activity
(b)List activities by name
(c)List activities by location
(d)List activities by Type
(e)Remove an activity
(f)Search by activity name
(q)Quit
```

**b**

1. Catan;Epic Gaming;Easy;4;Games
2. Oil Painting;Fun Studios;Tricky;6;Arts
3. Pottery;Fun Studios;Easy;7;Arts
4. Skiing;Mt Hood Meadows;Difficult;6;Athletics

5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

6. Wine Making;Umpqua Valley;Complicated;9;Food

Pick an option from below:

- (a)Add a new activity
- (b)List activities by name
- (c)List activities by location
- (d)List activities by Type
- (e)Remove an activity
- (f)Search by activity name
- (q)Quit

**p**

Invalid option!! Please try again!

Pick an option from below:

- (a)Add a new activity
- (b)List activities by name
- (c)List activities by location
- (d)List activities by Type
- (e)Remove an activity
- (f)Search by activity name
- (q)Quit

**d**

Enter Type number(0-Athletics, 1-Food, 2-Arts, 3-Games, and 4-Others): **0**

Skiing;Mt Hood Meadows;Difficult;6;Athletics

Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

Pick an option from below:

- (a)Add a new activity
- (b)List activities by name
- (c)List activities by location
- (d)List activities by Type
- (e)Remove an activity
- (f)Search by activity name
- (q)Quit

C

Enter location name: Hood Meadows

1. Skiing;Mt Hood Meadows;Difficult;6;Athletics
2. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

Pick an option from below:

- (a)Add a new activity
- (b)List activities by name
- (c)List activities by location
- (d)List activities by Type
- (e)Remove an activity
- (f)Search by activity name
- (q)Quit

e

1. Catan;Epic Gaming;Easy;4;Games
2. Oil Painting;Fun Studios;Tricky;6;Arts
3. Pottery;Fun Studios;Easy;7;Arts
4. Skiing;Mt Hood Meadows;Difficult;6;Athletics
5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics
6. Wine Making;Umpqua Valley;Complicated;9;Food

Pick the index to remove: 4

Activity removed!

1. Catan;Epic Gaming;Easy;4;Games
2. Oil Painting;Fun Studios;Tricky;6;Arts
3. Pottery;Fun Studios;Easy;7;Arts
4. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics
5. Wine Making;Umpqua Valley;Complicated;9;Food

Pick an option from below:

- (a)Add a new activity

(b)List activities by name  
(c)List activities by location  
(d)List activities by Type  
(e)Remove an activity  
(f)Search by activity name  
(q)Quit

**a**

Enter the activity name (50 characters or less): **Rowing**  
Enter the activity location (50 characters or less): **Oaks Amusement Park**  
Enter the activity level : **Tricky**  
Enter the activity rating : **aaa**  
Invalid rating! Please enter a valid rating!  
Enter the activity rating : **8**  
Enter Type number(0-Athletics, 1-Food, 2-Arts, 3-Games, and 4-Others): **0**

Activity added!

1. Catan;Epic Gaming;Easy;4;Games
2. Oil Painting;Fun Studios;Tricky;6;Arts
3. Pottery;Fun Studios;Easy;7;Arts
4. Rowing;Oaks Amusement Park;Tricky;8;Athletics
5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics
6. Wine Making;Umpqua Valley;Complicated;9;Food

Pick an option from below:

(a)Add a new activity  
(b)List activities by name  
(c)List activities by location  
(d)List activities by Type  
(e)Remove an activity  
(f)Search by activity name  
(q)Quit

**f**

Enter the activity name (50 characters or less): **Snowboarding**

Activity found!

5. Snowboarding;Mt Hood Meadows;Not for Faint of heart;8;Athletics

Pick an option from below:

- (a)Add a new activity
- (b)List activities by name
- (c)List activities by location
- (d)List activities by Type
- (e)Remove an activity
- (f)Search by activity name
- (q)Quit

**f**

Enter the activity name (50 characters or less): **Skiing**  
Activity not found!!

Pick an option from below:

- (a)Add a new activity
- (b)List activities by name
- (c)List activities by location
- (d)List activities by Type
- (e)Remove an activity
- (f)Search by activity name
- (q)Quit

**q**

Activities written to file! Thank you for using my program!!

## 4. Algorithmic Design

Before you begin coding, **you must first plan out the logic** and think about what data you will use to test your program for correctness. All programmers plan before coding - this saves a lot of time and frustration! Use the steps below to identify the inputs and outputs, calculations, and steps needed to solve the problem.

Use the pseudocode syntax shown in the document, supplemented with English phrases if necessary. **Do not include any implementation details (e.g. source code file names, class or struct definitions, or language syntax).** Do not include any C++ specific syntax or data types.

**Algorithmic design:**

- Identify and list all of the user input variables and their data types. Include a variable name, data type, and description. Data types include string, integer, floating point, (single) character, and boolean. Data structures should be referenced by name, e.g. “array of integer” or “array of string”.

- Char option
- String name, location, level
- Int rating, type

- Identify and list all of the user output variables and their data types. Include a variable name, data type, and description. Data types include string, integer, floating point, (single) character, and boolean. Data structures should be referenced by name, e.g. “array of integer” or “array of string”.

- String name, location, level
- Int rating, type

- What calculations do you need to do to transform inputs into outputs? List all formulas needed, if applicable. If there are no calculations needed, state there are no calculations for this algorithm. Formulae should reference the variable names from step a and step b as applicable.

- Addition/subtraction to add/remove activities

- Design the logic of your program using pseudocode. Here is where you would use conditionals, loops or functions (if applicable) and list the steps in transforming inputs into outputs. Walk through your logic steps with the test data from the assignment document or the sample run above.

**Use the syntax shown at the bottom of this document. Do not include any implementation details (e.g. file names) or C++ specific syntax.**

#### ACTIVITY.H

1. DECLARE const MAXCHAR = 51
2. CLASS Activity
  - a. PRIVATE
    - i. activityName as a char array pointer
    - ii. activityLocation as a char array pointer
    - iii. activityLevel as as a char array pointer
    - iv. activityType as a char array pointer
    - v. activityRating as an integer

b. PUBLIC

i. Constructors/Destructor

1. Activity - default constructor
2. Activity - constructor w/ params
3. Activity - copy constructor
4. Activity - destructor

ii. Mutators

1. setActivityName - assign char array pointer to activityName
2. setActivityLocation - assign char array pointer to activityName
3. setActivityLevel - assign char array pointer to activityName
4. setActivityRating - assign char array pointer to activityName
5. setActivityType - assign char array pointer to activityName

iii. Accessors

1. getActivityName - assign char array pointer to activityName
2. getActivityLocation - assign char array pointer to activityName
3. getActivityLevel - assign char array pointer to activityName
4. getActivityRating - assign char array pointer to activityName
5. getActivityType - assign char array pointer to activityName

iv. printActivity - Display single activity to console

v. writeActivity - Write single activity to outFile

vi. &operator - assignment operator overloading

## ACTIVITYLIST.H

3. CLASS ActivityList

a. PRIVATE

i. STRUCT Node

1. Activity data
2. Node next

ii. Node head

iii. Node tail

iv. Int size

b. PUBLIC

i. Constructors

1. ActivityList - default constructor
2. ActivityList - constructor w/ params

ii. Destructors

1. ~ActivityList

iii. Database Functions

1. addActivity
2. searchByName - display activities of a certain name
3. searchByType - display activities of a certain type
4. searchByLocation - display activities of a certain location
5. showList - display list



6. removeActivity - remove activity from list
7. writeData - Write data to outfile

## TOOLS.H

1. Function Prototypes
  - a. readInt - determines input is a positive integer
  - b. readString - determines input does not contain a semicolon
  - c. isLessThan - helper function utilized by loadData for sorting
  - d. displayMenu - displays options to user
  - e. readOption - reads user input
  - f. exeCmd - calls function associated with user input
  - g. welcomeMessage - displays welcome message
  - h. goodbyeMessage - displays goodbye message
  - i. addActivity - displays prompts and calls setter methods
  - j. retrieveActivityType - maps type number to type string

## ACTIVITY.CPP

4. DEFAULT CONSTRUCTOR Activity
  - a. SET activityName = "no name"
  - b. SET activityLocation = "no location"
  - c. SET activityLevel = "no level"
  - d. SET activityRating = "0"
  - e. SET activityType = "no type"

END CONSTRUCTOR

5. CONSTRUCTOR Activity
  - a. SET activityName = new char[MAXCHAR]
  - b. SET activityLocation = new char[MAXCHAR]
  - c. SET activityLevel = new char[MAXCHAR]
  - d. SET activityRating = new char[MAXCHAR]
  - e. SET activityType = new char[MAXCHAR]

END CONSTRUCTOR

6. COPY CONSTRUCTOR Activity
  - a. SET activityName = new char[activityName length + 1]
  - b. SET activityLocation = new char[activityLocation length + 1]
  - c. SET activityLevel = new char[activityLevel length + 1]
  - d. SET activityRating = new char[activityType length + 1]
  - e. This = activity

END CONSTRUCTOR

7. DESTRUCTOR Activity
  - a. IF(activityName)

- i. Delete activityName
  - ii. activityName = NULL
- b. IF(activityLocation)
  - i. Delete activityLocation
  - ii. activityName = NULL
- c. IF(activityLevel)
  - i. Delete activityLevel
  - ii. activityName = NULL
- d. IF(activityType)
  - i. Delete activityType
  - ii. activityName = NULL

#### MUTATOR FUNCTIONS

- 8. FUNCTION setActivityName
  - a. IF(activityName)
    - i. Delete activityName
    - ii. activityName = NULL
  - b. SET activityName = new char[activityName length + 1]
  - c. SET activityName = newActivityName

END FUNCTION setActivityName

- 9. FUNCTION setActivityLocation
  - a. IF(activityLocation)
    - i. Delete activityLocation
    - ii. activityLocation = NULL
  - b. SET activityLocation = new char[activityLocation length + 1]
  - c. SET activityLocation = newActivityLocation

END FUNCTION setActivityLocation

- 10. FUNCTION setActivityLevel
  - a. IF(activityLevel)
    - i. Delete activityLevel
    - ii. activityLevel = NULL
  - b. SET activityLevel = new char[activityLevel length + 1]
  - c. SET activityLevel = newActivityLevel

END FUNCTION setActivityLevel

- 11. FUNCTION setActivityRating
  - a. SET activityRating = newActivityRating

END FUNCTION setActivityRating

12. FUNCTION setActivityType
  - a. IF(activityType)
    - i. Delete activityType
    - ii. activityType = NULL
  - b. SET activityType = new char[activityType length + 1]
  - c. SET activityType = newActivityType

END FUNCTION setActivityType

#### ACCESSOR FUNCTIONS

13. FUNCTION getActivityName
  - a. returnActivityName = activityName

END FUNCTION getActivityName

14. FUNCTION getActivityLocation
  - a. returnActivityLocation = activityLocation

END FUNCTION getActivityLocation

15. FUNCTION getActivityLevel
  - a. returnActivityName = activityLevel

END FUNCTION getActivityLevel

16. FUNCTION getActivityRating
  - a. returnActivityRating = activityRating

END FUNCTION getActivityRating

17. FUNCTION getActivityType
  - a. returnActivityType = activityType

END FUNCTION getActivityType

18. FUNCTION printActivity
  - a. DISPLAY activityName
  - b. DISPLAY activityLocation
  - c. DISPLAY activityLevel
  - d. DISPLAY activityRating
  - e. DISPLAY activityType

END FUNCTION printActivity

19. FUNCTION ASSIGNMENT OPERATOR Activity
  - a. IF(this = anActivity)
    - i. RETURN this

- b. ELSE
  - i. CALL setActivityName
  - ii. CALL setActivityLocation
  - iii. CALL setActivityLevel
  - iv. CALL setActivityRating
  - v. CALL setActivityType
- c. RETURN this

FUNCTION ASSIGNMENT OPERATOR Activity

20. FUNCTION writeActivity

- a. OUTPUT activityName
- b. OUTPUT activityLocation
- c. OUTPUT activityLevel
- d. OUTPUT activityRating
- e. OUTPUT activityType

END FUNCTION writeActivity

### **ACTIVITYLIST.CPP**

21. DEFAULT CONSTRUCTOR ActivityList

- a. SET head = NULL
- b. SET tail = NULL
- c. SET size = 0

END CONSTRUCTOR

22. CONSTRUCTOR ActivityList

- a. SET head = NULL
- b. SET tail = NULL
- c. SET size = 0
- d. DECLARE inFile
- e. DECLARE activity
- f. DECLARE tempString
- g. DECLARE tempNumber
- h. IF(!inFile)
  - i. DISPLAY error message
  - ii. EXIT
- i. WHILE(inFile)
  - i. CALL setActivityName
  - ii. CALL setActivityLocation
  - iii. CALL setActivityLevel
  - iv. CALL set ActivityRating
  - v. CALL setActivityType

- vi. CALL addActivity
- j. END WHILE LOOP

END CONSTRUCTOR

#### 23. DESTRUCTOR ActivityList

- a. DECLARE Node curr = head
- b. WHILE(!curr)
  - i. SET head = next
  - ii. DELETE curr
  - iii. SET curr = head
- c. SET tail = NULL

END DESTRUCTOR

#### 24. FUNCTION addActivity

- a. DECLARE Node \*newNode = NULL, \*curr = NULL, \*prev = NULL
- b. DECLARE char str1[MAXCHAR], str2[MAXCHAR]
- c. SET newNode = new Node
- d. SET newNode->data = anActivity
- e. SET newNode->next = NULL
- f. IF (!head)
  - i. SET head = newNode
  - ii. SET tail = newNode
- g. ELSE
  - i. SET curr = head
  - ii. SET curr->data.getActivityName(str1)
  - iii. SET newNode->data.getActivityName(str2)
- h. WHILE (curr && isLessThan(str1, str2))
  - i. SET prev = curr
  - ii. SET curr = curr->next
  - iii. IF (curr)
    - 1. SET curr->data.getActivityName(str1)
- i. IF (!curr)
  - i. SET tail->next = newNode
  - ii. SET tail = newNode
- j. ELSE IF (prev)
  - i. SET newNode->next = curr
  - ii. SET prev->next = newNode
- k. ELSE
  - i. SET newNode->next = curr
  - ii. SET head = newNode
- l. size++

END FUNCTION addActivity

25. FUNCTION searchByName

- a. DECLARE searchName
- b. DECLARE tempName
- c. DECLARE count = 0
- d. CALL readString(prompt, searchName)
- e. FOR (Node curr = head, curr, curr = next)
  - i. CALL getActivityName(tempName)
  - ii. DISPLAY "Activity Found!"
  - iii. IF(tempName = searchName)
    - 1. CALL printActivity()
- f. END FOR LOOP
- g. IF(count == size)
  - i. DISPLAY "Activity Not Found!"

END FUNCTION

26. FUNCTION searchByLocation

- a. DECLARE searchLocation
- b. DECLARE tempLocation
- c. DECLARE count = 0
- d. CALL readString(prompt, searchLocation)
- e. FOR (Node curr = head, curr, curr = next)
  - i. CALL getActivityLocation(tempLocation)
  - ii. DISPLAY "Activity Found!"
  - iii. IF(tempLocation = searchLocation)
    - 1. CALL printActivity()
- f. END FOR LOOP
- g. IF(count == size)
  - i. DISPLAY "Activity Not Found!"

END FUNCTION searchByLocation

27. FUNCTION searchByType

- a. DECLARE searchType
- b. DECLARE tempType
- c. DECLARE count = 0
- d. CALL readInt(prompt, searchType)
- e. FOR (Node curr = head, curr, curr = next)
  - i. CALL getActivityType()
  - ii. DISPLAY "Activity Found!"
  - iii. IF(tempType = searchType)
    - 1. CALL printActivity()
- f. END FOR LOOP

- g. IF(count == size)
  - i. DISPLAY "Activity Not Found!"

END FUNCTION searchByType

28. FUNCTION writeData

- a. DECLARE count = 0
- b. DECLARE outFile
- c. OPEN outFile
- d. DECLARE Node curr
- e. FOR curr = head, curr, curr = next
  - i. CALL writeActivity
  - ii. SET count++
- f. END FOR LOOP

END FUNCTION writeDate

29. FUNCTION showList

- a. DECLARE count
- b. FOR curr = head, curr, curr = next
  - i. DISPLAY count
  - ii. CALL printActivity()
  - iii. SET count++
- c. END FOR LOOP

END showList

30. FUNCTION removeActivity

- a. // NOT IMPLEMENTED YET

END removeActivity

**TOOLS.CPP**

31. FUNCTION readInt

- a. DISPLAY prompt
- b. INPUT temp
- c. WHILE (!cin || temp < 0)
  - i. DISPLAY "Invalid number! Please try again!!!"
  - ii. DISPLAY prompt
  - iii. INPUT temp
- d. RETURN temp

END FUNCTION readInt

32. FUNCTION readString

- a. DISPLAY prompt;

```

    b. INPUT tempString
    c. FOR i = 0 to tempString.length
        i. IF(temp[i] == ';')
            1. DISPLAY "Invalid string! Please remove all commas from entry!"
            2. CALL readString
    d. END FOR LOOP

END FUNCTION readString

33. FUNCTION isLessThan
    a. IF (activityName1 = activityName2) {
        i. RETURN true
    b. ELSE
        i. RETURN false

END FUNCTION isLessThan

34. FUNCTION displayMenu
    a. DISPLAY "Pick an option from below:"
    b. DISPLAY "(a)Add a new activity"
    c. DISPLAY "(b)List activities by name"
    d. DISPLAY "(c)List activities by location"
    e. DISPLAY "(d)List activities by Type"
    f. DISPLAY "(e)Remove an activity"
    g. DISPLAY "(f)Search by activity name"
    h. DISPLAY "(q)Quit"

END FUNCTION displayMenu

35. FUNCTION readOption() {
    a. DECLARE input;
    b. INPUT input;
    c. RETURN input;

END FUNCTION readOption()

36. FUNCTION executeCommand
    a. SELECT tempNumber
        i. CASE a
            1. CALL addActivity()
            2. CALL activityList.addActivity()
            3. CALL activityList.showList()
            4. BREAK
        ii. CASE b
            1. CALL showList()
            2. BREAK

```



- iii. CASE c
  - 1. CALL searchByLocation()
  - 2. BREAK
- iv. CASE d
  - 1. CALL searchByType()
  - 2. BREAK
- v. CASE e
  - 1. CALL showList()
  - 2. CALL removeActivity()
  - 3. CALL showList()
  - 4. BREAK
- vi. CASE f
  - 1. CALL searchByName()
  - 2. BREAK
- vii. CASE q
  - 1. BREAK
- viii. DEFAULT
  - 1. DISPLAY "Invalid Option"

END FUNCTION executeCommand

### 37. FUNCTION writeData

- a. OPEN outFile
- b. FOR int i =0 until size
  - i. INPUT activityList[i].name;
  - ii. INPUT activityList[i].location;
  - iii. INPUT activityList[i].level;
  - iv. INPUT activityList[i].rating;
  - v. INPUT activityList[i].type;
- c. END FOR LOOP
- d. CLOSE outFile
- e. DISPLAY "Activities written to file! Thank you for using my program!!"

END FUNCTION writeData

### 38. FUNCTION welcomeMessage

- a. DISPLAY "Welcome"
- b. DISPLAY "This program will help you manage your activities"

END FUNCTION

### 39. FUNCTION goodbyeMessage

- a. DISPLAY "Activities written to file! Thank you for using my program!!"

END FUNCTION

```

40. FUNCTION retrieveActivityType
    a. SELECT tempNumber
        i. CASE 0
            1. activityListType = "Athletics"
        ii. CASE 1
            1. activityListType = "Food"
        iii. CASE 2
            1. activityListType = "Arts"
        iv. CASE 3
            1. activityListType = "Games"
        v. CASE 4
            1. activityListType = "Others"

```

```

END FUNCTION printActivityType

```

```

41. FUNCTION addActivity
    a. DECLARE letter
    b. DECLARE tempName, tempLocation, tempLevel, tempType
    c. DECLARE tempRating, tempIntType
    d. DISPLAY prompt
    e. INPUT tempName
    f. DISPLAY prompt
    g. INPUT tempLocation
    h. DISPLAY prompt
    i. INPUT tempLevel
    j. DISPLAY prompt
    k. INPUT tempRating
    l. DISPLAY prompt
    m. INPUT tempIntType
    n. CALL retrieveActivityType()
    o. CALL setActivityName()
    p. CALL setActivityLocation()
    q. CALL getActivityLevel()
    r. CALL setActivityRating()
    s. CALL setActivityType()

```

```

END FUNCTION addActivity

```

### **DRIVER.CPP**

```

42. FUNCTION main()
    a. DECLARE fileName
    b. DECLARE option
    c. CALL welcomeMessage()
    d. CALL ActivityList

```

<pre> e. DO     i.  CALL displayMenu()     ii. SET option = readOption()     iii. CALL executeCommand() f. WHILE (option != q) g. CALL writeData() h. CALL goodbyeMessage()  END FUNCTION main() </pre>
---

## 5. Pseudocode Syntax

Think about each step in your algorithm as an action and use the verbs below:

To do this:	Use this verb:	Example:
Create a variable	DECLARE	DECLARE integer num_dogs
Print to the console window	DISPLAY	DISPLAY "Hello!"
Read input from the user into a variable	INPUT	INPUT num_dogs
Update the contents of a variable	SET	SET num_dogs = num_dogs + 1
<b>Conditionals</b>		
Use a single alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> END IF	IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF
Use a dual alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> ELSE <i>statement</i> <i>statement</i> END IF	IF num_dogs > 10 THEN DISPLAY "You have more than 10 dogs!" ELSE DISPLAY "You have ten or fewer dogs!" END IF
Use a switch/case statement	SELECT <i>variable</i> or <i>expression</i> CASE <i>value_1</i> : <i>statement</i> <i>statement</i> CASE <i>value_2</i> : <i>statement</i>	SELECT num_dogs CASE 0: DISPLAY "No dogs!" CASE 1: DISPLAY "One dog.." CASE 2: DISPLAY "Two dogs.." CASE 3: DISPLAY "Three dogs.." DEFAULT: DISPLAY "Lots of dogs!"

	<i>statement</i> CASE <i>value_2</i> : <i>statement</i> <i>statement</i> DEFAULT: <i>statement</i> <i>statement</i> END SELECT	END SELECT
<b>Loops</b>		
Loop while a condition is true - the loop body will execute 0 or more times.	WHILE <i>condition</i> <i>statement</i> <i>statement</i> END WHILE	SET num_dogs = 1 WHILE num_dogs < 10 DISPLAY num_dogs, " dogs!" SET num_dogs = num_dogs + 1 END WHILE
Loop while a condition is true - the loop body will execute 1 or more times.	DO <i>statement</i> <i>statement</i> WHILE <i>condition</i>	SET num_dogs = 1 DO DISPLAY num_dogs, " dogs!" SET num_dogs = num_dogs + 1 WHILE num_dogs < 10
Loop a specific number of times.	FOR <i>counter</i> = <i>start</i> TO <i>end</i> <i>statement</i> <i>statement</i> END FOR	FOR count = 1 TO 10 DISPLAY num_dogs, " dogs!" END FOR
<b>Functions</b>		
Create a function	FUNCTION <i>return_type</i> <i>name (parameters)</i> <i>statement</i> <i>statement</i> END FUNCTION	FUNCTION Integer add(Integer num1, Integer num2) DECLARE Integer sum SET sum = num1 + num2 RETURN sum END FUNCTION
Call a function	CALL <i>function_name</i>	CALL add(2, 3)
Return data from a function	RETURN <i>value</i>	RETURN 2 + 3