CHAPTER 8

Algorithms

Review Questions

1. An algorithm is an ordered set of unambiguous steps that produces a result and terminates in a finite time.
2. Sequence: A sequence of instructions. Decision: Using a boolean value to determine which code to execute. Repetition: Using a boolean condition to determine how many times a sequence of instructions is executed.
3. A flowchart is a pictorial representation of an algorithm.
4. Pseudocode is an English-like representation of an algorithm.
5. Sub-algorithms, subprograms, subroutines, procedures, functions, methods, and modules.
6. A structure chart is a high-level design tool that shows the relationship between different modules in an algorithm. It is used during the design phase of program development.
7. The purpose of the summation algorithm is to find the sum of a list of numbers
8. The purpose of the product algorithm is to find the product of a list of numbers
9. The purpose of a sorting algorithm is to arrange a list of data according to their values.
10. The three types of sorting algorithms are the selection sort, the bubble sort, and the insertion sort.
11. Each of these sorting algorithms divides the list into a sorted and an unsorted list and each takes n - 1 passes (n is the number of elements) to completely sort the data. All of these sorting algorithms use a different method for moving data from the unsorted list to the sorted list.
12. The purpose of a searching algorithm is to find a particular target piece of data among a collection of data.
13. The two major searching algorithms are the sequential search and the binary search. The sequential search is used for searching unordered lists while the binary search is used for searching ordered lists.
14. An algorithm is iterative if it uses a loop construct to perform a repetitive task.

15. An algorithm is recursive if the algorithm executes itself, that is, if the algorithm appears within its own definition.

Multiple-Choice Questions

16. d
17. c
18. c
19. b
20. a
21. c
22. a
23. c
24. b
25. d
26. b
27. b
28. a
29. c
30. d
31. d
32. d
33. c
34. a
35. b
36. b
37. a
38. b
39. c

Exercises

40.

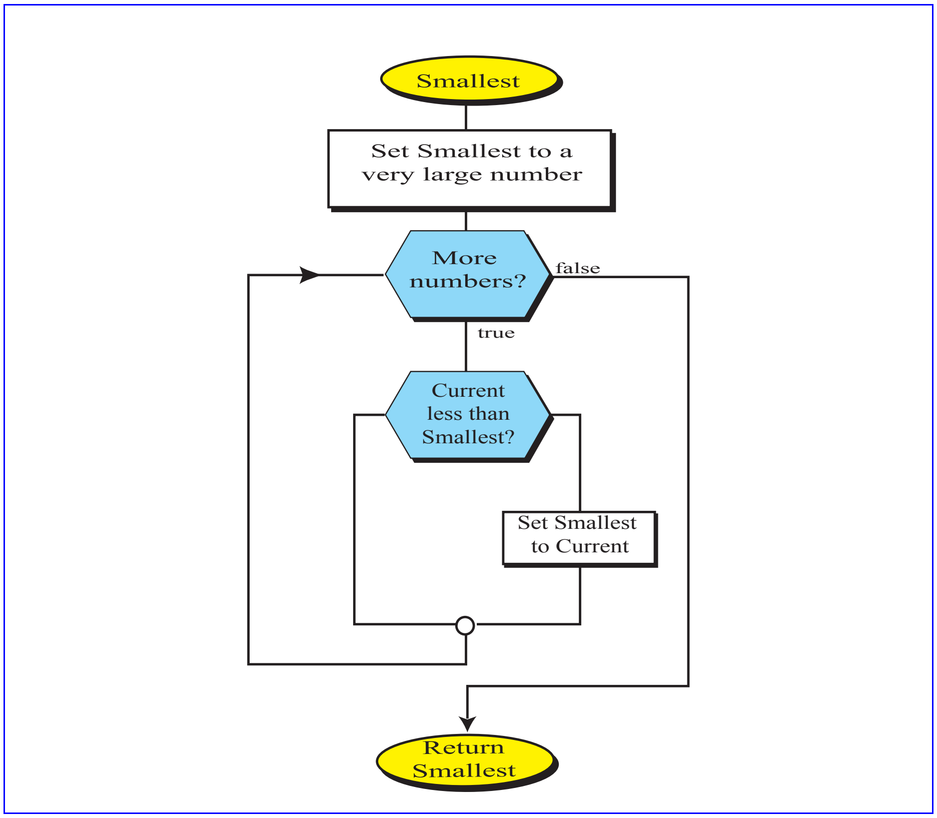
```
Summation
Input: A list of numbers
1. Set Sum to 0
2. While there are more numbers
        2.1 Add the current number to Sum
End loop
3. Return Sum
End
```

41.

```
Product
Input: A list of numbers
1. Set Product to 1
2. While there are more numbers
    2.1 Multiply the current number with Product and store the result in Product
End loop
3. Return Product
End
```

42. See Figure 8.1.

Figure 8.1 *Exercise 42*



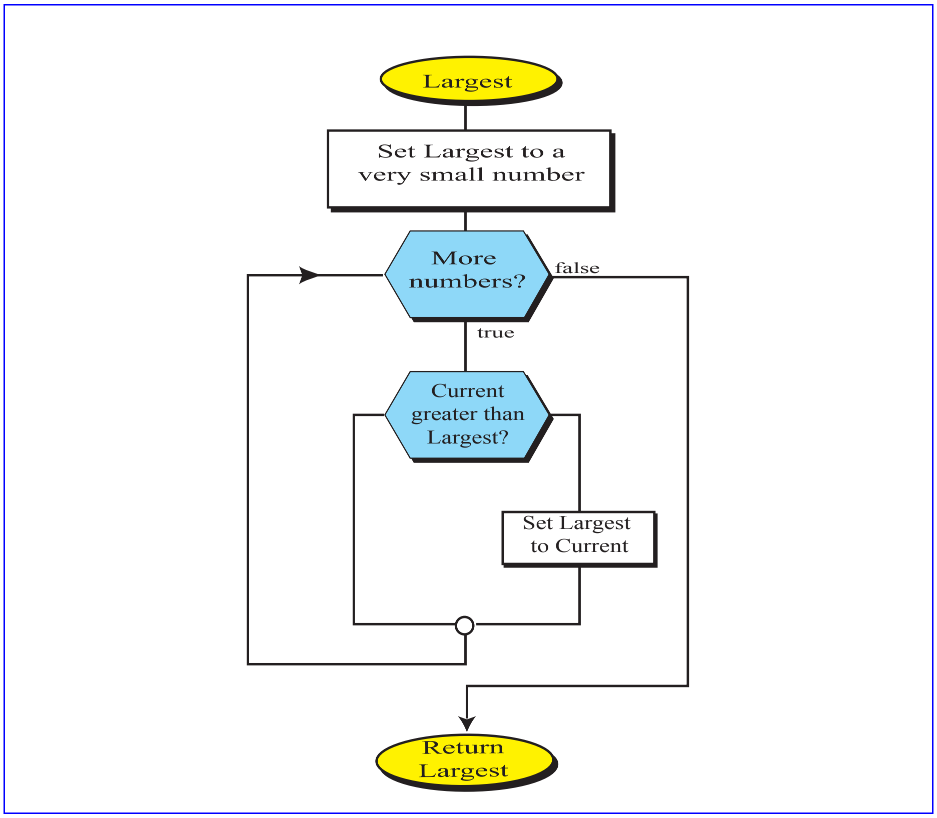
43.

```
Smallest
Input: A list of numbers
1. Set Smallest to the largest possible number
2. While there are more numbers
```

```
        2.1 If the current number is smaller than Smallest
            Then
              2.1.1 Set Smallest to current number
            End if
        End loop
    3. Return Smallest
    End
```

44. See Figure 8.2

Figure 8.2 *Exercise 44*



45.

```
    Largest
    Input: A list of numbers
    1. Set Largest to the smallest possible number
    2. While there are more numbers
        2.1 If the current number is larger than Largest
            Then
            2.1.1 Set Largest to current number
        End if
      End loop
3. Return Largest
End
```

46.

```
Selection Sort
Input: Unsorted list of data
1. Set First to indicate first element of list
2. While First is less than or equal to the second to last element of the list
   2.1 Set Smallest equal to Smallest (First through the end of the list)
   2.2 Swap (Smallest, First)
   2.3 Advance First
  End loop
3. Return list
End
```

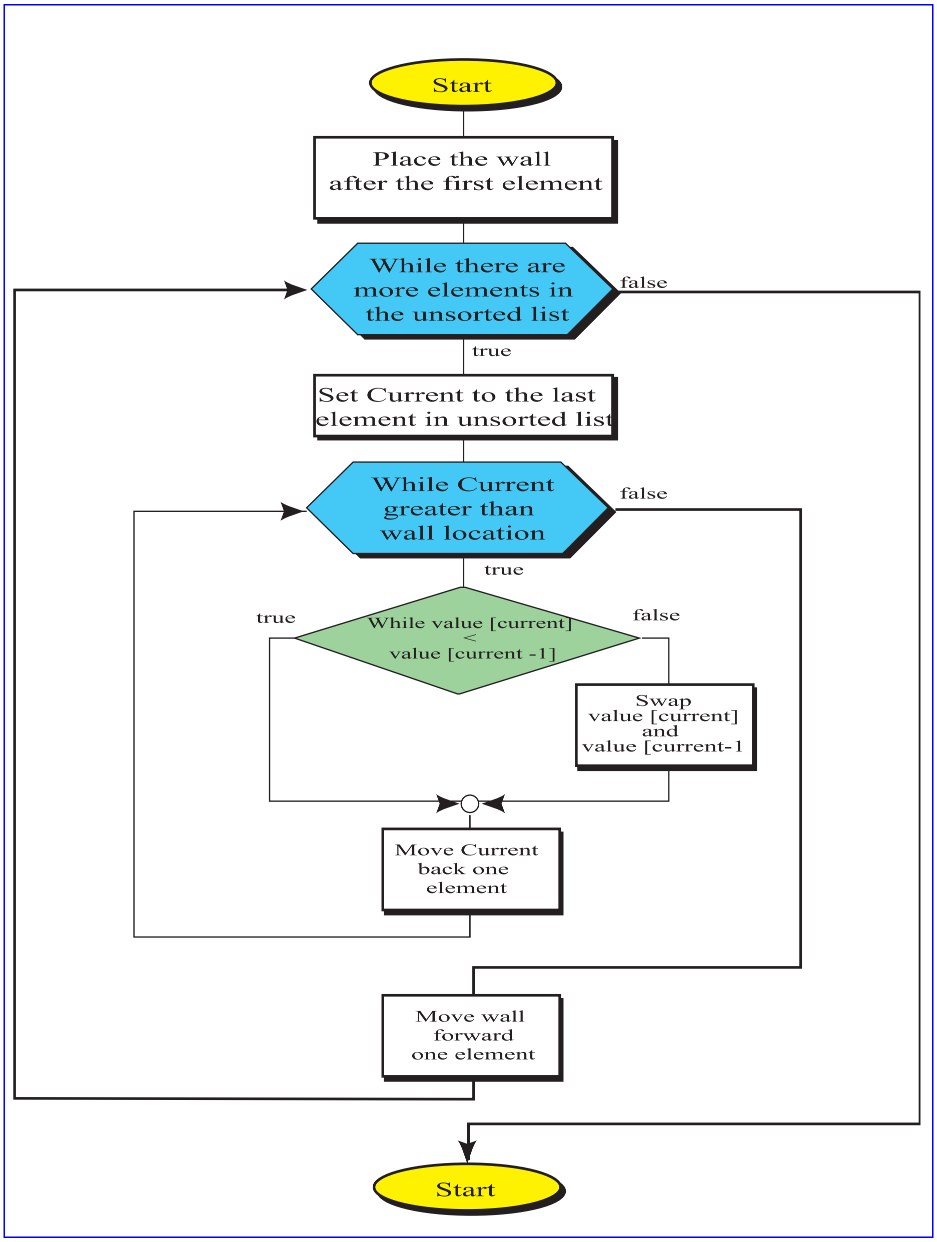
47. See Figure 8.3

```
Bubble Sort
Input: Unsorted list of data
1. Set First to indicate first element of list
2. While First is less than or equal to the second to last element of the list
  2.1 Set Current to indicate last element of list
  2.2 While Current is not equal to First
        2.2.1 If Current is less than element before Current
               2.2.1.1 Swap (Current, element before Current)
           End if
        2.2.2 Set Current to indicate element before Current
      End loop
  2.4 Advance First
 End loop
3. Return list
End
```

48. See Figure 8.4.

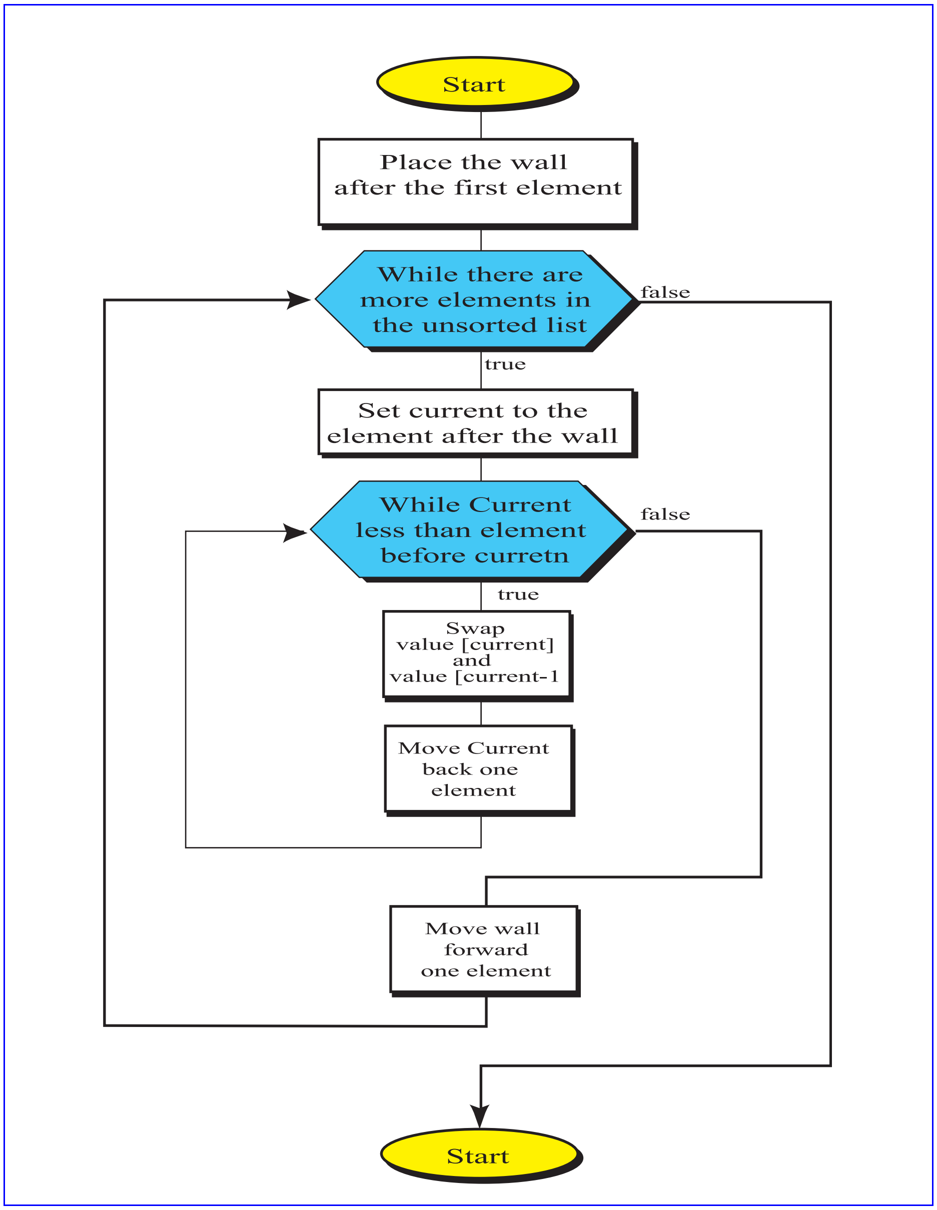
```
Insertion Sort
Input: Unsorted list of data
1. Set First to indicate first element of list
2. While First is less than or equal to the second to last element of the list
     2.1 Set Current to indicate element after First
     2.2 Set Temp equal to value at Current
```

Figure 8.3 *Exercise 47*


Start
Place the wall after the first element
While there are more elements in the unsorted list
false
true
Set Current to the last element in unsorted list
While Current greater than wall location
false
true
true
While value [current] < value [current -1]
false
Swap value [current] and value [current-1
Move Current back one element
Move wall forward one element
Start

```
2.3 While Current does not indicate the first element of the list AND
        the value of Temp is less than the value of element before Current
        2.3.1 Set value at Current to the value of the element before Current
        2.3.2 Set Current to indicate the element before Current
        End While
```

Figure 8.4 *Exercise 48*


Start
Place the wall after the first element
While there are more elements in the unsorted list
false
true
Set current to the element after the wall
While Current less than element before curretn
false
true
Swap value [current] and value [current-1
Move Current back one element
Move wall forward one element
Start

```
        2.4 Set the value at Current to the value in Temp
        2.5 Advance First
      End While
   3. Return list
   End
```

49. Selection sort:

14 7 23 31 40 56 78 9 2
2 | 7 23 31 40 56 78 9 14
2 7 | 23 31 40 56 78 9 14
2 7 9 | 31 40 56 78 23 14
2 7 9 14 | 40 56 78 23 31
2 7 9 14 23 | 56 78 40 31
2 7 9 14 23 31 | 78 40 56
2 7 9 14 23 31 40 | 78 56
2 7 9 14 23 31 40 56 | 78 (sorted)

50. Bubble sort:
14 7 23 31 40 56 78 9 2
2 | 14 7 23 31 40 56 78 9
2 7 | 14 9 23 31 40 56 78
2 7 9 | 14 23 31 40 56 78 (sorted)
2 7 9 14 | 23 31 40 56 78
2 7 9 14 23 | 31 40 56 78
2 7 9 14 23 31 | 40 56 78
2 7 9 14 23 31 40 | 56 78
2 7 9 14 23 31 40 56 | 78

51. Insertion sort:
14 7 23 31 40 56 78 9 2
14 | 7 23 31 40 56 78 9 2
7 14 | 23 31 40 56 78 9 2
7 14 23 | 31 40 56 78 9 2
7 14 23 31 | 40 56 78 9 2
7 14 23 31 40 | 56 78 9 2
7 14 23 31 40 56 | 78 9 2
7 14 23 31 40 56 78 | 9 2
7 9 14 23 31 40 56 78 | 2
2 7 9 14 23 31 40 56 78 | (sorted)

52. 7 8 13 23 26 44 98 57

53. 7 8 13 23 26 44 57 98

54. 3 7 13 23 26 44 98 57

55.

Pass 1:
first: 8 (index 0)
mid: 26 (index 3)
last: 97 (index 7)

```
Pass 2:
   first: 44 (index 4)
   mid: 56 (index 5)
   last: 97 (index 7)
Pass 3:
   first: 88 (index 6)
   mid: 88 (index 6) -- Target found
   last: 97 (index 7)
```

56.

```
Pass 1:
   first: 8 (index 0)
   mid: 26 (index 3)
   last: 97 (index 7)
Pass 2:
   first: 8 (index 0)
   mid: 13 (index 1)
   last: 17 (index 2) -- Target not found (Target > 17)
```

57.

```
GCD
Input: 2 integers (X and Y)
1. If X < Y
     1.1 Return GCD(Y, X)
2. Else if Y equals 0
     2.2 Return X
3. Else
     3.3 Return GCD(Y, X % Y )
   End if
End
```

58.

```
Combination
Input: 2 integers (n and k)
1. If k equals 0 OR if n equals k
   1.1 Return 1
2. Else if n is greater than k AND k is greater than 0
   2.2 Return (Combination (n - 1, k) + Combination (n - 1, k - 1) )
   End if
End
```