LAB 7

1. Lab 9 Abstract Data Type (Lab9\_Manual.pdf)

a. Exercise 1

1. Start the Stack-queue app.
2. Push the three names Alice, Bob, Carl onto the stack. When you pop the stack 3 times, what do you see in the text field next to Pop?

ANS: After 1st pop: Carl

After 2nd pop: Bob

After 3rd pop: Alice

After that stack is empty.

1. Write an algorithm in pseudocode or English that would describe how to use a stack to reverse any list. Write your answer below or type it in a word processor.

ANS: To reverse any list, we need to put elements(filling the stack) into the stack at first using push and then pop all the items (emptying the stack)one by one and print it. The printout is the reversed list.

Pseudocode:

WHILE(more data)

Read item // item is the element of the list

Push (myList, item) //myList is the name of the list

END WHILE

WHILE( NOT IsEmpty(myList))

Pop(myList, item)

Write item

END WHILE

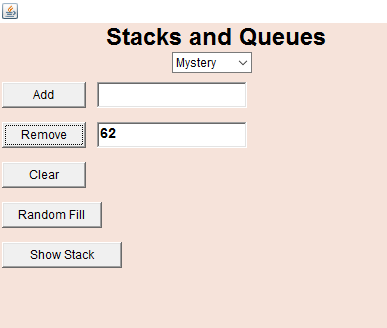
1. Describe how you could you use a stack in conjunction with the original list to determine whether a sequence is a palindrome. Write your answer below or type it in a word processor.

ANS: It can be done by repeating the steps that we did in above question that is by reversing the sequence. If the reverse of the sequence is same as the original then the sequence is palindrome.

b. Exercise 2

1. Start the app.
2. Click clear, then Random fill. Select mystery from the pull down menu.
3. Click once on Remove. Whatever number appears next to remove button is what you now type into the text field next to Add. Then click on add.
4. Now, click on remove. What values appear? Is it same or different? What does this mean? Is your mystery object a stack or queue? Take a screenshot and write your answer on the paper. You can then click on Show to confirm your conclusion.

ANS: 62 appears. It is same as what I added. This shows Last In, First Out which means mystery object is a stack.

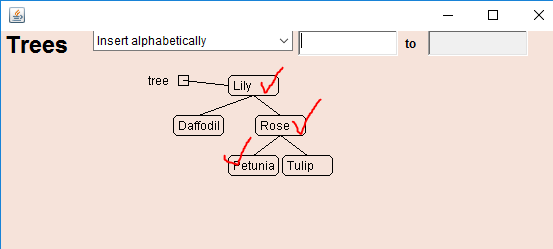


c. Exercise 3

1. Start the trees app.
2. Select Insert alphabetically from the pull-down menu. Then type the following flowers into the text field, pressing Enter after each one:

Lily, Rose, Daffodil, Tulip, Petunia

1. Take a screenshot. Now perform a manual search for “Poppy”. Put a check mark next to each node that you visit during search. You can confirm your ideas by selecting Find from the pull-down and asking the app to search for “Poppy”.

ANS: 

I visited Lily, Rose and Petunia to search for Poppy.

1. If you were to manually insert “Poppy” into this tree, what would you put into the boxes and which option would you select?

Check one blank on the left of Attach.. and fill in the two blanks surrounding to.

Lily Attach as left child Petunia to Rose

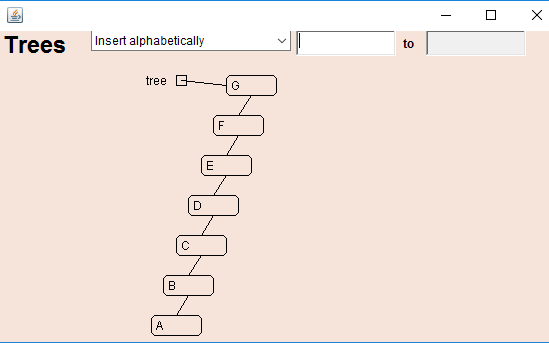
Poppy Attach as right child

It would be attached to the right of Petunia.

d. Exercise 4

1. Start the trees app.
2. The purpose is to build a binary search tree.
3. Type in the following flower names in the given order. Take a screenshot when done.

Gladiolus, Fern, Echinacea, Daisy, Coreopsis, Bluebell, Aster

ANS: 

1. What does your tree look like? Describe it in words below.

ANS: It is actually looking linear because all the elements are attached in the order that the latter is attached to the left of former.

1. What generalization can you make about inserting elements from a sorted list into a tree?

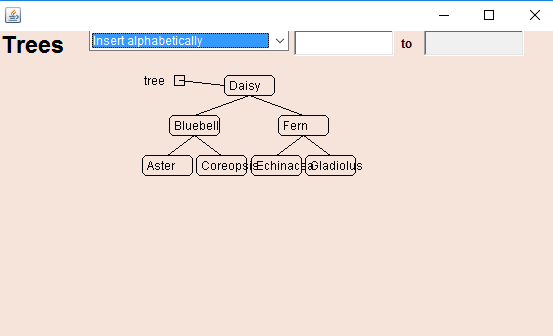
ANS: For sorted lists, as per my view the tree will attach all the elements of the list in linear order from right to left for decreasing order and left to right for increasing order.

1. Could you convince someone that a list is a special kind of tree? If so, what would you say?

ANS: Yes, a list is a special kind of tree because we can write all the items of a list in the tree. A tree is similar to an unordered list and binary search tree is like a sorted or ordered list.

1. Clear the tree and insert flowers so that the tree is balanced and complete. Take a screenshot. Also, write down the correct order that you used to make the tree look this way.

ANS:



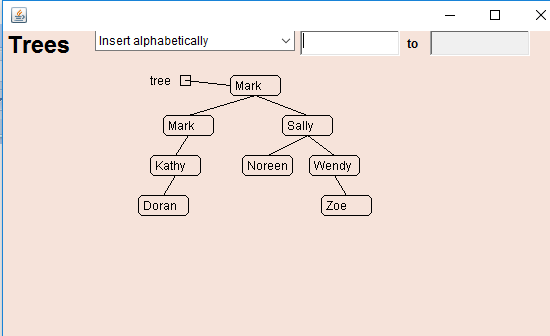
Order that I used is Daisy, Bluebell, Fern, Aster, Coreopsis, Gladiolus, Echinacea.

1. Is there more than one order that you could have used to create the same balanced, complete tree? Why there are multiple orders possible?

ANS: Yes, there can be many other orders to create a same balanced tree. Because the tree will do alphabetical ordering while inserting the members but the thing that we need to make sure is that the root node should be Daisy.

e. Exercise 5

1. Start the trees app.
2. The purpose of this is to investigate what happens when you have duplicate nodes. Choose example 2.
3. Select insert alphabetically. Then type “Mark” into the text field and press Enter. Take a screenshot.

ANS: 

1. Insert another name that is already in the tree. What does this app do at this?

ANS: This app just attaches the name to the left of the location where this name is already placed.

1. List two alternatives strategies that the app could have used.

ANS: (a.) It could insert the name to right of the previous one.

(b.) It could have used the count method(that is to write the number of times that name occurred in the brackets at the end of the name).

1. Does final still work? Try to find a name, such as Mark, that is represented by two nodes. Describe what happens.

ANS: Yes, Find still works and it just stops when it find the required name for the first time.

2. What is the output of the following stack algorithm? Push(myStack, 'Jack')

Push(myStack, 'Jose')

Push(myStack, 'Ali ')

Push(myStack, 'Mike ')

Pop(myStack, name)

Write name + ", "

Pop(myStack, name)

Push(myStack, name)

Push(myStack, name)

Push(myStack, 'Harman ' )

WHILE (NOT IsEmtpy(myStack))

Pop(myStack, name)

Write name + ", "

END WHILE

ANS:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | Harman |  |
|  |  |  | Mike |  |  |  | Ali | Ali |  |
|  |  | Ali | Ali | Ali |  | Ali | Ali | Ali |  |
|  | Jose | Jose | Jose | Jose | Jose | Jose | Jose | Jose |  |
| Jack | Jack | Jack | Jack | Jack | Jack | Jack | Jack | Jack |  |

At First Pop(myStack, name) :name was Mike

Printout: Mike, Harman, Ali, Ali, Jose, Jack,

3. What is the output of the following stack algorithm? Push(myStack, 2)

Push(myStack, 7)

Push(myStack, 8)

Push(myStack, 5)

Push(myStack, 9)

Push(myStack, 1)

Pop(myStack, item)

Write item + ", "

Pop(myStack, item)

Pop(myStack, item)

Write item + ", "

Push(myStack, item)

Push(myStack, item)

WHILE (NOT IsEmtpy(myStack))

Pop(myStack, item)

Write item + ", "

END WHILE

ANS:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 1 |  |  |  |  |  |
|  |  |  |  | 9 | 9 | 9 |  |  |  | 5 |
|  |  |  | 5 | 5 | 5 | 5 | 5 |  | 5 | 5 |
|  |  | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
|  | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

ITEM:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 1 | 9 | 5 |  |  |

Printout:

1, 5, 5, 5, 8, 7, 2,

4. What is the output of the following queue algorithm? Enque(myQueue, 'Mary ' )

Enque(myQueue, 'Susan ' )

Deque(myQueue, name)

Write name + ", "

Enque(myQueue, 'Sara ' )

Enque(myQueue, 'Tom ' )

Deque(myQueue, name)

Write name + ", "

Deque(myQueue, name)

Enque(myQueue, 'Chang')

Enque(myQueue, name)

Enque(myQueue, name)

WHILE (NOT IsEmtpy(myQueue))

Deque(myQueue, name)

Write name + ", "

END WHILE

ANS:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BACK |  |  |  |  |  | Front | Name |
| Enque(myQueue,’Mary’) |  |  |  |  |  |  | Mary |  |
| Enque(myQueue,’Susan’) |  |  |  |  |  | Susan | Mary |  |
| Deque(myQueue, name) |  |  |  |  |  | Susan |  | Mary |
| Enque(myQueue,’Sara’) |  |  |  |  | Sara | Susan |  |  |
| Enque(myQueue,’Tom’) |  |  |  | Tom | Sara | Susan |  |  |
| Deque(myQueue,name) |  |  |  | Tom | Sara |  |  | Susan |
| Deque(myQueue,name) |  |  |  | Tom |  |  |  | Sara |
| Enque(myQueue,’Chang’) |  |  | Chang | Tom |  |  |  |  |
| Enque(myQueue, name) |  | Sara | Chang | Tom |  |  |  |  |
| Enque(myQueue,name) | Sara | Sara | Chang | Tom |  |  |  |  |
| LOOP |  |  |  |  |  |  |  |  |
| Deque(myQueue,name) | Sara | Sara | Chang |  |  |  |  | Tom |
| Deque(myQueue,name) | Sara | Sara |  |  |  |  |  | Chang |
| Deque(myQueue,name) | Sara |  |  |  |  |  |  | Sara |
| Deque(myQueue,name) |  |  |  |  |  |  |  | Sara |
|  |

Printout: Mary, Susan, Tom, Chang, Sara, Sara,

5. What is the output of the following queue algorithm? Enque(myQueue, 2)

Enque(myQueue, 7)

Enque(myQueue, 8)

Enque(myQueue, 5)

Enque(myQueue, 9)

Enque(myQueue, 1)

Deque(myQueue, item)

Write item + " "

Deque(myQueue, item)

Deque(myQueue, item)

Write item + " "

Enque(myQueue, item)

WHILE (NOT IsEmtpy(myQueue))

Deque(myQueue, item)

Write item + " " END WHILE

ANS:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | BACK |  |  |  |  | FRONT | item |
| Enque(myQueue,2) |  |  |  |  |  | 2 |  |
| Enque(myQueue,7) |  |  |  |  | 7 | 2 |  |
| Enque(myQueue,8) |  |  |  | 8 | 7 | 2 |  |
| Enque(myQueue,5) |  |  | 5 | 8 | 7 | 2 |  |
| Enque(myQueue,9) |  | 9 | 5 | 8 | 7 | 2 |  |
| Enque(myQueue,1) | 1 | 9 | 5 | 8 | 7 | 2 |  |
| Deque(myQueue,item) | 1 | 9 | 5 | 8 | 7 |  | 2 |
| Deque(myQueue,item) | 1 | 9 | 5 | 8 |  |  | 7 |
| Deque(myQueue,item) | 1 | 9 | 5 |  |  |  | 8 |
| Enque(myQueue,item) | 1 | 9 | 5 | 8 |  |  |  |
| LOOP |  |  |  |  |  |  |  |
| Deque(myQueue,item) | 1 | 9 | 5 |  |  |  | 8 |
| Deque(myQueue,item) | 1 | 9 |  |  |  |  | 5 |
| Deque(myQueue,item) | 1 |  |  |  |  |  | 9 |
|  |  |  |  |  |  |  | 1 |

Printout: 2 8 8 5 9

6. Write a pseudocode that sets variable bottom equal to the last element in a stack:

a. leaving the stack empty.

ANS: Let’s say we have stack named myStack

WHILE (NOT IsEmpty(myStack)){

Pop(myStack, bottom)

Print bottom

}

b. leaving the stack unchanged. [Hint: Store the stack elements in another stack as you pop them and then push them back to the original stack.]

ANS: Let the number of elements in original stack is x.

Let original stack = originalStack and new stack = newStack

Set x= 0;

WHILE (NOT IsEmpty(myStack)){

Pop(originalStack, bottom)

Push(newStack, bottom)

}

END WHILE

7. Write a pseudocode that sets variable last equal to the last element in a queue:

a. leaving the queue empty.

ANS: WHILE (NOT IsEmpty(originalQueue)){

Pop(originalQueue, last)

Print last

}

END WHILE

b. leaving the queue unchanged. [Hint: Store the queue elements into another queue as you dequeue them and then store them back to the original queue.]

ANS: WHILE(NOT isEmpty(myQueue)){

Deque(originalQueue, last )

Enque(newQueue, last)

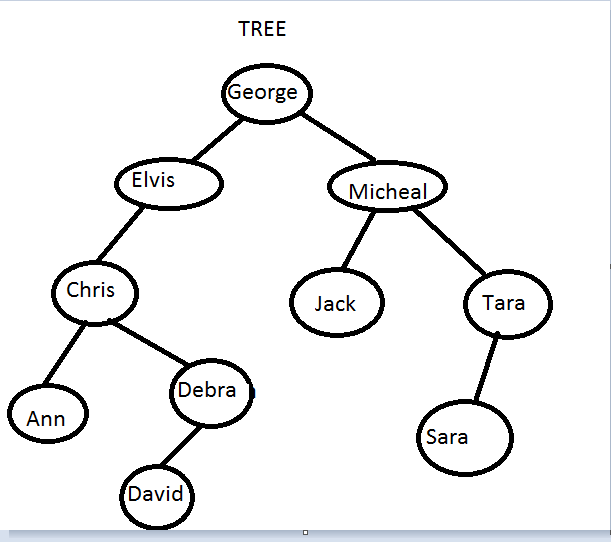
}

END WHILE

8. Given the following set of names:

George Michael Jack Elvis Tara Chris Debra Ann David Sara

a. Draw the Binary Search Tree that results from inserting the names in the given order.

ANS: 

b. How many comparisons are required to find Debra in this tree?

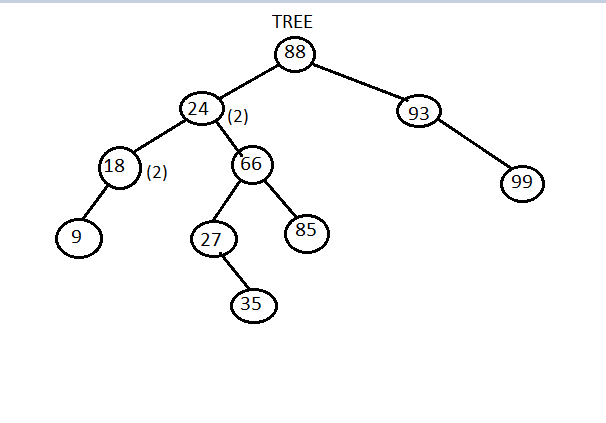
ANS: 4 Comparisons are required.

9. Given the following set of numbers:

4 88 93 24 18 24 99 9 66 27 18 35 85

a. [6] Draw the Binary Search Tree that results from inserting the numbers in the given order.

ANS:



b. How many comparisons are required to find the value 27 in this tree?

ANS: 5 Comparisons are required.