//////////////////////////////////////////////////////////////////////////////////

/\* CE1007/CZ1007 Data Structures

2016/17 S1

Author and Lab Group: ALVIN LEE YONG TECK / FSP7

Program name: FSP7\_ALVIN\_LEE\_YONG\_TECK

Date: 03 November 2016

Purpose: Implementing the required functions for Assignment 1\*/

//////////////////////////////////////////////////////////////////////////////////

Question 1

//////////////////////////////////////////////////////////////////////////////////

// Program Listing

//////////////////////////////////////////////////////////////////////////////////

int insertSortedLL(LinkedList \*ll, int item)

{

ListNode \*cur; // Declare ListNote cur (a.k.a current) to navigate the  
LinkedList

int index = 0; // Declare index of the item to be inserted

cur = ll->head; // Let ListNode cur be the first ll's head (ll->head)

do {

// If current ListNode is empty...

if (cur == NULL) {

insertNode(ll, index, item); // Insert item into LinkedList   
ll at index (0, 1, 2, ...)

return index; // Return value of index

}

// If current ListNode is greater than item to be inserted...   
 (e.g. 6 > 4)

else if (cur->item > item) {

insertNode(ll, index, item); // Insert item into LinkedList   
ll at index (0, 1, 2, ...)

return index; // Return value of index

}

// If current ListNode is lesser than item to be inserted...   
 (e.g. 1 < 3)

else if (cur->item < item) {

cur = cur->next; // Navigate ListNode cur to the next listnode

index++; // Increment index (+ 1)

}

// If current ListNode is equal to the item to be inserted...   
 (e.g. 2 == 2)

else if (cur->item == item) {

return -1; // Return: -1 as item already existed! Unable to   
insert!

}

} while (1);

}

///////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////

// Test Cases

|  |  |
| --- | --- |
| **Input** | **Output** |
| - | 1: Insert an integer to the sorted linked list:  2: Print the index of the most recent input value:  3: Print sorted linked list:  0: Quit:  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 2 | The resulting linked list is: 2  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 3 | The resulting linked list is: 2 3  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 5 | The resulting linked list is: 2 3 5  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 7 | The resulting linked list is: 2 3 5 7  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 9 | The resulting linked list is: 2 3 5 7 9  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 8 | The resulting linked list is: 2 3 5 7 8 9  Please input your choice(1/2/3/0): |
| 2 | The value 8 was added at index 4  Please input your choice(1/2/3/0): |
| 3 | The resulting sorted linked list is: 2 3 5 7 8 9  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 5 | The resulting linked list is: 2 3 5 7 8 9  Please input your choice(1/2/3/0): |
| 2 | The value 5 was added at index -1  Please input your choice(1/2/3/0): |
| 3 | The resulting sorted linked list is: 2 3 5 7 8 9  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 11 | The resulting linked list is: 2 3 5 7 8 9 11  Please input your choice(1/2/3/0): |
| 2 | The value 11 was added at index 6  Please input your choice(1/2/3/0): |
| 3 | The resulting sorted linked list is: 2 3 5 7 8 9 11  Please input your choice(1/2/3/0): |

//////////////////////////////////////////////////////////////////////////////////

Question 2

//////////////////////////////////////////////////////////////////////////////////

// Program Listing

//////////////////////////////////////////////////////////////////////////////////

void moveOddItemsToBack(LinkedList \*ll)

{

int i; // Declare 'for' loop index i for looping

int curIndex; // Declare curIndex (a.k.a current index) to identify the   
 index for removeNode & findNode

ListNode \*cur; // Declare ListNote cur (a.k.a current) to navigate the   
 LinkedList

curIndex = 0; // Set curIndex to be 0

// Return if LinkedList ll is empty / NULL

if (ll == NULL) {

return;

}

cur = ll->head; // Let ListNode cur be the first ll's head (ll->head)

// Loop through the LinkedList ll from index 0 to the size of ll

for (i = 0; i < ll->size; i++) {

// If current item (cur->item) divided by 2 has a remainder of 1...

// \*Odd number\*

if (cur->item % 2 == 1) {

// Insert cur->item into back of LinkedList ll at index

insertNode(ll, ll->size, cur->item);

// Remove the current index's item in the LinkedList ll

removeNode(ll, curIndex);

// Relocate the position of cur to the new current index's item cur = findNode(ll, curIndex);

}

// \*Even number\*

else {

cur = cur->next; // Navigate ListNode cur to the next listnode

curIndex++; // Increment current index (+ 1)

}

}

}

//////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////

// Test Cases

// Scenario 1

|  |  |
| --- | --- |
| **Input** | **Output** |
| - | 1: Insert an integer to the sorted linked list:  2: Moves all odd integers to the back of the linked list:  0: Quit:  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 2 | The resulting linked list is: 2  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 3 | The resulting linked list is: 2 3  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 4 | The resulting linked list is: 2 3 4  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 7 | The resulting linked list is: 2 3 5 7  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 15 | The resulting linked list is: 2 3 5 7 15  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 18 | The resulting linked list is: 2 3 5 7 18  Please input your choice(1/2/0): |
| 2 | The resulting Linked List after moving odd integers to the back of the Linked List is: 2 4 18 3 7 15  Please input your choice(1/2/0): |

// Scenario 2

|  |  |
| --- | --- |
| **Input** | **Output** |
| - | 1: Insert an integer to the sorted linked list:  2: Moves all odd integers to the back of the linked list:  0: Quit:  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 1 | The resulting linked list is: 1  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 3 | The resulting linked list is: 1 3  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 5 | The resulting linked list is: 1 3 5  Please input your choice(1/2/0): |
| 2 | The resulting Linked List after moving odd integers to the back of the Linked List is: 1 3 5  Please input your choice(1/2/0): |

// Scenario 3

|  |  |
| --- | --- |
| **Input** | **Output** |
| - | 1: Insert an integer to the sorted linked list:  2: Moves all odd integers to the back of the linked list:  0: Quit:  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 2 | The resulting linked list is: 2  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 4 | The resulting linked list is: 2 4  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 6 | The resulting linked list is: 2 4 6  Please input your choice(1/2/0): |
| 2 | The resulting Linked List after moving odd integers to the back of the Linked List is: 2 4 6  Please input your choice(1/2/0): |

//////////////////////////////////////////////////////////////////////////////////

Question 3

//////////////////////////////////////////////////////////////////////////////////

// Program Listing

//////////////////////////////////////////////////////////////////////////////////

void createQueueFromLinkedList(LinkedList \*ll, Queue \*q)

{

ListNode \*cur; // Declare ListNote cur (a.k.a current) to navigate the   
LinkedList

int i; // Declare 'for' loop index i for looping

// If the Queue q is not empty...

if (!isEmptyQueue(q)) {

removeAllItemsFromQueue(q); // Remove all the items from Queue q

}

cur = ll->head; // Let ListNode cur be the first ll's head (ll->head)

// Loop through the LinkedList ll from index 0 to the size of ll

for (i = 0; i < ll->size; i++) {

enqueue(q, cur->item); // Enqueue the current item into Queue q

cur = cur->next; // Navigate ListNode cur to the next listnode

}

}

void removeOddValues(Queue \*q)

{

int i; // Declare 'for' loop index i for looping

int output; // Declare the 'output' to store the dequeue's output

int size; // Declare the 'size' for Queue

size = q->ll.size; // Set the 'size' to be the size of Queue q (q->ll.size)

// Loop from index 0 to the size of Queue q (Number of times to dequeue)

for (i = 0; i < size; i++) {

output = dequeue(q); // Store the dequeue's output into 'output'

// If output divided by 2 has a remainder of 0...

// \*Even number\*

if (output % 2 == 0) {

enqueue(q, output); // Enqueue the EVEN number output into the   
back of Queue q

}

}

}

//////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////

// Test Cases

|  |  |
| --- | --- |
| **Input** | **Output** |
| - | 1: Insert an integer into the linked list:  2: Create the queue from the linked list:  3: Remove odd numbers from the queue:  0: Quit:  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 1 | The resulting linked list is: 1  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 2 | The resulting linked list is: 1 2  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 3 | The resulting linked list is: 1 2 3  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 4 | The resulting linked list is: 1 2 3 4  Please input your choice(1/2/3/0): |
| 1 | Input an integer that you want to add to the linked list: |
| 5 | The resulting linked list is: 1 2 3 4 5  Please input your choice(1/2/3/0): |
| 2 | The resulting queue is: 1 2 3 4 5  Please input your choice(1/2/3/0): |
| 3 | The resulting queue after removing odd integers is: 2 4  Please input your choice(1/2/3/0): |

//////////////////////////////////////////////////////////////////////////////////

Question 4

//////////////////////////////////////////////////////////////////////////////////

// Program Listing

//////////////////////////////////////////////////////////////////////////////////

void printPostOrderIterative(BSTNode \*root)

{

Stack \*s1, \*s2; // Declare 2 Stacks (s1 & s2)

BSTNode \*cur; // Declare BSTNode cur (a.k.a current) to

s1 = malloc(sizeof(s1)); // Dynamically allocate memory for Stack s1

s2 = malloc(sizeof(s2)); // Dynamically allocate memory for Stack s2

s1->top = NULL; // Point the top of Stack s1 to NULL

s2->top = NULL; // Point the top of Stack s2 to NULL

// If Stack s1 is not empty, point the top of stack to NULL

if (!isEmpty(s1)) {

s1->top = NULL;

}

// If Stack s2 is not empty, point the top of stack to NULL

if (!isEmpty(s2)) {

s2->top = NULL;

}

// If BSTNode root is NULL, return back to main function

if (root == NULL) {

return;

}

push(s1, root); // Push the root node to the first stack

// Repeat until the Stack s1 is empty

while (!isEmpty(s1)) {

// Pop a node out from Stack s1 and push it into Stack s2

push(s2, pop(s1));

// Set the node that recently being pushed into Stack s2 to BSTNode cur

cur = s2->top->item;

// Push the node's left child to Stack s1

if (cur->left != NULL) {

push(s1, cur->left);

}

// Push the node's right child to Stack s1

if (cur->right != NULL) {

push(s1, cur->right);

}

}

// Repeat until Stack s2 is empty

while (!isEmpty(s2)) {

// Print the output that being popped out from Stack s2

printf("%d ", pop(s2)->item);

}

}

//////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////

// Test Cases

|  |  |
| --- | --- |
| **Input** | **Output** |
| - | 1:Insert an integer into the binary search tree;  2:Print the post-order traversal of the binary search tree;  0:Quit;  Please input your choice(1/2/0): |
| 1 | Input an integer that you want to insert into the Binary Search Tree: |
| 6 | Please input your choice(1/2/0): |
| 1 | Input an integer that you want to insert into the Binary Search Tree: |
| 34 | Please input your choice(1/2/0): |
| 1 | Input an integer that you want to insert into the Binary Search Tree: |
| 17 | Please input your choice(1/2/0): |
| 1 | Input an integer that you want to insert into the Binary Search Tree: |
| 19 | Please input your choice(1/2/0): |
| 1 | Input an integer that you want to insert into the Binary Search Tree: |
| 16 | Please input your choice(1/2/0): |
| 1 | Input an integer that you want to insert into the Binary Search Tree: |
| 10 | Please input your choice(1/2/0): |
| 1 | Input an integer that you want to insert into the Binary Search Tree: |
| 23 | Please input your choice(1/2/0): |
| 1 | Input an integer that you want to insert into the Binary Search Tree: |
| 3 | Please input your choice(1/2/0): |
| 2 | The resulting post-order traversal of the binary search tree is: 3 10 16 23 19 17 34 6  Please input your choice(1/2/0): |