Module Code / Name	**:**	**CZ1007 / Data Structures**
Academic Year	**:**	**2016/2017 Semester 1**
Author and lab group	**:**	**HTET NAING / FSP 7**
Program Name	**:**	**FSP7_Htet Naing**
Date	**:**	**Nov 10, 2016**
Purpose	**:**	**Implementing the required functions for Assignment 1**

CONTENTS

QUESTION (1)

```
/*	The function below aims to sort the items in the linkedlist in
	the ascending order.
	Whenever a new item is added, it will be inserted at the suitable
	place to maintain the ascending order.
	If the insertion succeeds, the function returns the index
	position of the new node that is added.
	Otherise it returns -1. */

int insertSortedLL(LinkedList *ll, int item)
{
	//i and j are used to keep track of the current node and the
	index position of the new item in the linkedlist respectively.
	int i, j, sameNumber;
	//sameNumber is used to check if the new item has the same value
	as the existing item. Initially set to 0.
	i = j = sameNumber = 0;
	//ptr is the temporary pointer used to traverse nodes in the
	linkedlist.
	ListNode *ptr = ll->head;
	//First the program checks whether the size of the linked list is
	zero.
	if (ll->size == 0) {
		//If it is zero, the first node is inserted at index 0.
		insertNode(ll, j, item);
	}
	else {
		//Otherwise, the new item is compared against all the
		existing items.
		//It can be done by traversing existing nodes in the
		linkedlist using the temporary pointer(ptr).
		while (ptr != NULL) {
			//If the new item is greater than current item, the
			current index+1 is assigned to j
			//(because the bigger number should come after the
			current index number).
			if (item > ptr->item) {
				j = i + 1;
			}
			//If the new item is similar to one of the existing
			items, sameNumber is set to 1 and j to -1.
			else if (item == ptr->item) {
```

```
                j = -1;
                sameNumber = 1;
            }
            ptr = ptr->next;
            i++;
        }
        //the new item will be added only if the sameNumber is not
        set to 1.
        if (!sameNumber) {
            insertNode(ll, j, item);
        }
    }
    //The function returns j value which represents the index
    position of the new node.
    return j;
}
```

Question (1): GIVEN TEST CASES

```
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:
```

INPUT	CONSOLE WINDOW	OUTPUT

1
2
	`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`
		2

1
3
	`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`
		2 3

1
5
	`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`
		2 3 5

1
7
	`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`
		2 3 5 7

1
9
	`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`
		2 3 5 7 9

1
8
	`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`
		2 3 5 7 8 9

2	`Please input your choice(1/2/3/0): 2`
	`The value 8 was added at index 4`
		index 4

3	`Please input your choice(1/2/3/0):3`
	`The resulting sorted linked list is: 2 3 5 7 8 9`
		2 3 5 7 8 9

1 5	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**	**2 3 5 7 8 9**
2	Please input your choice(1/2/3/0): **2** **The value 5 was added at index -1**	**index -1**
3	Please input your choice(1/2/3/0): **3** The resulting sorted linked list is: **2 3 5 7 8 9**	**2 3 5 7 8 9**
1 11	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**	**2 3 5 7 8 9 11**
2	Please input your choice(1/2/3/0): **2** **The value 11 was added at index 6**	**index 6**
3	Please input your choice(1/2/3/0): **3** The resulting sorted linked list is: **2 3 5 7 8 9 11**	**2 3 5 7 8 9 11**

Question (1): SELF-GENERATED TEST CASES

```
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:
```

INPUT	**CONSOLE WINDOW**	**OUTPUT**
1 **2**	Please input your choice(1/2/3/0): **1** Input an integer that you want to add to the linked list: **2** The resulting sorted linked list is: **2**	**2**
1 **0**	Please input your choice(1/2/3/0): 1 Input an integer that you want to add to the linked list: **0** The resulting sorted linked list is: **0 2**	**0 2**

1 1	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: **0 1 2**	0 1 2
1 -1	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 0 1 2**	-1 0 1 2
1 2	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 0 1 2**	-1 0 1 2
2	Please input your choice(1/2/3/0): 2 **The value 2 was added at index -1**	index -1
1 8	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: **-1 0 1 2 8**	-1 0 1 2 8
2	Please input your choice(1/2/3/0): **2** **The value 8 was added at index 4**	index 4
1 5	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 0 1 2 5 8**	-1 0 1 2 5 8
2	Please input your choice(1/2/3/0): **2** **The value 5 was added at index 4**	index 4
3	Please input your choice(1/2/3/0): **3** The resulting sorted linked list is: **-1 0 1 2 5 8**	-1 0 1 2 5 8

QUESTION (2)

```
// The function below aims to move all the odd items in a linkedlist
to the back and the even items, to the front.

void moveOddItemsToBack(LinkedList *ll)
{
      //temp pointer is used to traverse the nodes in each linkedlist.
      ListNode *temp = ll->head;
      //even linkedlist and odd linkedlist are used to store even and
      odd items respectively.
      LinkedList evenList, oddList;
      //Initially, the sizes and head pointers of the above two
      linkedlists are set to 0 and NULL respectively.
      evenList.size = oddList.size = 0;

      evenList.head = oddList.head = NULL;

      while (temp != NULL) {
            //First all the items in the current linkedlist are checked
            if they are odd or even.
            if (temp->item % 2 == 0) {
            //Even items are added to evenList and odd items, to oddList
            accordingly.
                  insertNode(&evenList, evenList.size, temp->item);
            }
            else {
                  insertNode(&oddList, oddList.size, temp->item);
            }
            temp = temp->next;
      }
      //After that, all items in the current linkedlist are removed.
      removeAllItems(ll);

      temp = evenList.head;
      while (temp != NULL) {
            //Next, even items are added to the empty linkedlist first.
            //When adding new items, "size" variable of the respective
            linkedlist is used as the index positions.
            insertNode(ll, ll->size, temp->item);


            temp = temp->next;
```

```
	}

	temp = oddList.head;
	while (temp != NULL) {
		//After adding even items, odd items are added back to the
		linkedlist.
		insertNode(ll, ll->size, temp->item);
		temp = temp->next;
	}
}
```

Question (2): GIVEN TEST CASES

```
1: Insert an integer to the sorted linked list:
2: Moves all odd integers to the back of the linked list:
0: Quit:
```

INPUT	CONSOLE WINDOW	OUTPUT
	If the current linked list is **2 3 4 7 15 18:**	
2	The resulting Linked List after moving odd integers to the back of the Linked List is: **2 4 18 3 7 15**	2 4 18 3 7 15
	If the current linked list is **1 3 5:**	
2	The resulting Linked List after moving odd integers to the back of the Linked List is: **1 3 5**	1 3 5
	If the current linked list is **2 4 6:**	
2	The resulting Linked List after moving odd integers to the back of the Linked List is: **2 4 6**	2 4 6

Question (2): SELF-GENERATED TEST CASES

INPUT	CONSOLE WINDOW	OUTPUT
	If the current linked list is **-2 -1 0 1 2 6 17 23:**	
2	The resulting Linked List after moving odd integers to the back of the Linked List is: **-2 0 2 6 -1 1 17 23**	-2 0 2 6 -1 1 17 23

QUESTION (3)

```
//The function below aims to create a Queue based on the items from a
linkedlist.

void createQueueFromLinkedList(LinkedList *ll, Queue *q)
{
	//ptr is used as the temporary pointer to traverse nodes in the
	linkedlist.
	ListNode *ptr = ll->head;
	 //In case, if the queue is not empty, all the items are removed
	first.
	if (!isEmptyQueue(q)) {
		removeAllItemsFromQueue(q);
	}
	while (ptr != NULL) {
	//A new queue is created by enqueuing all the items from the
	linkedlist.
		enqueue(q, ptr->item);
		ptr = ptr->next;
	}
}

//The function below aims to remove all the odd items from the Queue

void removeOddValues(Queue *q)
{
	//tempQueue is used as the temporary queue to store even items
	from the current Queue.
	Queue tempQueue;
	//Initially the size and head pointer of the tempQueue are set to
	0 and NULL respectively.
	tempQueue.ll.size = 0;
	tempQueue.ll.head = NULL;

	while (!isEmptyQueue(q)) {
		//Firstly, the program checks if the item in the current
		queue is even.
		if (q->ll.head->item % 2 == 0) {
			//If it is even, the item is added to tempQueue.
			enqueue(&tempQueue, q->ll.head->item);
		}
```

```
            //Then the item is dequeued from the current queue.

            //The above procedure repeats until the current queue is
            empty.
            dequeue(q);
        }
        while (!isEmptyQueue(&tempQueue)) {
            //Secondly, the even items from the tempQueue are added to
            the original queue which is currently empty.
            enqueue(q, tempQueue.ll.head->item);
            //Then the first even item is dequeued from the tempQueue.
            //The above procedure repeats until the tempQueue is empty.
            dequeue(&tempQueue);
        }
    }
```

Question (3): GIVEN TEST CASES

```
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:
```

INTPUT	CONSOLE WINDOW	OUTPUT

	`(if the current linked list is 1 2 3 4 5):`
	`The resulting linked list is: 1 2 3 4 5`
		1 2 3 4 5

2	`Please input your choice(1/2/3/0): 2`
	`The resulting queue is: 1 2 3 4 5`
		1 2 3 4 5

3	`Please input your choice(1/2/3/0): 3`
	`The resulting queue after removing odd integers is: 2 4`
		2 4

Question (3): SELF-GENERATED TEST CASES

INPUT	CONSOLE WINDOW	OUTPUT

	`(if the current linked list is -3 -6 5 0 2 1 4):`
	`The resulting linked list is: -3 -6 5 0 2 1 4`
		-3 -6 5 0 2 1 4

2	`Please input your choice(1/2/3/0): 2`
	`The resulting queue is: -3 -6 5 0 2 1 4`
		-3 -6 5 0 2 1 4

3	`Please input your choice(1/2/3/0): 3`
	`The resulting queue after removing odd integers is: -6 0 2 4`
		-6 0 2 4

QUESTION (4)

```
//The function below aims to print the items in a Binary Search Tree
in post-order manner by using iterative method with two stacks.

void printPostOrderIterative(BSTNode *root)
{
      //Two stacks are created to store the nodes while traversing the
      binary search tree.
      Stack a;
      Stack b;
      //Initially, the top pointers of the two new stacks are set to
      NULL;
      a.top =  NULL;
      b.top =  NULL;
      //temp is used as the temporary pointer to store the current
      node.
      BSTNode *temp;

      //First, the root node is pushed to Stack a.
      push(&a, root);
      while (!isEmpty(&a)){
      //Then it is popped and assigned to temp.
            temp = pop(&a);
            //Next, temp is pushed to Stack b.
            push(&b, temp);
            //Now Stack a is empty and Stack b is holding the root node
            pushed by temp.
            //After that, the program checks if the current root node
            has any left or right child.

            if (temp->left != NULL) {
            //If it has any of them, they are pushed to stack a.
                  push(&a, temp->left);
            }
            if (temp->right != NULL) {
                  push(&a, temp->right);
            //Now Stack a is holding the child/children of the current
            root node.
            }
      //The above procedure repeats until the stack a is empty (i.e.
      until when there are no more nodes to traverse in the tree).
      }
```

```
    //At this stage, Stack a is empty and Stack b has all the nodes
    arranged in post-order traversal manner (top to bottom).

    while (!isEmpty(&b)) {
         //Now, the top item(node) in Stack b is popped to temp.
         temp = pop(&b);
         //The item in the temp node is then printed out.
         printf("%d ", temp->item);
    //The above procedure repeats until the stack b is empty.
    }
}
```

Question (4): TEST CASES

```
A sample input and output session is given below: (Add the following
nodes to a binary search tree in the order they appear:
6 34 17 19 16 10 23 3)

1:Insert an integer into the binary search tree;
2:Print the post-order traversal of the binary search tree;
0:Quit;
```

INPUT	CONSOLE WINDOW	OUTPUT

1
6
	`Please input your choice(1/2/0): `**`1`**
	`Input an integer that you want to insert into the Binary Search Tree: `**`6`**	

1
34
	`Please input your choice(1/2/0): `**`1`**
	`Input an integer that you want to insert into the Binary Search Tree: `**`34`**	

1
17
	`Please input your choice(1/2/0): `**`1`**
	`Input an integer that you want to insert into the Binary Search Tree: `**`17`**	

1
19
	`Please input your choice(1/2/0): `**`1`**
	`Input an integer that you want to insert into the Binary Search Tree: `**`19`**	

1
16
	`Please input your choice(1/2/0): `**`1`**
	`Input an integer that you want to insert into the Binary Search Tree: `**`16`**	

1
10
	`Please input your choice(1/2/0): `**`1`**
	`Input an integer that you want to insert into the Binary Search Tree: `**`10`**	

1
23
	`Please input your choice(1/2/0): `**`1`**
	`Input an integer that you want to insert into the Binary Search Tree: `**`23`**	

1
3
	`Please input your choice(1/2/0):` **`1`**
	`Input an integer that you want to insert into the Binary Search Tree:` **`3`**	

2	`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`
		`3 10 16 23 19 17 34 6`

Question (4): SELF-GENERATED TEST CASES

```
A sample input and output session is given below: (Add the following
nodes to a binary search tree in the order they appear:
-3 -6 5 0 2 1 6 -9)
```

```
1:Insert an integer into the binary search tree;
2:Print the post-order traversal of the binary search tree;
0:Quit;
```

INPUT	CONSOLE WINDOW	OUTPUT

1
-3
	`Please input your choice(1/2/0):` **`1`**
	`Input an integer that you want to insert into the Binary Search Tree:` **`-3`**	

1
-6
	`Please input your choice(1/2/0):` **`1`**
	`Input an integer that you want to insert into the Binary Search Tree:` **`-6`**	

1
5
	`Please input your choice(1/2/0):` **`1`**
	`Input an integer that you want to insert into the Binary Search Tree:` **`5`**	

1
0
	`Please input your choice(1/2/0):` **`1`**
	`Input an integer that you want to insert into the Binary Search Tree:` **`0`**	

1
2
	`Please input your choice(1/2/0):` **`1`**
	`Input an integer that you want to insert into the Binary Search Tree:` **`2`**	

1
1
	`Please input your choice(1/2/0):` **`1`**
	`Input an integer that you want to insert into the Binary Search Tree:` **`1`**	

1
6
	`Please input your choice(1/2/0):` **`1`**
	`Input an integer that you want to insert into the Binary Search Tree:` **`6`**	

1
-9
	`Please input your choice(1/2/0):` **`1`**
	`Input an integer that you want to insert into the Binary Search Tree:` **`-9`**	

2	`Please input your choice(1/2/0):` **`2`**
	`The resulting post-order traversal of the binary search tree is:`
	`-9 -6 1 2 0 6 5 -3`
		`-9 -6 1 2 0 6 5 -3`