Boris Wrabel
Student ID: 2206447w
Course: Algorithms and Data Structures
2018/2019

# Assessed Exercise

## Part A

**Algorithms used in JavaExercise1:**

1. MyListIterator

```
1. Initialise iterator over the elements in list
2. Print out the section title
3. While iterator has more elements, repeat:
   3.1. Print out the next element in the iteration
4. Terminate yielding iterator
```

2. MyEnhancedLoop

```
1. Add an element “Peter Prison Museum” at index 0 to list
2. Add an element “Wick Heritage Museum” at index 1 to list
3. Print out the section title “Title – ArrayList after
   adding elements”
4. For each element in list repeat:
   4.1. Print out element
5. Terminate
```

3. MyWhileLoop

```
1. Initialise counter
2. Remove element from list
3. Print out the section title
4. While the size of list is greater than counter, repeat:
   4.1. Print out the element at the specified index in list.
   4.2. Increment counter
5. Terminate
```

4. MyForLoop

```
1. Remove the specified element at index 4
2. Print out the section title
3. For i = 0, i less than the size of list, increment i
   3.1. Print out the element at the specified index in list
4. Terminate
```

**Algorithms used in MySets:**

1. union

   1. Construct an empty set “union”
   2. Add all of the elements in set 1 to union
   3. Add all of the elements in set 2 to union
   4. Terminate yielding union

2. union1

   1. Construct an empty set “union”
   2. Add all of the elements in set 1 to union
   3. Add all of the elements in set 2 to union
   4. Terminate yielding union

3. convSet

   1. Construct an empty list “uList”
   2. Add all of the elements in the set to uList
   3. Terminate yielding uList

4. intersection1

   1. Construct an empty set “intersection”
   2. Add all of the elements in set 1 to intersection
   3. Remove from intersection all of its elements that are not contained in set 2
   4. Remove from intersection all of its elements that are not contained in set 3
   5. Terminate yielding intersection

5. diff

   1. Construct an empty set “difference”
   2. Add all of the elements in the set 1 to difference
   3. Remove from difference all of its elements that are contained in set 2
   4. Terminate yielding difference

6. iterator

```
1. Initialise iterator over the elements of the set
2. While iterator has more elements, repeat:
   2.1. Print out the next element in the iteration
3. Terminate yielding iterator
```

Part B

**Algorithms used in MyBST_Exercise:**

Section 1

1. iterateMyList

```
1. Print out the section title
2. For each element in linked list
   2.1. Print out element
3. Terminate
```

2. checkGoose

```
1. Initialise a variable containing the element
2. If list contains the specified element
   2.1. Print out message saying that list contains the element
3. Else
   3.1. Print out message saying that list does not contain the element
4. Terminate
```

3. insertGoose

```
1. Initialise a variable that is equal to the index of the specified element
2. Add a new element at the specified index(use the variable adding 1 to it)
3. Print out the section title
4. Print out the list
5. Terminate
```

4. notEndingWithBerries

```
1. Initialise a variable (null)
2. For each element in the linked list
   2.1. If element does not contain the specified element
      2.1.1. Assign element to the variable
3. Remove from the linked list the specified element(the variable)
4. Print out the variable
5. Terminate
```

5. convertLinkedListToArray

```
1. Construct an array
2. Print out the section title
3. For each element in the array
   3.1. Print out element
4. Terminate
```

```
6. sortedLinkedList

   1. Sort the linked list into ascending order
   2. Print out the section title
   3. Print out the sorted linked list
   4. Terminate
```

## Section 2

```
1. Initialise a variable “direction” and set it to 0
2. Create an object “parent” of type FruitNode and set it to null
3. Create an object “current” of type FruitNode and assign root to it
4. While looping indefinitely:
   4.1. If the value of current is equal to null:
      4.1.1. Create an object “ins” of type FruidNode using the method’s argument (the element)
      4.1.2. If root is equal to null:
         4.1.2.1. Assign ins to root
      4.1.3. Else:
         4.1.3.1. If direction is less than 0:
            4.1.3.1.1. Assign ins to the left parent
         4.1.3.2. Else:
            4.1.3.2.1. Assign ins to the right parent
      4.1.4. Exit the indefinite loop
   4.2. Compare the current element to the next one assigning it to direction at the same time.
   4.3. If direction is equal to zero
      4.3.1. Exit the indefinite loop
   4.4. Assign the current element to the parent variable
   4.5. If direction is less than zero
      4.5.1. Assign the current element to the left node
   4.6. Else:
      4.6.1. Assign the current element to the right node
5. Terminate
```