Day-10 Assignment Questions:
1. Write a Java program
a. to create a new array list, add some colours (string) and print out the collection.
b. to iterate through all elements in an array list.
c. to insert an element into the array list at the first position.
d. to retrieve an element (at a specified index) from a given array list.
e. to update specific array element by given element.
f. to remove the third element from an array list.
g. to search an element in an array list.
h. to sort a given array list.
i. to copy one array list into another.
j. to shuffle elements in an array list.
□
2. Write a Java program to,
a. append the specified element to the end of a linked list.
b. iterate through all elements in a linked list.
c. iterate through all elements in a linked list starting at the specified position.
d. iterate a linked list in reverse order.
e. insert the specified element at the specified position in the linked list.
f. insert elements into the linked list at the first and last position.
g. insert the specified element at the front of a linked list.
h. insert the specified element at the end of a linked list.
i. insert some elements at the specified position into a linked list.
j. get the first and last occurrence of the specified elements in a linked list.
□
3. Write a Java program to,
a. append the specified element to the end of a hash set.
b. iterate through all elements in a hash list.
c. get the number of elements in a hash set.
d. empty the hash set.
e. test a hash set is empty or not.
f. clone a hash set to another hash set.
g. convert a hash set to an array.
h. convert a hash set to a tree set.
i. convert a hash set to a List/ArrayList.
j. compare two hash set.
□
4. Write a Java program to,
a. create a new tree set, add some colours (string) and print out the tree set.
b. iterate through all elements in a tree set.
c. add all the elements of a specified tree set to another tree set.
d. create a reverse order view of the elements contained in a given tree set.
e. get the first and last elements in a tree set.
f. clone a tree set list to another tree set.
g. get the number of elements in a tree set.
h. compare two tree sets.
i. find the numbers less than 7 in a tree set.
j. get the element in a tree set which is greater than or equal to the given element.
k. get the element in a tree set which is less than or equal to the given element.
l. get the element in a tree set which is strictly greater than or equal to the given element.
m. get an element in a tree set which is strictly less than the given element.
n. retrieve and remove the first element of a tree set.
o. retrieve and remove the last element of a tree set.

p. remove a given element from a tree set.

5. Write a Java program to,
1. create a new priority queue, add some colors (string) and print out the elements of the priority queue.
2. iterate through all elements in priority queue.
3. add all the elements of a priority queue to another priority queue.
4. insert a given element into a priority queue.
5. remove all the elements from a priority queue.
6. count the number of elements in a priority queue.
7. compare two priority queues.
8. retrieve the first element of the priority queue.
9. retrieve and remove the first element.
10. convert a priority queue to an array containing all of the elements of the queue.

6. Write a Java program to,
1. associate the specified value with the specified key in a Tree Map.
2. copy a Tree Map content to another Tree Map.
3. search a key in a Tree Map.
4. search a value in a Tree Map.
5. get all keys from the given a Tree Map.
6. delete all elements from a given Tree Map.
7. sort keys in Tree Map by using comparator.
8. get a key-value mapping associated with the greatest key and the least key in a map.
9. get the first (lowest) key and the last (highest) key currently in a map.
10. get a reverse order view of the keys contained in a given map.

7. Write a Java program to,
1. associate the specified value with the specified key in a HashMap.
2. count the number of key-value (size) mappings in a map.
3. copy all of the mappings from the specified map to another map.
4. remove all of the mappings from a map.
5. check whether a map contains key-value mappings (empty) or not.
6. get a shallow copy of a HashMap instance.
7. test if a map contains a mapping for the specified key.
8. test if a map contains a mapping for the specified value.
9. create a set view of the mappings contained in a map.
10. get the value of a specified key in a map.

8. Develop a Java program to manage a list of bank accounts using ArrayList.
--->Create an Account class with the following attributes:
a. accountNumber (int)
b. holderName (String)
c. balance (double)
--->Use autoboxing to store the balance and interest as Double wrapper objects.
--->Use manual boxing to convert a primitive interest rate into a Double object.
--->Unbox the values (both automatic and manual) to calculate and update the new balance.
--->Add at least three Account objects to an ArrayList<Account>.
--->For each account:
a. Display the holder name, account number, original balance
b. Apply 5% interest
c. Show the new balance using primitive values (unboxed).