Practical 7

1. Implement the method `remove` in Chapter7\array\SortedArrayList class.
Note

anEntry = 3, i = 2, length = 5

0	1	2	3	4
1	2	4	5	6

//Ming Yeu

```
public boolean remove(T anEntry) {
      int i = 0;
      if(!isEmpty() && contains(anEntry)){

            while (i < length && anEntry.compareTo(array[i]) >= 0) {
                  i++;
            }
            removeGap(i + 1);
            length--;
            return true;
      }
       return false;
}
```

Question 2

//Kean Min

Implement the entity classes shown in Figure 1 as follows:

○ The Employee class implements the **Comparable** interface based on id. The Clerk and Manager inherits the **compareTo** method from the superclass Employee.

○ In each class, include constructors, setters, getters and the method toString.

```
public abstract class Employee implements Comparable<Employee>{
      private int id;
      private String name;
      private double basicSalary;

//Constructor
public Employee(){}

public Employee(int id, String name, double basicSalary){
      this.id = id;
      this.name = name;
      this.basicSalary = basicSalary;
```

```
}

public int getId() {
      return id;
}

public void setId(int id) {
      this.id = id;
}

public String getName() {
      return name;
}

public void setName(String name) {
      this.name = name;
}

public double getBasicSalary() {
      return basicSalary;
}

public void setBasicSalary(double basicSalary) {
      this.basicSalary = basicSalary;
}

public abstract double computeMonthlyPay();

public int compareTo(Employee temp){
      return this.getId() - temp.getId();
}

public String toString(){
      return this.id + "\t" + String.format("%-15s", this.name) +
                  "\t"; }
}
```

//Raphael

○ Implement the additional methods in the classes as follows:

■ A clerk’s overtime pay is the overtime hours multiplied by the overtime rate.

```
public double computeOvertime(){

      double overtimePay = overtimeHours * overtimeRate;
      return overtimePay;
```

```
}
```

■ A clerk’s monthly pay is the basic salary plus overtime pay.

```
@Override
public double computeMonthlyPay(){

      double monthlyPay = basicSalary + computeOvertime();
      return monthlyPay;
}
```

■ A manager’s monthly pay is the basic salary plus allowance

```
@Override
public double computeMonthlyPay(){
      double monthlyPay = basicSalary + allowance;
      return monthlyPay;
}
```

//Yin Lam

//Employee

```
@Override
public String toString(){
      return id+ “. ” + this.name + “ - ” + “ RM ”+ basicSalary;
}
```

//Manager

```
@Override
public String toString(){
      return super.toString() + “Allowance ” + “RM ”+ allowance;
}
```

//Clerk

```
@Override
public String toString(){
      return super.toString() + “Overtime Rate : ” + overtimeRate + “ per
hours”;
}
```

Question 3

//Yeu Yang

Implement the method remove in Chapter7\linked\SortedLinkedList class.

```
public boolean remove(T anEntry) {
	int position = 0;
	Node currentNode = firstNode;
	Node previousNode = firstNode;

	while(currentNode != null && currentNode.data.compareTo(anEntry) <
	0){

		previousNode = currentNode;
		currentNode = currentNode.next;
		//position++;
	}

	//firstNode
	if(currentNode != null && currentNode.data.equals(anEntry)){
		if(currentNode == firstNode){ //first node
			firstNode = firstNode.next;
		}else{
			previousNode.next = currentNode.next;
		}

		length--;
		return true;
	}
	return false;
}
```

Question 4 (Continue this in week 10)

//Yong Kang

a. Create a linked implementation of the ADT sorted list which provides an iterator. Define the iterator class as an inner class of the ADT.

```
public interface SortedListWithIteratorInterface<T extends Comparable<T>>
                extends SortedListInterface{

      Iteractor<T> getIterator();
}

public class SortedListWithIterator<T extends Comparable<T>>
                        implements SortedListWithIteratorInterface<T>
{
      private Node firstNode;
      private int length;

      @Override
      public Iterator<T> getIterator()
      {
            return new ListIterator();
      }


      private class ListIterator implements Iterator<T>{
            private Node current;

            @Override
            public T next()
            {
                  T value = null;

                  if(hasNext()){
                        value = current.data;
                        current = current.next;
                  }
                  return value;
            }

            @Override
            public boolean hasNext()
            {
                  return (current != null);
            }
      }
}//end of class
```

b. Write a client program that creates a sorted list of employees with entries comprising manager and clerk objects using the entity classes from part a.

```
//Yin Lam

SortedListWithIterator <Employee> emp = new SortedListWithIterator<>();

emp.add (new Manager(2222, “COl. Sanders”, 5900));

emp.add (new Clerk(3333, “Tony Fey”, 7127));

emp.add (new Clerk(5555, “Jack Bauer”, 10088.88));
```

c. Use the iterator to display the following monthly payroll report:

```
//Wai Kian

Iterator<Employee> it = emp.getIterator();

//print the header
while(it.hasNext()){

	emp = it.next();
	System.out.print(emp + “”);

}
```