Question 01.

```
package Question01;
public class Test {
    public static void main(String[] args) {
        Functions functions =  new Functions();
        functions.addElement();
        functions.view();
    }
}
```

```
package Question01;
public class Functions {
    List numberList = new List(6);
    void addElement(){
        numberList.insertLast(new Number(0, new int[]{0}));
        numberList.insertLast(new Number(1, new int[]{0, 2, 2, 4}));
        numberList.insertLast(new Number(2, new int[]{5}));
        numberList.insertLast(new Number(6, new int[]{5}));
        numberList.insertLast(new Number(7, new int[]{1, 8}));
        numberList.insertLast(new Number(8, new int[]{6}));
    }
    void view(){
        for (int i = 0; i < numberList.listSize(); i++) {
            System.out.println(numberList.retrieveListData(i));
        }
    }
}
```

```
package Question01;
public class List {
    private int maxSize;
    private int position;
    private Number[] listEntry;

    public List(int size) {
        maxSize = size;
        listEntry = new Number[maxSize];
        position = -1;
    }

    public boolean isListEmpty() {
        return (position == -1);
    }

    public boolean isListFull() {
        return (position == maxSize - 1);
    }

    public int listSize() {
        return (position + 1);
    }

    public void insertLast(Number x) {
        if (isListFull()) {
            System.out.println("Error: Attempt to insert at the end of a full
list");
        } else {
            listEntry[++position] = x;
        }
    }
```

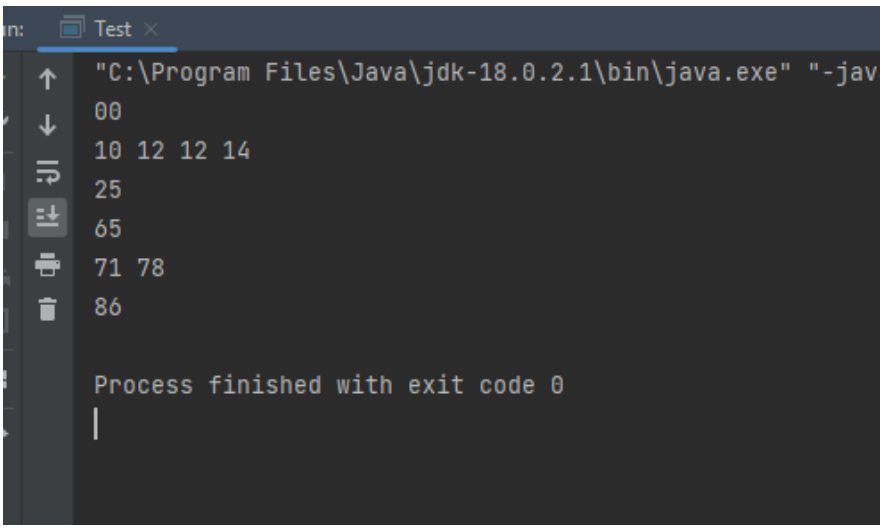
```
    public void insertList(int p, Number element) {
        if (isListFull()) {
            System.out.println("Error: Attempt to insert an entry into a full
list");
        } else if (p < 0 || p > listSize()) {
            System.out.println("Error: Attempt to insert at position " + p + "
which is out of bounds [0, " + listSize() + "]");
        } else {
            for (int i = listSize(); i > p; i--) {
                listEntry[i] = listEntry[i - 1];
            }
            listEntry[p] = element;
            position++;
        }
    }

    public Number deleteList(int p) {
        Number element;
        if (isListEmpty()) {
            System.out.println("Error: Attempt to delete an entry from an empty
list");
            return null;
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Error: Attempt to delete position " + p + "
which is not in the list [0, " + (listSize() - 1) + "]");
            return null;
        } else {
            element = listEntry[p];
            for (int i = p; i < listSize() - 1; i++) {
                listEntry[i] = listEntry[i + 1];
            }
            position--;
```

```
            return element;
        }
    }

    public Number retrieveList(int p) {
        if (isListEmpty()) {
            System.out.println("Error: Attempt to retrieve an entry from an
empty list");
            return null;
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Error: Attempt to retrieve entry at position "
+ p + " which is not in the list [0, " + (listSize() - 1) + "]");
            return null;
        } else {
            return listEntry[p];
        }
    }

    public void replaceList(int p, Number x) {
        if (isListEmpty()) {
            System.out.println("Error: Attempt to replace an entry of an empty
list");
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Error: Attempt to replace entry at position " +
p + " which is not in the list [0, " + (listSize() - 1) + "]");
        } else {
            listEntry[p] = x;
        }
    }

    public void traverseList() {
        if (isListEmpty()) {
            System.out.println("List is empty.");
```

```
            return;
        }
        System.out.print("List: [");
        for (int i = 0; i < listSize(); i++) {
            System.out.print(listEntry[i]);
            if (i < listSize() - 1) {
                System.out.print(", ");
            }
        }
        System.out.println("]");
    }

    public void clearList() {
        position = -1;
    }

    public int[] getInternalArrayCopy() {
        if (isListEmpty()) {
            return new int[0];
        }
        int[] copy = new int[listSize()];
        System.arraycopy(listEntry, 0, copy, 0, listSize());
        return copy;
    }

    public String retrieveListData(int p) {
        if (isListEmpty()) {
            System.out.println("Error: Attempt to retrieve an entry from an
empty list");

        } else if (p < 0 || p >= listSize()) {
```

```
            System.out.println("Error: Attempt to retrieve entry at position "
+ p + " which is not in the list [0, " + (listSize() - 1) + "]");


        } else {
            return listEntry[p].viewNumbers();
        }


        return null;
    }
}
```

```java
package Question01;

public class Number {

    int stem;

    int[] leaves;

    public Number(int stem, int[] leaves) {

        this.stem = stem;

        this.leaves = leaves;

    }


    String viewNumbers(){

        for (int leaf:leaves ) {

            System.out.print(stem + ""+leaf + " ");

        }

        return " ";

    }

}
```



s

Question 02.

```
package Question02;

public class Test {

    public static void main(String[] args) {


        Functions function = new Functions();

        function.addData();

        function.bestProductOfRgionNorth();

        function.bestProductOfRgionEast();

        function.bestProductOfRgionSouth();

        function.bestofQuarter();

    }

}
```

```
 Total Best All Quarter Sale :   B - 13361
kavindus@kavindus-MacBook-Air BECS-21223-Data-Structures-and-Algorithms-LAB-05 %
Structures-and-Algorithms-LAB-05 ; /usr/bin/env /Library/Java/JavaVirtualMachine
va --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/kavindus/
r/workspaceStorage/ecd30729660bc53e03f369dcb91994f0/redhat.java/jdt_ws/BECS-2122
05_431795d7/bin Question02.Test
 North  B - 5678
 East   C - 6983
 South  B - 4892


 Total Best All Quarter Sale :   B - 13361
kavindus@kavindus-MacBook-Air BECS-21223-Data-Structures-and-Algorithms-LAB-05 %
```

```java
package Question02;

public class Products {

    String name;

    int north;

    int south;

    int east;

    int totalSales;


    public Products(String name, int north, int south, int east) {

        this.name = name;

        this.north = north;

        this.south = south;

        this.east = east;

        this.totalSales = north + east +south;

    }


}
```

```
package Question02;

public class List {
    private int maxSize;
    private int position;
    private Products[] listEntry;

    public List(int size) {
        maxSize = size;
        listEntry = new Products[maxSize];
        position = -1;
    }

    public boolean isListEmpty() {
        return (position == -1);
    }

    public boolean isListFull() {
        return (position == maxSize - 1);
    }

    public int listSize() {
        return (position + 1);
    }

    public void insertLast(Products x) {
        if (isListFull()) {
            System.out.println("Error: Attempt to insert at the end of a full
list");
        } else {
            listEntry[++position] = x;
        }}
```

```
    public void insertList(int p, Products element) {
        if (isListFull()) {
            System.out.println("Error: Attempt to insert an entry into a full
list");
        } else if (p < 0 || p > listSize()) {
            System.out.println("Error: Attempt to insert at position " + p + "
which is out of bounds [0, " + listSize() + "]");
        } else {
            for (int i = listSize(); i > p; i--) {
                listEntry[i] = listEntry[i - 1];
            }
            listEntry[p] = element;
            position++;
        }
    }
    public Products deleteList(int p) {
        Products element;
        if (isListEmpty()) {
            System.out.println("Error: Attempt to delete an entry from an empty
list");
            return null;
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Error: Attempt to delete position " + p + "
which is not in the list [0, " + (listSize() - 1) + "]");
            return null;
        } else {
            element = listEntry[p];
            for (int i = p; i < listSize() - 1; i++) {
                listEntry[i] = listEntry[i + 1];
            }
            position--;
            return element;
        }}
```

```
    public Products retrieveList(int p) {
        if (isListEmpty()) {
            System.out.println("Error: Attempt to retrieve an entry from an
empty list");
            return null;
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Error: Attempt to retrieve entry at position "
+ p + " which is not in the list [0, " + (listSize() - 1) + "]");
            return null;
        } else {
            return listEntry[p];
        }
    }

    public void replaceList(int p, Products x) {
        if (isListEmpty()) {
            System.out.println("Error: Attempt to replace an entry of an empty
list");
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Error: Attempt to replace entry at position " +
p + " which is not in the list [0, " + (listSize() - 1) + "]");
        } else {
            listEntry[p] = x;
        }
    }
```

```
    public void traverseList() {
        if (isListEmpty()) {
            System.out.println("List is empty.");
            return;
        }
        System.out.print("List: [");
        for (int i = 0; i < listSize(); i++) {
            System.out.print(listEntry[i]);
            if (i < listSize() - 1) {
                System.out.print(", ");
            }
        }
        System.out.println("]");
    }

    public void clearList() {
        position = -1;
    }

    public int[] getInternalArrayCopy() {
        if (isListEmpty()) {
            return new int[0];
        }
        int[] copy = new int[listSize()];
        System.arraycopy(listEntry, 0, copy, 0, listSize());
        return copy;
    }
```

```
    public String retrieveListData(int p) {

        if (isListEmpty()) {

            System.out.println("Error: Attempt to retrieve an entry from an
empty list");


        } else if (p < 0 || p >= listSize()) {

            System.out.println("Error: Attempt to retrieve entry at position "
+ p + " which is not in the list [0, " + (listSize() - 1) + "]");


        } else {

            return listEntry[p].name;

        }


        return null;

    }

}
```

```
package Question02;
public class Functions {
    List list = new List(4);
    void addData(){
        Products productA = new Products("A", 1450, 467, 3800);
        Products productB = new Products("B", 5678, 4892, 2791);
        Products productC = new Products("C", 2987, 270, 6983);
        Products productD = new Products("D", 829, 1500, 29);
        list.insertLast(productA);
        list.insertLast(productB);
        list.insertLast(productC);
        list.insertLast(productD);
    }

    String bestProductOfRgionNorth(){
        String productName = null;
        int initia = 0;
        for (int i = 0; i < 4; i++) {
            if (list.retrieveList(i).north > initia){
                initia = list.retrieveList(i).north;
                productName = list.retrieveListData(i);
            }
        }
        System.out.println(" North \t" +productName+" - " + initia);
       return productName;
    }
```

```
    String bestProductOfRgionSouth(){
        String productName = null;
        int initia = 0;
        for (int i = 0; i < 4; i++) {
            if (list.retrieveList(i).south > initia){
                initia = list.retrieveList(i).south;
                productName = list.retrieveListData(i);
            }
        }
        System.out.println(" South \t" +productName + " - " + initia);
       return productName;
    }
    String bestProductOfRgionEast(){
        String productName = null;
        int initia = 0;
        for (int i = 0; i < 4; i++) {
            if (list.retrieveList(i).east > initia){
                initia = list.retrieveList(i).east;
                productName = list.retrieveListData(i);
            }
        }
        System.out.println(" East \t" +productName+" - " + initia);
       return productName;
    }
```

```
    String bestofQuarter(){
        String productName = null;
        int initia = 0;
        for (int i = 0; i < 4; i++) {
            if (list.retrieveList(i).totalSales > initia){
                initia = list.retrieveList(i).totalSales;
                productName = list.retrieveListData(i);
            }
        }
        System.out.println(" \n\nTotal Best All Quarter Sale :\t"
+productName+" - " + initia);
       return productName;
    }


}
```