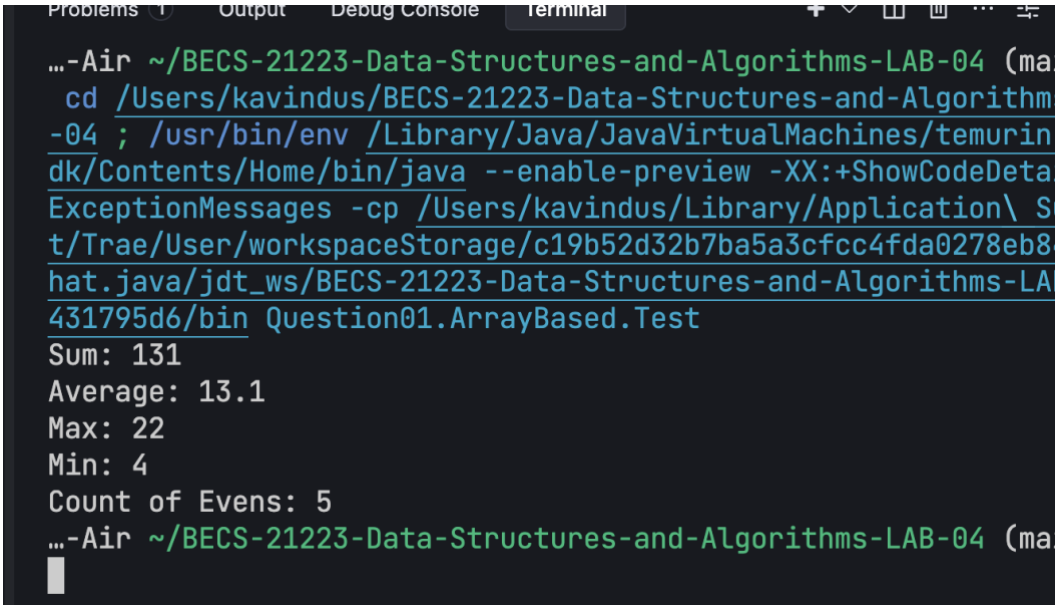
Question 01.

```
package Question01.ArrayBased;
public class Functions {
    List list01 = new List(10);
    public Functions(int[] numberList) {
        for (int i: numberList) {
            list01.insertLast(i);
        }
    }

    int Sum(){
        int sum = 0;
        for(int j=0 ; j<list01.listSize(); j++){
            sum+= list01.retrieveList(j);
        }
        return sum;
    }

    double Avg(){
        return (double) Sum()/list01.listSize();
    }

    int Max(){
        int max=Integer.MIN_VALUE;
        for (int i = 0; i <list01.listSize(); i++) {
            if (max<= list01.retrieveList(i))
                max = list01.retrieveList(i);
        }
        return max;
    }
```

```
    int Min(){
        int min=Integer.MAX_VALUE;
        for (int i = 0; i <list01.listSize(); i++) {
            if (min>= list01.retrieveList(i))
                min = list01.retrieveList(i);

        }
        return min;
    }

    int EvvenCount(){
        int count=0;
        for (int i = 0; i <list01.listSize(); i++) {
            if (list01.retrieveList(i)%2==0)
                count++;
        }
        return count;
    }
}
```

```java
package Question01.ArrayBased;

public class Test {

    public static void main(String[] args) {

        Functions functions = new Functions(new int[]{15, 22, 7, 13, 8, 6, 19,
20, 4, 17});

        System.out.println("Sum: "+functions.Sum());

        System.out.println("Average: "+functions.Avg());

        System.out.println("Max: "+functions.Max());

        System.out.println("Min: "+functions.Min());

        System.out.println("Count of Evens: "+functions.EvvenCount());

    }

}
```



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

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

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

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

    int listSize() {
        return position + 1;
    }

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

```
    void insertList(int p, int element) {
        if (isListFull())
            System.out.println("Attempt to insert into a full list");
        else if (p < 0 || p > listSize())
            System.out.println("Invalid position for insertion");
        else {
            for (int i = position; i > p; i--)
                listEntry[i] = listEntry[i-1];
            listEntry[p] = element;
            position++;
        }
    }

    int deleteList(int p) {
        int element;
        if (isListEmpty()) {
            System.out.println("Attempt to delete from an empty list");
            return 0;
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Invalid position for deletion");
            return 0;
        } else {
            element = listEntry[p];
            for (int i = p; i < position; i++)
                listEntry[i] = listEntry[i+1];
            position--;
            return element;
        }
    }
```

```
    int retrieveList(int p) {
        if (isListEmpty()) {
            System.out.println("Attempt to retrieve from an empty list");
            return 0;
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Invalid position for retrieval");
            return 0;
        } else {
            return listEntry[p];
        }
    }

    void replaceList(int p, int x) {
        if (isListEmpty()) {
            System.out.println("Attempt to replace in an empty list");
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Invalid position for replacement");
        } else {
            listEntry[p] = x;
        }
    }

    void traverseList() {
        for (int i = 0; i <= position; i++)
            System.out.println(listEntry[i]);
    }
}
```

Question 02.

```
package Question02.ArrayBased.Primitive;
import java.util.ArrayList;
import java.util.Arrays;

public class Functions {
    List ballList = new List(6);
    List scoreList = new List(6);

    ArrayList<String> nameList = new ArrayList<String>();

    public Functions(String[] NameList,int[] BallList,int[] ScoreList) {
        for (int i: BallList) {
            ballList.insertLast(i);
        }
        for (int i: ScoreList) {
            scoreList.insertLast(i);
        }

        for (String j : NameList
             ) {
this.nameList.add(j);
        }
    }

    public String HighestScorePlayer(){
        int pointer = 0;
        int maxScore = Integer.MIN_VALUE;
        for (int i = 0; i < scoreList.listSize(); i++) {
            if (maxScore<=scoreList.retrieveList(i)){
```

```
                maxScore = scoreList.retrieveList(i);
                pointer = i;
            }
        }
        return nameList.get(pointer);
    }
    public String LowestNumberOfBalls(){
        int pointer = 0;
        int lowBalls = Integer.MAX_VALUE;
        for (int i = 0; i < ballList.listSize(); i++) {
            if (lowBalls>=scoreList.retrieveList(i)){
                lowBalls = scoreList.retrieveList(i);
                pointer = i;
            }
        }
        return nameList.get(pointer);
    }
    int BattingPointer ;
    public void displayBattingStrike(){
        double HighbattingStrikeRate = 0;
        double Pointer = 0;

        for (int i = 0; i < ballList.listSize(); i++) {
            double currentStrikerate =
calculateStrikeRate(scoreList.retrieveList(i), ballList.retrieveList(i));
            System.out.printf("Player "+nameList.get(i)+ " : %.2f\n",
currentStrikerate);
            if (HighbattingStrikeRate <= currentStrikerate){
                HighbattingStrikeRate = currentStrikerate;
                Pointer = i;
            }
        this.BattingPointer = (int) Pointer;
```

```
        }
    }


    public String ManOfTheMatch(){
        return nameList.get(BattingPointer);
    }
    public static double calculateStrikeRate(int battingScore, int ballsFaced)
{
        if (ballsFaced == 0) {
            System.out.println("Error: Balls faced cannot be zero.");
            return 0.0;
        }
        return (double) battingScore * 100 / ballsFaced;
    }
}
```

```
-04 ; /usr/bin/env /Library/Java/JavaVirtualMachir
dk/Contents/Home/bin/java --enable-preview -XX:+Sh
ExceptionMessages -cp /Users/kavindus/Library/Appl
t/Trae/User/workspaceStorage/c19b52d32b7ba5a3cfcc4
hat.java/jdt_ws/BECS-21223-Data-Structures-and-Alg
431795d6/bin Question02.ArrayBased.Primitive.Test
Chamari Athapaththu
Hasini Perera
STRIKE RATES
Player Chamari Athapaththu : 131.25
Player Anushka Sanjeewani : 109.09
Player Kavisha Dilhari : 94.44
Player Harshitha Samarawickrama : 92.31
Player Sugandika Kumari : 111.11
Player Hasini Perera : 100.00
MOM Chamari Athapaththu
…-Air ~/BECS-21223-Data-Structures-and-Algorithms-
```

```
package Question02.ArrayBased.Primitive;
public class Test {
    public static void main(String[] args) {
        Functions functions = new Functions(
                new String[] {
                        "Chamari Athapaththu",
                        "Anushka Sanjeewani",
                        "Kavisha Dilhari",
                        "Harshitha Samarawickrama",
                        "Sugandika Kumari",
                        "Hasini Perera"
                },new int[] {48, 22, 18, 13, 9, 3},
                new int[] {63, 24, 17, 12, 10, 3}

        );
// nama, bola, lakunu danna
        System.out.println(functions.HighestScorePlayer());
        System.out.println(functions.LowestNumberOfBalls());
        System.out.println("STRIKE RATES");
        functions.displayBattingStrike();
        System.out.println("MOM "+functions.ManOfTheMatch());




    }
}
```

```
package Question02.ArrayBased.Primitive;
public class List {
    private int maxSize;
    private int position;
    private int[] listEntry;

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

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

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

    int listSize() {
        return position + 1;
    }

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

```
    void insertList(int p, int element) {
        if (isListFull())
            System.out.println("Attempt to insert into a full list");
        else if (p < 0 || p > listSize())
            System.out.println("Invalid position for insertion");
        else {
            for (int i = position; i > p; i--)
                listEntry[i] = listEntry[i-1];
            listEntry[p] = element;
            position++;
        }
    }

    int deleteList(int p) {
        int element;
        if (isListEmpty()) {
            System.out.println("Attempt to delete from an empty list");
            return 0;
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Invalid position for deletion");
            return 0;
        } else {
            element = listEntry[p];
            for (int i = p; i < position; i++)
                listEntry[i] = listEntry[i+1];
            position--;
            return element;
        }
    }
```

```
    int retrieveList(int p) {
        if (isListEmpty()) {
            System.out.println("Attempt to retrieve from an empty list");
            return 0;
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Invalid position for retrieval");
            return 0;
        } else {
            return listEntry[p];
        }
    }
    void replaceList(int p, int x) {
        if (isListEmpty()) {
            System.out.println("Attempt to replace in an empty list");
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Invalid position for replacement");
        } else {
            listEntry[p] = x;
        }
    }

    void traverseList() {
        for (int i = 0; i <= position; i++)
            System.out.println(listEntry[i]);
    }
}
```

Question 03.

```
package Question03;
public class Functions {

    private List partyList;

    public Functions(List partyList) {
        this.partyList = partyList;
    }

    public void findWinningPartiesPerDistrict() {
        if (partyList.isListEmpty()) {
            System.out.println("No party data available.");
            return;
        }

        String[] districts = {"Gampaha", "Colombo", "Kalutara"};
        for (int i = 0; i < districts.length; i++) {
            Party winner = null;
            int maxVotes = -1;
            for (int j = 0; j < partyList.listSize(); j++) {
                Party currentParty = partyList.retrieveList(j);
                int currentVotes = 0;
                if (i == 0) currentVotes = currentParty.getGampahaVotes();
                else if (i == 1) currentVotes = currentParty.getColomboVotes();
                else currentVotes = currentParty.getKalutaraVotes();

                if (currentVotes > maxVotes) {
                    maxVotes = currentVotes;
                    winner = currentParty;
```

```
                }
            }
            if (winner != null) {
                System.out.println("Winning party in " + districts[i] + ": " +
winner.getName() + " (" + maxVotes + " votes)");
            }
        }
    }

    public void findOverallWinner() {
        if (partyList.isListEmpty()) {
            System.out.println("No party data available.");
            return;
        }
        Party overallWinner = null;
        int maxTotalVotes = -1;
        for (int i = 0; i < partyList.listSize(); i++) {
            Party currentParty = partyList.retrieveList(i);
            if (currentParty.getTotalVotes() > maxTotalVotes) {
                maxTotalVotes = currentParty.getTotalVotes();
                overallWinner = currentParty;
            }
        }
        if (overallWinner != null) {
            System.out.println("Overall Provincial Election Winner: " +
overallWinner.getName() + " (" + maxTotalVotes + " total votes)");
        }
    }

    public void findMinVotesPerDistrict() {
        if (partyList.isListEmpty()) {
            System.out.println("No party data available.");
```

```
            return;
        }
        String[] districts = {"Gampaha", "Colombo", "Kalutara"};
        for (int i = 0; i < districts.length; i++) {
            Party loser = null;
            int minVotes = Integer.MAX_VALUE;
            for (int j = 0; j < partyList.listSize(); j++) {
                Party currentParty = partyList.retrieveList(j);
                int currentVotes = 0;
                if (i == 0) currentVotes = currentParty.getGampahaVotes();
                else if (i == 1) currentVotes = currentParty.getColomboVotes();
                else currentVotes = currentParty.getKalutaraVotes();

                if (currentVotes < minVotes) {
                    minVotes = currentVotes;
                    loser = currentParty;
                }
            }
            if (loser != null) {
                System.out.println("Party with minimum votes in " +
districts[i] + ": " + loser.getName() + " (" + minVotes + " votes)");
            }
        }
    }

    public void findEligibleParties() {
        if (partyList.isListEmpty()) {
            System.out.println("No party data available.");
            return;
        }
        System.out.println("Eligible parties (more than 100,000 total
votes):");
```

```
        boolean found = false;
        for (int i = 0; i < partyList.listSize(); i++) {
            Party currentParty = partyList.retrieveList(i);
            if (currentParty.getTotalVotes() > 100000) {
                System.out.println("- " + currentParty.getName() + " (" +
currentParty.getTotalVotes() + " total votes)");
                found = true;
            }
        }
        if (!found) {
            System.out.println("No parties are eligible for a seat based on
this criterion.");
        }
    }
}
```

```
package Question03;
public class List {
```

```
    private int maxSize;
    private int position;
    private Party[] listEntry;

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

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

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

    int listSize() {
        return position + 1;
    }

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

    void insertList(int p, Party element) {
        if (isListFull())
```

```
            System.out.println("Attempt to insert into a full list");
        else if (p < 0 || p > listSize())
            System.out.println("Invalid position for insertion");
        else {
            for (int i = position; i >= p; i--) // Corrected loop condition
                listEntry[i + 1] = listEntry[i];
            listEntry[p] = element;
            position++;
        }
    }

    Party deleteList(int p) {
        Party element;
        if (isListEmpty()) {
            System.out.println("Attempt to delete from an empty list");
            return null;
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Invalid position for deletion");
            return null;
        } else {
            element = listEntry[p];
            for (int i = p; i < position; i++)
                listEntry[i] = listEntry[i + 1];
            position--;
            return element;
        }
    }

    Party retrieveList(int p) {
        if (isListEmpty()) {
            System.out.println("Attempt to retrieve from an empty list");
```

```
            return null;
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Invalid position for retrieval");
            return null;
        } else {
            return listEntry[p];
        }
    }

    void replaceList(int p, Party x) {
        if (isListEmpty()) {
            System.out.println("Attempt to replace in an empty list");
        } else if (p < 0 || p >= listSize()) {
            System.out.println("Invalid position for replacement");
        } else {
            listEntry[p] = x;
        }
    }

    void traverseList() {
        for (int i = 0; i <= position; i++)
            System.out.println(listEntry[i].toString());
    }
}
```

```
package Question03;
public class Party {
    private String name;
    private int gampahaVotes;
```

```
    private int colomboVotes;
    private int kalutaraVotes;

    public Party(String name, int gampahaVotes, int colomboVotes, int
kalutaraVotes) {
        this.name = name;
        this.gampahaVotes = gampahaVotes;
        this.colomboVotes = colomboVotes;
        this.kalutaraVotes = kalutaraVotes;
    }

    public String getName() {
        return name;
    }

    public int getGampahaVotes() {
        return gampahaVotes;
    }

    public int getColomboVotes() {
        return colomboVotes;
    }

    public int getKalutaraVotes() {
        return kalutaraVotes;
    }

    public int getTotalVotes() {
        return gampahaVotes + colomboVotes + kalutaraVotes;
    }

    public String toString() {
```

```
        return "Party: " + name + ", Gampaha: " + gampahaVotes + ", Colombo: "
+ colomboVotes + ", Kalutara: " + kalutaraVotes + ", Total: " +
getTotalVotes();
    }
}
```

```
package Question03;

public class Test {
    public static void main(String[] args) {
```

```
        List electionData = new List(5);

        electionData.insertLast(new Party("A", 12453, 89023, 60250));
        electionData.insertLast(new Party("B", 23457, 41900, 35890));
        electionData.insertLast(new Party("C", 74129, 23000, 56000));
        electionData.insertLast(new Party("D", 202, 57, 354));
        electionData.insertLast(new Party("E", 87, 5, 457));

        Functions analysis = new Functions(electionData);
        analysis.findWinningPartiesPerDistrict();
        analysis.findOverallWinner();
        analysis.findMinVotesPerDistrict();
         analysis.findEligibleParties();

    }
}
```

```
dk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetails
ExceptionMessages -cp /Users/kavindus/Library/Application\ Supp
t/Trae/User/workspaceStorage/c19b52d32b7ba5a3cfcc4fda0278eb84/r
hat.java/jdt_ws/BECS-21223-Data-Structures-and-Algorithms-LAB-0
431795d6/bin Question03.Test
Winning party in Gampaha: C (74129 votes)
Winning party in Colombo: A (89023 votes)
Winning party in Kalutara: A (60250 votes)
Overall Provincial Election Winner: A (161726 total votes)
Party with minimum votes in Gampaha: E (87 votes)
Party with minimum votes in Colombo: E (5 votes)
Party with minimum votes in Kalutara: D (354 votes)
Eligible parties (more than 100,000 total votes):
- A (161726 total votes)
- B (101247 total votes)
- C (153129 total votes)
…-Air ~/BECS-21223-Data-Structures-and-Algorithms-LAB-04 (main)
```