Assignment 1: Collections
SET A

A1) Write a java program to accept names of ‘n’ cities, insert same into array list collection and display the contents of same array list, also remove all these elements.

```
import java.util.*;
public class A1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        ArrayList al = new ArrayList();

        System.out.println("Enter How many cities :");
        int n = sc.nextInt();

        System.out.println("Enter the Cities :");
        sc.nextLine();
        for (int i = 0; i < n; i++) {
            String c = sc.nextLine();
            al.add(c);
        }
        System.out.println("Cities :" + al);

        System.out.println("ArrayList after removing the elements  :");
        al.clear();

        sc.close();
    }
}
```

Program 2
A2. Write a java program to read ‘n’ names of your friends, store it into linked list, also display contents of the same.

```
import java.util.*;
public class A2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        LinkedList ll = new LinkedList();
        System.out.println("Enter How many Friends :");
        int n = sc.nextInt();

        System.out.println("Enter the "+n+" Friends :");
        sc.nextLine();
        for (int i = 0; i < n; i++) {
            String fl = sc.nextLine();
            ll.add(fl);
        }
        System.out.println("Friends :" + ll);
        sc.close();
    }
}
```

Program 3

A3. Write a program to create a new tree set, add some colors (string) and print out the tree set.

```
import java.util.Scanner;
import java.util.TreeSet;
public class A3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        TreeSet ts = new TreeSet();

        System.out.println("Enter How many Colours :");
        int n = sc.nextInt();

        System.out.println("Enter the "+n+" Colours :");
        sc.nextLine();
        for (int i = 0; i < n; i++) {
            String c = sc.nextLine();
            ts.add(c);
        }
        System.out.println("Colours :" + ts);
        sc.close();
    }
}
```

Program 4

A4. Create the hash table that will maintain the mobile number and student name. Display the contact list.

```
import java.util.Hashtable;
public class A4 {
   public static void main(String[] args) {
      Hashtable<String, String> hashtable = new Hashtable<String, String>();
      hashtable.put("Prasad", "8796465800");
      hashtable.put("Ashish", "8806503414");
      hashtable.put("Suhani", "8629913414");
      hashtable.put("Sanket", "7118919895");

      System.out.println(hashtable);
   }
}
```

Set B

B1. Accept 'n' integers from the user. Store and display integers in sorted order having proper collection class. The collection should not accept duplicate elements.

```
import java.util.TreeSet;

import java.util.Scanner;

public class B1 {

   public static void main(String[] args)

{

      Scanner sc = new Scanner(System.in);

      TreeSet<Object> ts = new TreeSet<>();



      System.out.println("Enter how many Numbers: ");

      int n = sc.nextInt();
```

```
        System.out.println("Enter the " + n + " Numbers: ");

        for (int i = 0; i < n; i++) {

            int num = sc.nextInt();

            ts.add(num);

        }

        System.out.println("Numbers in Sorted Order and without Duplication :" + ts);

        sc.close();

    }

}
```

B2. Write a program to sort HashMap by keys and display the details before sorting and after sorting.

```
import java.util.HashMap;

import java.util.TreeMap;

public class B2 {

    public static void main(String[] args) {

        HashMap<String, Integer> map = new HashMap<>();

        map.put("Prasad", 2002);

        map.put("Ashish", 2001);

        map.put("Suhas", 2002);

        map.put("Swayam", 2001);

        map.put("Sanket", 2002);

        System.out.println("\nHashMap Details Before Sorting :\n" + map);

        TreeMap<Object,Object> tm = new TreeMap<>(map);

        System.out.println("\nHashMap Details After Sorting :\n" + tm);
    }
}
```

B3. Write a program that loads names and phone numbers from a text file where the data is organized as one line per record and each field in a record are separated by a tab (\t)or(:).it takes a name or phone number as input and prints the corresponding other value from the hash table (hint: use hash tables). (For file content use B3.txt)

```
import java.io.*;
import java.util.Hashtable;
import java.util.Scanner;
public class B3 {
    public static void main(String[] args) {
        try {
            File f = new File("B3.txt");
            BufferedReader br = null;
            br = new BufferedReader(new FileReader(f));
            Hashtable<String, String> table = new Hashtable<>();
            Scanner sc = new Scanner(System.in);
            String line = "";
            while ((line = br.readLine()) != null) {
                String[] parts = line.split(":");
                String name = parts[0].trim();
                String number = parts[1].trim();
                if (!name.equals("") && !number.equals("")) {
                    table.put(name, number);
                }
            }
            System.out.println("Enter Name :");
            String key = sc.nextLine();

            if (table.containsKey(key)) {
                System.out.println(table.get(key));
                br.close();
                sc.close();
            }
        } catch (Exception e) {
            System.out.println(e);
        }

    }
}
```

B3.txt

```
ABC:8796465800
XYZ:9876543286
AQZ:78654324679
RMD:9087654325
```

Set C
a) Create a java application to store city names and their STD codes using an appropriate collection. The GUI should allow the following operations:
i. Add a new city and its code (No duplicates)
ii. Remove a city from the collection
iii. Search for a city name and display the code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.HashMap;

public class CitySTDDirectory extends JFrame {

    private HashMap<String, String> cityMap = new HashMap<>();

    private JTextField cityField;
    private JTextField codeField;
    private JTextArea displayArea;

    public CitySTDDirectory() {
        setTitle("City STD Code Directory");
        setSize(400, 350);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // UI Components
        cityField = new JTextField(15);
        codeField = new JTextField(15);
        displayArea = new JTextArea(10, 30);
        displayArea.setEditable(false);

        JButton addButton = new JButton("Add City");
        JButton removeButton = new JButton("Remove City");
        JButton searchButton = new JButton("Search City");

        // Panel for text fields
        JPanel inputPanel = new JPanel();
        inputPanel.setLayout(new GridLayout(3, 2, 5, 5));
        inputPanel.add(new JLabel("City Name:"));
        inputPanel.add(cityField);
        inputPanel.add(new JLabel("STD Code:"));
        inputPanel.add(codeField);

        // Panel for buttons
        JPanel buttonPanel = new JPanel();
        buttonPanel.add(addButton);
        buttonPanel.add(removeButton);
        buttonPanel.add(searchButton);

        // Add listeners
        addButton.addActionListener(e -> addCity());
        removeButton.addActionListener(e -> removeCity());
        searchButton.addActionListener(e -> searchCity());

        // Layout setup
        setLayout(new BorderLayout());
        add(inputPanel, BorderLayout.NORTH);
        add(buttonPanel, BorderLayout.CENTER);
        add(new JScrollPane(displayArea), BorderLayout.SOUTH);
    }

    private void addCity() {
        String city = cityField.getText().trim();
        String code = codeField.getText().trim();

        if (city.isEmpty() || code.isEmpty()) {
            displayArea.setText("City or code cannot be empty!");
            return;
        }
        if (cityMap.containsKey(city)) {
            displayArea.setText("City already exists. Duplicate not allowed!");
        } else {
            cityMap.put(city, code);
            displayArea.setText("Added: " + city + " -> " + code);
        }
    }
    private void removeCity() {
        String city = cityField.getText().trim();

        if (city.isEmpty()) {
            displayArea.setText("Enter a city to remove!");
            return;
        }
        if (cityMap.remove(city) != null) {
            displayArea.setText("Removed city: " + city);
        } else {
            displayArea.setText("City not found!");
        }
    }

    private void searchCity() {
        String city = cityField.getText().trim();

        if (city.isEmpty()) {
            displayArea.setText("Enter a city to search!");
            return;
        }
        String code = cityMap.get(city);
        if (code != null) {
            displayArea.setText("STD Code for " + city + " = " + code);
        } else {
            displayArea.setText("City not found!");
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new CitySTDDirectory().setVisible(true));
    }
}
```

b) Write a program to create link list of integer objects. Do the following:
i. add element at first position
ii. delete last element
iii. display the size of link list

```
import java.util.LinkedList;

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

        // Create linked list of Integer objects
        LinkedList<Integer> list = new LinkedList<>();

        // --- i. Add element at first position ---
        list.addFirst(50);
        list.addFirst(30);
        list.addFirst(10);

        System.out.println("Linked List after adding elements at first: " + list);

        // --- ii. Delete last element ---
        if (!list.isEmpty()) {
            int removed = list.removeLast();
            System.out.println("Removed last element: " + removed);
        } else {
            System.out.println("List is empty, nothing to remove!");
        }

        // --- iii. Display the size of linked list ---
        System.out.println("Size of Linked List: " + list.size());
    }
}
```

c) Read a text file, specified by the first command line argument, into a list. The program should then display a menu which performs the following operations on the list:
1. Insert line   2. Delete line   3. Append line   4. Modify line   5. Exit
When the user selects Exit, save the contents of the list to the file and end the program.

```
import java.io.*;
import java.util.*;

public class TextFileEditor {

    public static void main(String[] args) {

        if (args.length < 1) {
            System.out.println("Usage: java TextFileEditor <filename>");
            return;
        }
```

```
String filename = args[0];
List<String> lines = new ArrayList<>();

// ---- Read file into list ----
try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
    String line;
    while ((line = br.readLine()) != null) {
        lines.add(line);
    }
} catch (IOException e) {
    System.out.println("Error reading file: " + e.getMessage());
    return;
}

Scanner sc = new Scanner(System.in);
int choice;

// ---- Menu Loop ----
do {
    System.out.println("\nMENU:");
    System.out.println("1. Insert line");
    System.out.println("2. Delete line");
    System.out.println("3. Append line");
    System.out.println("4. Modify line");
    System.out.println("5. Exit");
    System.out.print("Enter your choice: ");

    choice = sc.nextInt();
    sc.nextLine(); // consume newline

    switch (choice) {

        case 1:  // Insert line
            System.out.print("Enter line number to insert at (1-based): ");
            int insertPos = sc.nextInt();
            sc.nextLine();
            System.out.print("Enter the new line: ");
            String newLine = sc.nextLine();

            if (insertPos >= 1 && insertPos <= lines.size()+1) {
                lines.add(insertPos - 1, newLine);
                System.out.println("Line inserted.");
            } else {
                System.out.println("Invalid line number!");
    }
    break;

case 2:  // Delete line
    System.out.print("Enter line number to delete: ");
    int delPos = sc.nextInt();
    sc.nextLine();

    if (delPos >= 1 && delPos <= lines.size()) {
        lines.remove(delPos - 1);
        System.out.println("Line deleted.");
    } else {
        System.out.println("Invalid line number!");
    }
    break;

case 3:  // Append line
    System.out.print("Enter line to append: ");
    String appendLine = sc.nextLine();
    lines.add(appendLine);
    System.out.println("Line appended.");
    break;

case 4:  // Modify line
    System.out.print("Enter line number to modify: ");
    int modPos = sc.nextInt();
    sc.nextLine();

    if (modPos >= 1 && modPos <= lines.size()) {
        System.out.println("Current text: " + lines.get(modPos - 1));
        System.out.print("Enter new text: ");
        String modified = sc.nextLine();
        lines.set(modPos - 1, modified);
        System.out.println("Line modified.");
    } else {
        System.out.println("Invalid line number!");
    }
    break;

case 5:
    System.out.println("Saving changes and exiting...");
    break;

default:
```

```
                System.out.println("Invalid choice!");
            }

        } while (choice != 5);

        // ---- Save list back to file ----
        try (PrintWriter pw = new PrintWriter(new FileWriter(filename))) {
            for (String s : lines) {
                pw.println(s);
            }
        } catch (IOException e) {
            System.out.println("Error writing file: " + e.getMessage());
        }

        System.out.println("File updated successfully.");
    }
}
```