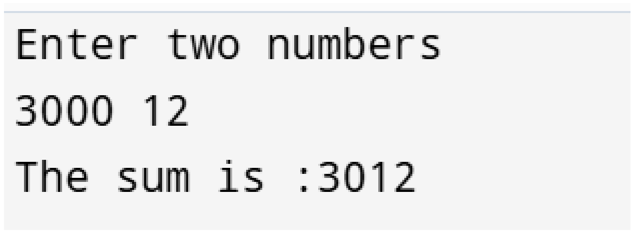
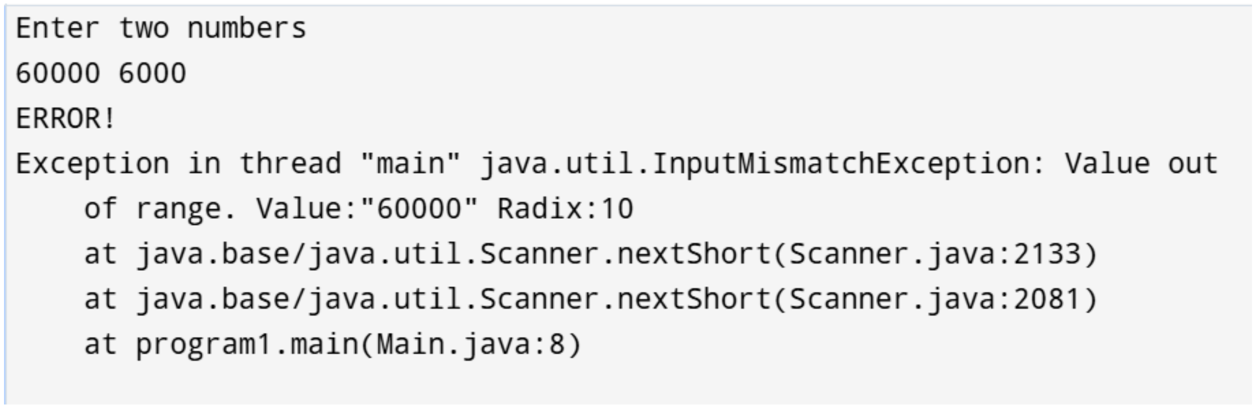
QUESTION 1:
Write a program to accept two short integers from the user and display the sum. Check what happens when the sum exceeds the maximum range of short.

CODE:
```
import java.util.*;
public class program1 {
        public static void main(String[] args) {
                Scanner ob = new Scanner(System.in);
                short a, b;
                System.out.println("Enter two numbers");
                a = ob.nextShort();
                b = ob.nextShort();
                short c = (short) (a + b);
                System.out.println("The sum is :" + c);
                ob.close();
        }
}
```

OUTPUT:



```
Enter two numbers
3000 12
The sum is :3012
```



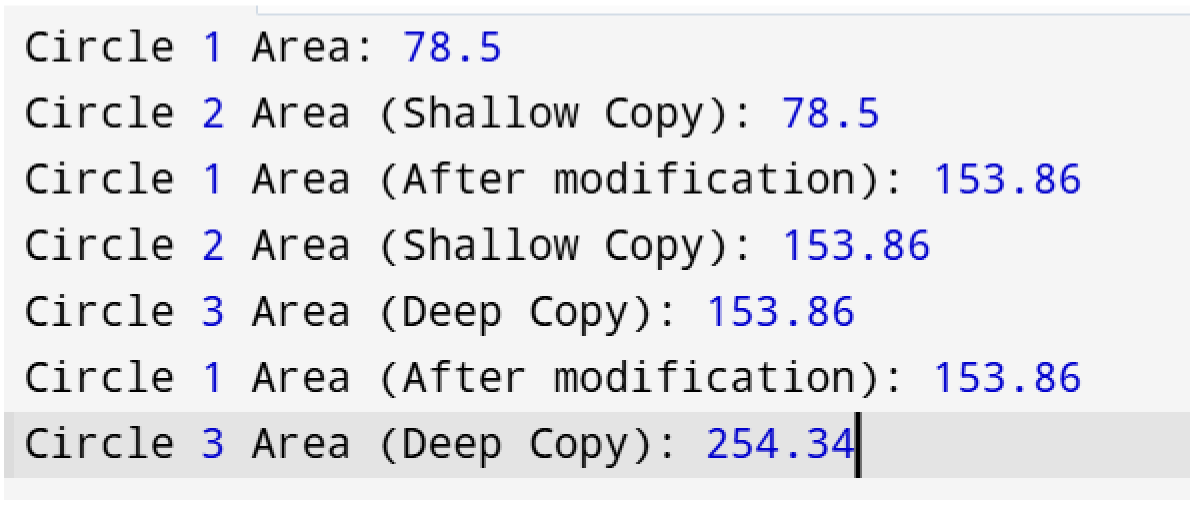
```
Enter two numbers
60000 6000
ERROR!
Exception in thread "main" java.util.InputMismatchException: Value out
    of range. Value:"60000" Radix:10
    at java.base/java.util.Scanner.nextShort(Scanner.java:2133)
    at java.base/java.util.Scanner.nextShort(Scanner.java:2081)
    at program1.main(Main.java:8)
```

QUESTION 2:
Write a program that accepts the radius of a circle and displays the area of the circle. Overload the constructor to accept radius as input and another circle object as input. Then show the effect of shallow vs deep copy of objects. Declare a constant pi equals to 3.14 using the OOP concept.

CODE:
```
class Circle {
  private final double PI = 3.14;
  private double radius;

  // Constructor to accept radius as input
  public Circle(double radius) {
    this.radius = radius;
  }
  public Circle(Circle other) {
    // Create a shallow copy of circle1
    this.radius = other.radius;
  }
  public double calculateArea() {
    return PI * radius * radius;
  }

  public double getRadius() {
    return radius;
  }

  public void setRadius(double radius) {
    this.radius = radius;
  }
}

public class program2{
  public static void main(String[] args) {
    Circle circle1 = new Circle(5);
    System.out.println("Circle 1 Area: " + circle1.calculateArea());

    Circle circle2 = circle1;
    System.out.println("Circle 2 Area (Shallow Copy): " + circle2.calculateArea());

    circle1.setRadius(7);
    System.out.println("Circle 1 Area (After modification): " + circle1.calculateArea());
    System.out.println("Circle 2 Area (Shallow Copy): " + circle2.calculateArea());

    Circle circle3 = new Circle(circle1.getRadius());
```

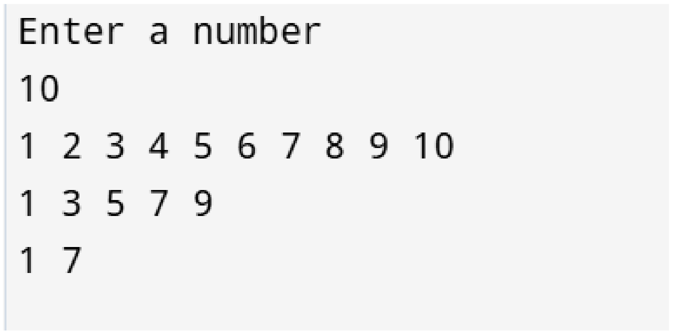
```
        System.out.println("Circle 3 Area (Deep Copy): " + circle3.calculateArea());

        circle3.setRadius(9);
        System.out.println("Circle 1 Area (After modification): " + circle1.calculateArea());
        System.out.println("Circle 3 Area (Deep Copy): " + circle3.calculateArea());
    }
}
```

OUTPUT:



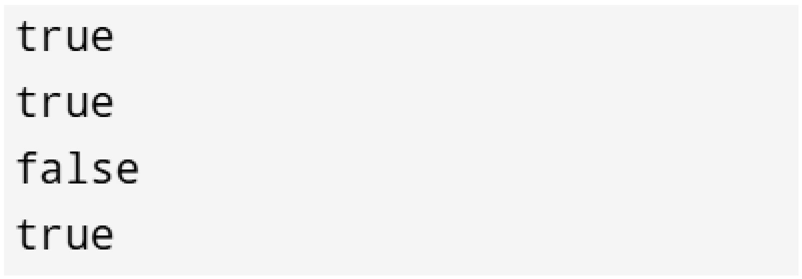
```
Circle 1 Area: 78.5
Circle 2 Area (Shallow Copy): 78.5
Circle 1 Area (After modification): 153.86
Circle 2 Area (Shallow Copy): 153.86
Circle 3 Area (Deep Copy): 153.86
Circle 1 Area (After modification): 153.86
Circle 3 Area (Deep Copy): 254.34
```

QUESTION 3:
Input n and consider an array of 1 to n natural numbers. Skip every second value and print the resulting series. Then select every third value from the remaining numbers. Print the resulting series. Repeat this process till the skip count becomes greater than the length of the list.
{1,2,3,4,5,6,7,8,9,10}→{1,3,5,7,9}→{1,7}

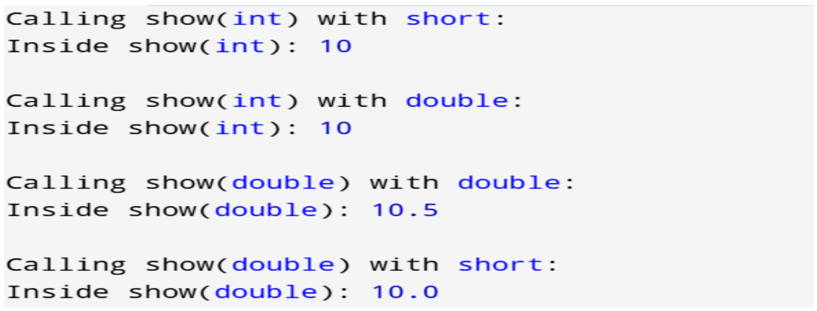
CODE:
```
import java.util.*;
public class program3 {
    public static void main(String[] args) {
        Scanner ob = new Scanner(System.in);
        System.out.println("Enter a number ");
        int n = ob.nextInt();
        int arr[];
        arr = new int[n];
        int diff = 1, multiplier = 2;
        for(int i = 0; i < n; i++) {
            arr[i] = i+1;
        }
        while(true){
            for(int i = 0; i < n; i+=diff){
                System.out.print(arr[i]+" ");
            }
            System.out.println();
            diff = diff*multiplier;
            multiplier++;
            if(diff >= n)
                break;
        }

        ob.close();
    }
}
```
OUTPUT:

```
Enter a number
10
1 2 3 4 5 6 7 8 9 10
1 3 5 7 9
1 7
```

QUESTION 4:
Write a program that accepts a String and assigns it to another. Check the outcome of comparison with == and equals() method. Take two Strings and put same input for them. Repeat the equality checking. Observe the outcome.

CODE:
```
import java.util.*;
public class program4{
public static void main(String[] args){
        String s1, s2, s3, s4;
        Scanner ob = new Scanner(System.in);
        s1 = "anirudh";
        s2 = s1;
        s3 = "modi";
        s4 = new String(s3);
        System.out.println(s1==s2);
        System.out.println(s1.equals(s2));
        System.out.println(s3==s4);
        System.out.println(s3.equals(s4));
        ob.close();
    }
}
```

OUTPUT:

```
true
true
false
true
```

QUESTION 5:
Write a program where class contains void show(int) to display the argument passed. Call the function once with short as actual parameter and again double as actual parameter. Add another function as void show(double) . Repeat the calls. Observe the outcomes in each case.

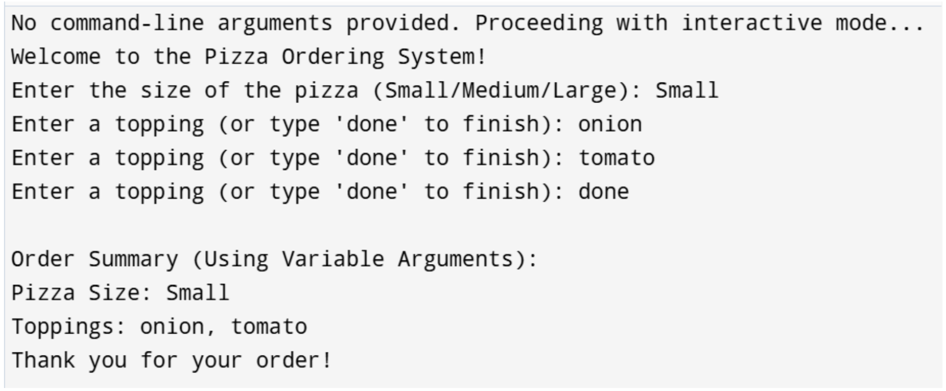
CODE:
```
class OverloadDemo {
    void show(int num) {
        System.out.println("Inside show(int): " + num);
    }

    void show(double num) {
        System.out.println("Inside show(double): " + num);
    }
}

public class Program5 {
    public static void main(String[] args) {
        OverloadDemo obj = new OverloadDemo();

        short shortValue = 10;
        System.out.println("Calling show(int) with short:");
        obj.show(shortValue);

        double doubleValue = 10.5;
        System.out.println("\nCalling show(int) with double:");
        obj.show((int) doubleValue);

        System.out.println("\nCalling show(double) with double:");
        obj.show(doubleValue);

        System.out.println("\nCalling show(double) with short:");
        obj.show((double) shortValue);
    }
}
```
OUTPUT:

```
Calling show(int) with short:
Inside show(int): 10

Calling show(int) with double:
Inside show(int): 10

Calling show(double) with double:
Inside show(double): 10.5

Calling show(double) with short:
Inside show(double): 10.0
```

QUESTION 6:
Create a program for ordering Pizza. The user should mention the size of the Pizza and the toppings he/she wants. A user may ask for any toppings. Implement this using (i) variable arguments concept and (ii) command line arguments.

CODE:
```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class program6 {

   public static void orderPizzaWithVarArgs(String size, String... toppings) {
      System.out.println("\nOrder Summary (Using Variable Arguments):");
      System.out.println("Pizza Size: " + size);
      System.out.println("Toppings: " + (toppings.length > 0 ? String.join(", ", toppings) : "No toppings
selected"));
      System.out.println("Thank you for your order!\n");
   }

   public static void main(String[] args) {
      if (args.length > 0) {
         String size = args[0];
         String[] toppings = Arrays.copyOfRange(args, 1, args.length);

         System.out.println("Order Summary (Using Command-Line Arguments):");
         System.out.println("Pizza Size: " + size);
         System.out.println("Toppings: " + (toppings.length > 0 ? String.join(", ", toppings) : "No toppings
selected"));
         System.out.println("Thank you for your order!\n");
         System.exit(0);
      } else {
         System.out.println("No command-line arguments provided. Proceeding with interactive mode...");
      }

      Scanner scanner = new Scanner(System.in);

      System.out.println("Welcome to the Pizza Ordering System!");

      System.out.print("Enter the size of the pizza (Small/Medium/Large): ");
      String size = scanner.nextLine();

      List<String> toppingsList = new ArrayList<>();
```

```
        while (true) {
            System.out.print("Enter a topping (or type 'done' to finish): ");
            String topping = scanner.nextLine();
            if (topping.equalsIgnoreCase("done")) {
                break;
            }
            toppingsList.add(topping);
        }

        String[] toppingsArray = toppingsList.toArray(new String[0]);

        orderPizzaWithVarArgs(size, toppingsArray);

        scanner.close();
    }
}
```

OUTPUT:

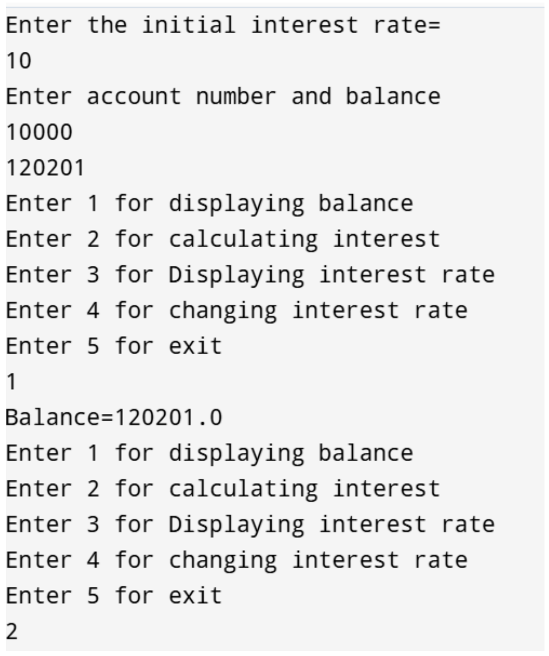


```
No command-line arguments provided. Proceeding with interactive mode...
Welcome to the Pizza Ordering System!
Enter the size of the pizza (Small/Medium/Large): Small
Enter a topping (or type 'done' to finish): onion
Enter a topping (or type 'done' to finish): tomato
Enter a topping (or type 'done' to finish): done

Order Summary (Using Variable Arguments):
Pizza Size: Small
Toppings: onion, tomato
Thank you for your order!
```

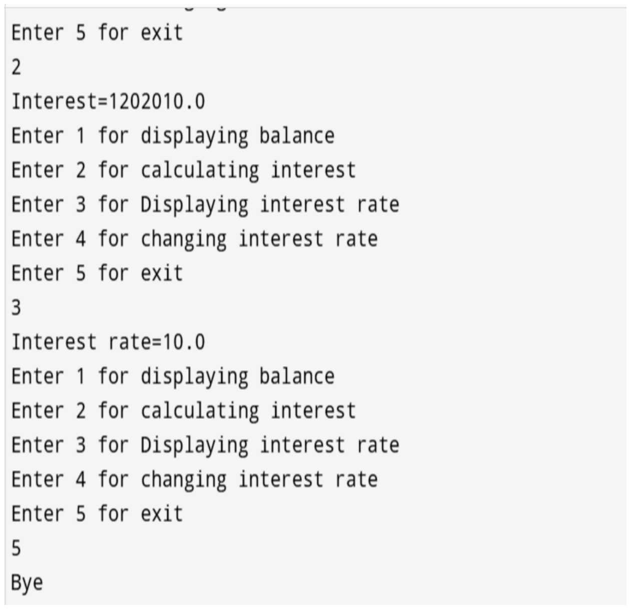
QUESTION 7:
Design a BankAcct class with account number, balance and interest rate as attribute. Interest rate is same for all account. Support must be there to initialize, change and display the interest rate. Also supports are to be there to return balance and calculate interest.

CODE:
```
import java.util.Scanner;
public class program7{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the initial interest rate=");
        double r=sc.nextDouble();
        new BankAcct(r);
        System.out.println("Enter account number and balance");
        long acct=sc.nextLong();
        double bal=sc.nextDouble();
        BankAcct bank=new BankAcct(acct,bal);
        while(true){
        System.out.println("Enter 1 for displaying balance\nEnter 2 for calculating interest\nEnter 3 for
Displaying interest rate\nEnter 4 for changing interest rate\nEnter 5 for exit");
            int ch=sc.nextInt();
            if(ch==5){
                System.out.println("Bye");
            break;
            }
            switch (ch) {
                case 1->bank.display_balance();
                case 2->System.out.println("Interest="+bank.calc_interest());
                case 3->bank.display_interest();
                case 4->{
                    System.out.println("Enter new Interest Rate");
                    BankAcct.set_interest(sc.nextDouble());
                }
                default-> System.out.println("Wrong Choice");

            }
    }
        sc.close();
    }
}
class BankAcct{
    @SuppressWarnings("unused")
    private long account;
    private double balance;
    static double interest;
    BankAcct(double interest_rate){
        account=0L;
        balance=0.0;
        interest=interest_rate;

    }
    BankAcct(long account,double balance){
        this.account=account;
        this.balance=balance;
    }
    static void set_interest(double rate){
        interest=rate;
    }
    void display_interest(){
        System.out.println("Interest rate="+interest);
    }
    void display_balance(){
        System.out.println("Balance="+balance);
    }
    double calc_interest(){
        return interest*balance;
    }

}
```

OUTPUT:



```
Enter the initial interest rate=
10
Enter account number and balance
10000
120201
Enter 1 for displaying balance
Enter 2 for calculating interest
Enter 3 for Displaying interest rate
Enter 4 for changing interest rate
Enter 5 for exit
1
Balance=120201.0
Enter 1 for displaying balance
Enter 2 for calculating interest
Enter 3 for Displaying interest rate
Enter 4 for changing interest rate
Enter 5 for exit
2
```



```
Enter 5 for exit
2
Interest=1202010.0
Enter 1 for displaying balance
Enter 2 for calculating interest
Enter 3 for Displaying interest rate
Enter 4 for changing interest rate
Enter 5 for exit
3
Interest rate=10.0
Enter 1 for displaying balance
Enter 2 for calculating interest
Enter 3 for Displaying interest rate
Enter 4 for changing interest rate
Enter 5 for exit
5
Bye
```

QUESTION 8:
For a programme (such as, BCSE), each Instructor has name and phone number. Each textbook has a title, author name and publisher. Each course (that is, subject) has a course name, instructor and text book.

- One can set the data for a textbook and view the same.
- One can view instructor information and set the information.
- One can set the course data and view the same.

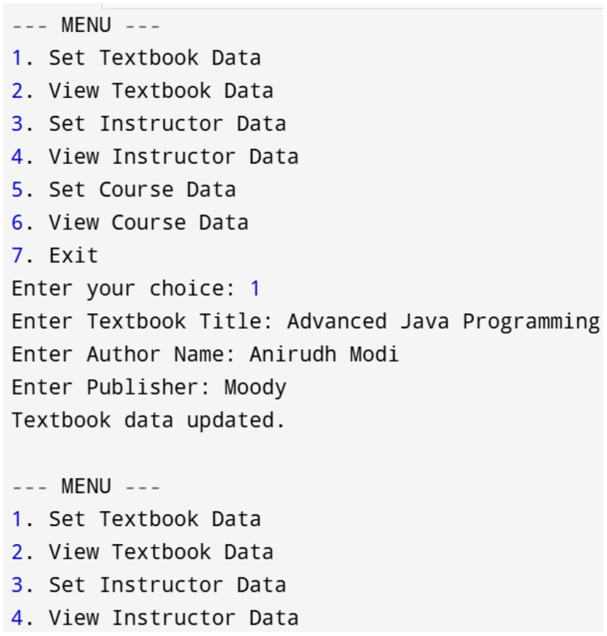
Design and implement the classes.

CODE:
```
import java.util.Scanner;
class Instructor {
    private String name;
    private String phone;
    // Constructor
    public Instructor(String name, String phone) {
        this.name = name;
        this.phone = phone;
    }
    // Set instructor data
    public void setData(String name, String phone) {
        this.name = name;
        this.phone = phone;
    }
    // Display instructor information
    public void display() {
        System.out.println("Instructor Name: " + name);
        System.out.println("Phone Number: " + phone);
    }
}
class Textbook {
    private String title;
    private String author;
    private String publisher;
    // Constructor
    public Textbook(String title, String author, String publisher) {
        this.title = title;
        this.author = author;
        this.publisher = publisher;
    }
    // Set textbook data
    public void setData(String title, String author, String publisher) {
        this.title = title;
        this.author = author;
        this.publisher = publisher;
    }
    // Display textbook information
    public void display() {
        System.out.println("Textbook Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Publisher: " + publisher);
    }
}
class Course {
    private String courseName;
    private Instructor instructor;
    private Textbook textbook;
    // Constructor
    public Course(String courseName, Instructor instructor, Textbook textbook) {
        this.courseName = courseName;
        this.instructor = instructor;
        this.textbook = textbook;
    }
    // Set course data
    public void setData(String courseName, Instructor instructor, Textbook textbook) {
        this.courseName = courseName;
        this.instructor = instructor;
        this.textbook = textbook;
    }
    // Display course information
    public void display() {
        System.out.println("Course Name: " + courseName);
        System.out.println("\nInstructor Details:");
        instructor.display();
        System.out.println("\nTextbook Details:");
        textbook.display();
    }
}
public class program8 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Create default Instructor, Textbook, and Course
        Instructor instructor = new Instructor("Default Instructor", "000-000-0000");
        Textbook textbook = new Textbook("Default Title", "Default Author", "Default Publisher");
        Course course = new Course("Default Course", instructor, textbook);
        int choice;
        do {
```

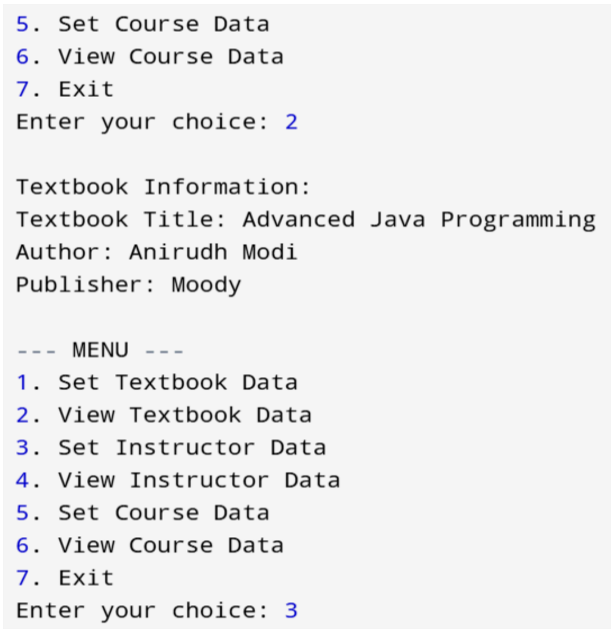
```
System.out.println("\n--- MENU ---");
System.out.println("1. Set Textbook Data");
System.out.println("2. View Textbook Data");
System.out.println("3. Set Instructor Data");
System.out.println("4. View Instructor Data");
System.out.println("5. Set Course Data");
System.out.println("6. View Course Data");
System.out.println("7. Exit");
System.out.print("Enter your choice: ");
choice = scanner.nextInt();
scanner.nextLine(); // Consume newline

switch (choice) {
   case 1:
      // Set Textbook Data
      System.out.print("Enter Textbook Title: ");
      String title = scanner.nextLine();
      System.out.print("Enter Author Name: ");
      String author = scanner.nextLine();
      System.out.print("Enter Publisher: ");
      String publisher = scanner.nextLine();
      textbook.setData(title, author, publisher);
      System.out.println("Textbook data updated.");
      break;

   case 2:
      // View Textbook Data
      System.out.println("\nTextbook Information:");
      textbook.display();
      break;

   case 3:
      // Set Instructor Data
      System.out.print("Enter Instructor Name: ");
      String name = scanner.nextLine();
      System.out.print("Enter Phone Number: ");
      String phone = scanner.nextLine();
      instructor.setData(name, phone);
      System.out.println("Instructor data updated.");
      break;

   case 4:
      // View Instructor Data
      System.out.println("\nInstructor Information:");
```

```
                    instructor.display();
                    break;

                case 5:
                    // Set Course Data
                    System.out.print("Enter Course Name: ");
                    String courseName = scanner.nextLine();
                    course.setData(courseName, instructor, textbook);
                    System.out.println("Course data updated.");
                    break;

                case 6:
                    // View Course Data
                    System.out.println("\nCourse Information:");
                    course.display();
                    break;

                case 7:
                    System.out.println("Exiting the program. Thank you!");
                    break;

                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 7);

        scanner.close();
    }
}
```

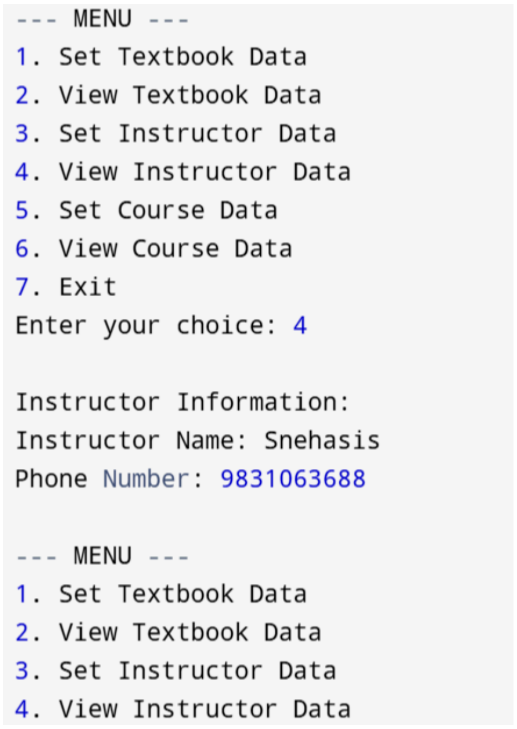
OUTPUT:



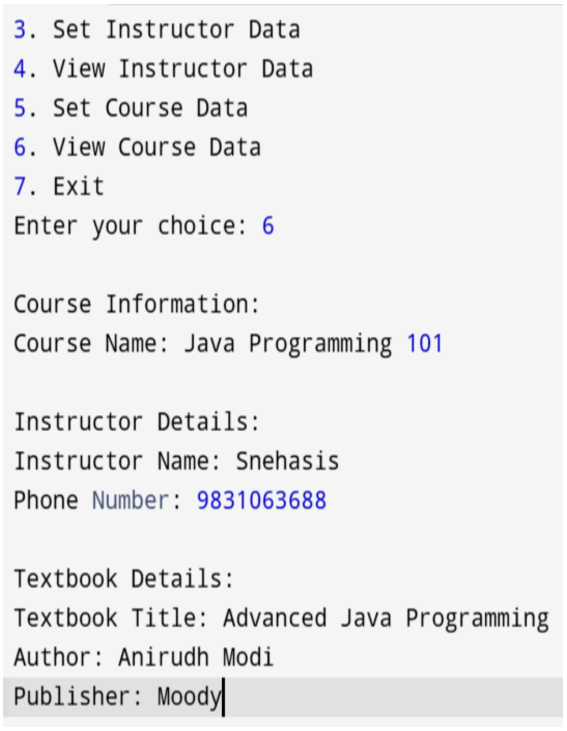
```
--- MENU ---
1. Set Textbook Data
2. View Textbook Data
3. Set Instructor Data
4. View Instructor Data
5. Set Course Data
6. View Course Data
7. Exit
Enter your choice: 1
Enter Textbook Title: Advanced Java Programming
Enter Author Name: Anirudh Modi
Enter Publisher: Moody
Textbook data updated.

--- MENU ---
1. Set Textbook Data
2. View Textbook Data
3. Set Instructor Data
4. View Instructor Data
```



```
5. Set Course Data
6. View Course Data
7. Exit
Enter your choice: 2

Textbook Information:
Textbook Title: Advanced Java Programming
Author: Anirudh Modi
Publisher: Moody

--- MENU ---
1. Set Textbook Data
2. View Textbook Data
3. Set Instructor Data
4. View Instructor Data
5. Set Course Data
6. View Course Data
7. Exit
Enter your choice: 3
```



```
--- MENU ---
1. Set Textbook Data
2. View Textbook Data
3. Set Instructor Data
4. View Instructor Data
5. Set Course Data
6. View Course Data
7. Exit
Enter your choice: 4

Instructor Information:
Instructor Name: Snehasis
Phone Number: 9831063688

--- MENU ---
1. Set Textbook Data
2. View Textbook Data
3. Set Instructor Data
4. View Instructor Data
```



```
3. Set Instructor Data
4. View Instructor Data
5. Set Course Data
6. View Course Data
7. Exit
Enter your choice: 6

Course Information:
Course Name: Java Programming 101

Instructor Details:
Instructor Name: Snehasis
Phone Number: 9831063688

Textbook Details:
Textbook Title: Advanced Java Programming
Author: Anirudh Modi
Publisher: Moody
```

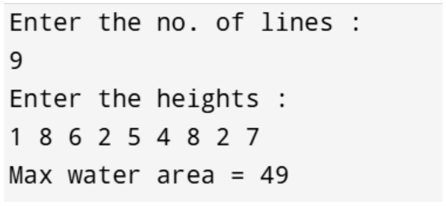
QUESTION 1:

An integer array representing height of length n is given as input. There are n vertical lines drawn from the array such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Any two lines along with the x-axis form a container that can hold water. Find the largest container that can hold maximum water. Collection classes are NOT allowed.

CODE:

```
import java.util.*;
public class program1{
    int rectangle(){
        Scanner in = new Scanner(System.in);
        int n;
        System.out.println("Enter the no. of lines : ");
        n = in.nextInt();
        int height[] = new int[n];
        System.out.println("Enter the heights :");
        for(int i = 0; i < n; i++)
            height[i] = in.nextInt();
        int left = 0, right = height.length - 1;
        int maxWater = 0;
        while(left < right){
            int width = right - left;
            int h = Math.min(height[left], height[right]);
            int area = width*h;
            maxWater = Math.max(maxWater, area);
            if(height[left] < height[right])
                left++;
            else
                right--;
        }
        in.close();
        return maxWater;
    }
    public static void main(String args[]){
        program1 ob = new program1();
        int answer = ob.rectangle();
        System.out.println("Max water area = "+ answer);
    }
}
```

OUTPUT:

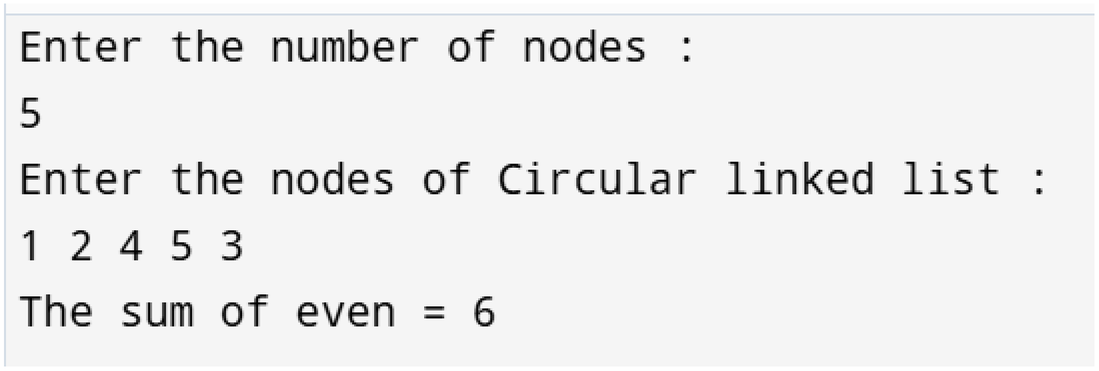


```
Enter the no. of lines :
9
Enter the heights :
1 8 6 2 5 4 8 2 7
Max water area = 49
```

QUESTION 2:
Print the sum of all even numbers stored in a circular linked list. Represent Node as a static inner class. Collection classes are NOT allowed.

CODE:
```
import java.util.*;
public class program2 {
    static class Node {
        int data;
        Node next;
        Node(int data) {
            this.data = data;
            this.next = null;
        }
    }
    private Node head = null;
    public void add(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            head.next = head;
        } else {
            Node temp = head;
            while (temp.next != head) {
                temp = temp.next;
            }
            temp.next = newNode;
            newNode.next = head;
        }
    }
    public int sumOfEven(){
        if(head == null){
            return 0;
        }
        Node temp = head;
        int sum = 0;
        do{
            if(temp.data % 2 == 0){
                sum += temp.data;
            }
            temp = temp.next;
        }while(temp != head);
        return sum;
    }
    public static void main(String args[]){
        program2 list = new program2();
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number of nodes : ");
        int n = in.nextInt();
        System.out.println("Enter the nodes of Circular linked list : ");
        for(int i = 0; i < n; i++){
            int x = in.nextInt();
            list.add(x);
        }
        int sum = list.sumOfEven();
        System.out.println("The sum of even = "+ sum);
        in.close();
    }
}
```

OUTPUT:



```
Enter the number of nodes :
5
Enter the nodes of Circular linked list :
1 2 4 5 3
The sum of even = 6
```

QUESTION 3:
Given the root of a binary tree, return all root-to-leaf paths in any order. A leaf is a node with no children. Treat a node as an inner class. Write appropriate code representing the node containing an integer (1>n>100) or a string (6 letter names). You may use ArrayList (List<String> example = new ArrayList<String/Integer>();).

CODE:
```
import java.util.*;

class BinaryTree {
    public static class Node {
        String data;
        Node left, right;
        Node(String data) {
            this.data = data;
        }
    }

    Node root;

    Node buildTree(String inputs[], int i) {
        if (i < inputs.length) {
            if (inputs[i].equals("null")) return null;
            Node node = new Node(inputs[i]);
            node.left = buildTree(inputs, 2 * i + 1);
            node.right = buildTree(inputs, 2 * i + 2);
            return node;
        }
        return null;
    }

    void build(String[] inputs) {
        root = buildTree(inputs, 0);
    }

    void printPath(Node node, ArrayList<String> array) {
        if (node == null) return;

        array.add(node.data);
        if (node.left == null && node.right == null) {
            for (String val : array)
                System.out.print(val + " ");
            System.out.println();
        } else {
            printPath(node.left, array);
            printPath(node.right, array);
        }
        array.remove(array.size() - 1);
    }
    void path() {
        ArrayList<String> array = new ArrayList<>();
        printPath(root, array);
    }
}
public class program3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        BinaryTree tree = new BinaryTree();
        System.out.print("Enter the number of nodes: ");
        int n = sc.nextInt();
        String[] inputs = new String[n];
        System.out.println("Enter node values in level-order (use 'null' for missing nodes):");
        for (int i = 0; i < n; i++) {
            inputs[i] = sc.next();
        }
        tree.build(inputs);
        System.out.println("Root-to-leaf paths:");
        tree.path();
        sc.close();
    }
}
```

OUTPUT:

```
Enter the number of nodes: 9
Enter node values in level-order (use 'null' for missing nodes):
1 2 3 4 5 6 7

Root-to-leaf paths:
1 2 4
1 2 5
1 3 6
1 3 7
```

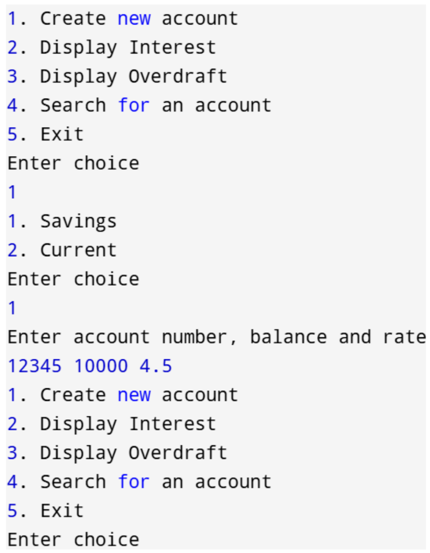
QUESTION 4:
Every bank account holds an account no and a calculateInterest method. A customer can have a "SavingsAccount" and/or a "CurrentAccount". For current account, there is a method called displayOverdraftAmount(). Different accounts can have different interest rates. User should be able to verify the existence of an account, adding new account and displaying all accounts. Implement appropriate objects utilizing inheritance and show its behavior from the parent class.

CODE:
```
import java.util.*;

abstract class BankAcct {
    long accno;
    double balance;
    double rate;
    char type;

    BankAcct(long accno, double balance, double rate, char t) {
        this.accno = accno;
        this.balance = balance;
        this.rate = rate;
        type = t;
    }

    double calculateInterest() {
        return balance * rate / 100;
    }

    abstract void displayOverdraftAmount();
}

class SavingsAcct extends BankAcct {
    SavingsAcct(long accno, double balance, double rate, char type) {
        super(accno, balance, rate, type);
    }

    @Override
    void displayOverdraftAmount() {
    }
}

class CurrentAcct extends BankAcct {
    private double overdraft;

    CurrentAcct(long accno, double balance, double rate, double overdraft, char type) {
        super(accno, balance, rate, type);
        this.overdraft = overdraft;
    }

    @Override
    void displayOverdraftAmount() {
        System.out.println("Overdraft limit: " + overdraft);
    }
}

class prog4 {
    static int search(ArrayList<BankAcct> b, long no) {
        for (int i = 0; i < b.size(); i++) {
            if (b.get(i).accno == no)
                return i;
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayList<BankAcct> a = new ArrayList<BankAcct>();
        while (true) {
            System.out.println(
                    "1. Create new account\n2. Display Interest\n3. Display Overdraft\n4. Search for an
account\n5. Exit\nEnter choice");
            int choice = sc.nextInt();
            long accno;
            double balance;
            double rate, overdraft;
            BankAcct b;
            switch (choice) {
                case 1:
                    System.out.println("1. Savings\n2. Current\nEnter choice");
                    int type = sc.nextInt();
                    if (type == 1) {
                        System.out.println("Enter account number, balance and rate");
                        accno = sc.nextLong();
                        balance = sc.nextDouble();
                        rate = sc.nextDouble();
                        SavingsAcct s = new SavingsAcct(accno, balance, rate, 's');
                        b = s;
                        a.add(b);
                    } else if (type == 2) {
```

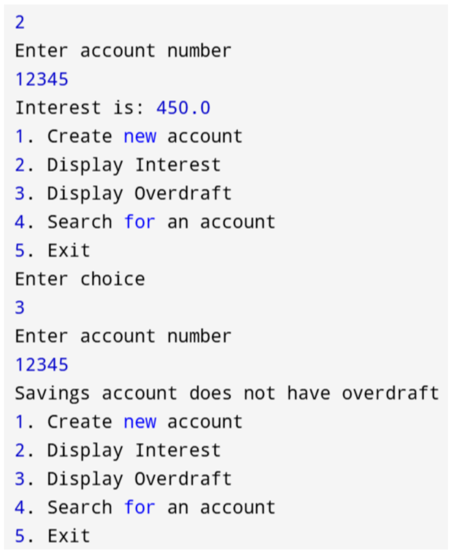
```
        System.out.println("Enter account number, balance, rate and overdraft limit");
        accno = sc.nextLong();
        balance = sc.nextDouble();
        rate = sc.nextDouble();
        overdraft = sc.nextDouble();
        CurrentAcct c = new CurrentAcct(accno, balance, rate, overdraft, 'c');
        b = c;
        a.add(b);
    }
    break;
case 2:
    System.out.println("Enter account number");
    accno = sc.nextLong();
    int n = search(a, accno);
    if (n != -1) {
        b = a.get(n);
        System.out.println("Interest is: " + b.calculateInterest());
    }
    break;
case 3:
    System.out.println("Enter account number");
    accno = sc.nextLong();
    int m = search(a, accno);
    if (m != -1) {
        b = a.get(m);
        if (b.type != 's')
            b.displayOverdraftAmount();
        else
            System.out.println("Savings account does not have overdraft");
    }
    break;
case 4:
    System.out.println("Enter account number");
    accno = sc.nextLong();
    int k = search(a, accno);
    if (k != -1) {
        System.out.println("Account found");
    } else {
        System.out.println("Account not found");
    }
    break;
case 5:
    System.out.println("Exiting....");
    sc.close();  // Close the scanner when exiting the program
```

```
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid input");
            }
        }
    }
}
```

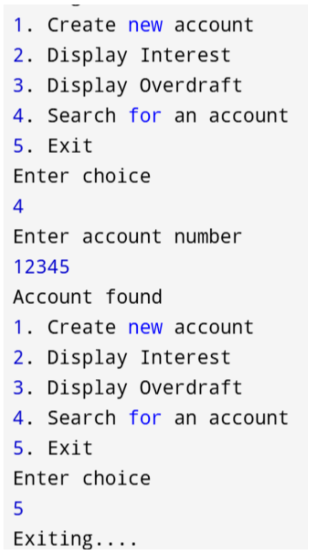
OUTPUT:



```
1. Create new account
2. Display Interest
3. Display Overdraft
4. Search for an account
5. Exit
Enter choice
1
1. Savings
2. Current
Enter choice
1
Enter account number, balance and rate
12345 10000 4.5
1. Create new account
2. Display Interest
3. Display Overdraft
4. Search for an account
5. Exit
Enter choice
```



```
2
Enter account number
12345
Interest is: 450.0
1. Create new account
2. Display Interest
3. Display Overdraft
4. Search for an account
5. Exit
Enter choice
3
Enter account number
12345
Savings account does not have overdraft
1. Create new account
2. Display Interest
3. Display Overdraft
4. Search for an account
5. Exit
```



```
1. Create new account
2. Display Interest
3. Display Overdraft
4. Search for an account
5. Exit
Enter choice
4
Enter account number
12345
Account found
1. Create new account
2. Display Interest
3. Display Overdraft
4. Search for an account
5. Exit
Enter choice
5
Exiting....
```

QUESTION 5:
Each customer of a bank has a customer id, name, and current loan amount and phone number. One can change the attributes like name, phone number. A customer may ask for a loan of a certain amount. It is granted provided the sum of current loan amount and asked amount does not exceed credit limit (fixed amount for all customers). A customer can be a privileged customer. For such customers credit the limit is higher. Once a loan is sanctioned necessary updates should be made. Any type of customer should be able to find his credit limit, current loan amount and amount of loan (s)he can seek. No customer can change customer id once created. Print customer name when the object is printed by toString() method.
Design and implement the classes. Show the working through a menu driven user interface.

CODE:
```
import java.util.ArrayList;
import java.util.Scanner;
// Base class for Customer
class Customer {
   private final int customerId; // Cannot be changed once set
   private String name;
   private String phoneNumber;
   protected double currentLoan;
   protected static final double CREDIT_LIMIT = 50000; // Default credit limit for all customers
   // Constructor
   public Customer(int customerId, String name, String phoneNumber) {
      this.customerId = customerId;
      this.name = name;
      this.phoneNumber = phoneNumber;
      this.currentLoan = 0;
   }
   // Getters and Setters
   public int getCustomerId() {
      return customerId;
   }
   public String getName() {
      return name;
   }
   public void setName(String newName) {
      this.name = newName;
   }
   public String getPhoneNumber() {
      return phoneNumber;
   }
   public void setPhoneNumber(String newPhoneNumber) {
      this.phoneNumber = newPhoneNumber;
   }
```

```
    public double getCurrentLoan() {
        return currentLoan;
    }
    public double getCreditLimit() {
        return CREDIT_LIMIT;
    }
    // Method to request a loan
    public boolean requestLoan(double amount) {
        if (currentLoan + amount <= CREDIT_LIMIT) {
            currentLoan += amount;
            return true;
        } else {
            return false;
        }
    }
    // Method to check loan eligibility
    public double getEligibleLoanAmount() {
        return CREDIT_LIMIT - currentLoan;
    }
    @Override
    public String toString() {
        return "ID: " + customerId + " | Name: " + name + " | Phone: " + phoneNumber +
                " | Loan: " + currentLoan + " | Limit: " + getCreditLimit();
    }
}
// Privileged Customer with a higher credit limit
class PrivilegedCustomer extends Customer {
    private static final double PRIVILEGED_CREDIT_LIMIT = 100000;
    public PrivilegedCustomer(int customerId, String name, String phoneNumber) {
        super(customerId, name, phoneNumber);
    }
    @Override
    public double getCreditLimit() {
        return PRIVILEGED_CREDIT_LIMIT;
    }
    @Override
    public boolean requestLoan(double amount) {
        if (currentLoan + amount <= PRIVILEGED_CREDIT_LIMIT) {
            currentLoan += amount;
            return true;
        } else {
            return false;
        }
    }
```

```
}
// Bank system to manage multiple customers
public class program5 {
    private static Scanner scanner = new Scanner(System.in);
    private static ArrayList<Customer> customers = new ArrayList<>();
    public static void main(String[] args) {
        while (true) {
            System.out.println("\n---- Bank System Menu ----");
            System.out.println("1. Add Customer");
            System.out.println("2. View All Customers");
            System.out.println("3. Manage Existing Customer");
            System.out.println("4. Exit");
            System.out.print("Enter choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            switch (choice) {
                case 1:
                    addCustomer();
                    break;
                case 2:
                    viewCustomers();
                    break;
                case 3:
                    manageCustomer();
                    break;
                case 4:
                    System.out.println("Exiting...");
                    return;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        }
    }
    // Method to add a new customer
    private static void addCustomer() {
        System.out.println("Creating Customer Profile...");
        System.out.print("Enter Customer ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("Enter Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Phone Number: ");
        String phone = scanner.nextLine();
        System.out.print("Is the customer privileged? (yes/no): ");
        String type = scanner.nextLine().toLowerCase();
        Customer customer;
        if (type.equals("yes")) {
            customer = new PrivilegedCustomer(id, name, phone);
        } else {
            customer = new Customer(id, name, phone);
        }
        customers.add(customer);
        System.out.println("Customer added successfully.");
    }
    // Method to view all customers
    private static void viewCustomers() {
        if (customers.isEmpty()) {
            System.out.println("No customers available.");
            return;
        }
        System.out.println("\n---- Customer List ----");
        for (Customer customer : customers) {
            System.out.println(customer);
        }
    }
    // Method to manage an existing customer
    private static void manageCustomer() {
        if (customers.isEmpty()) {
            System.out.println("No customers available to manage.");
            return;
        }
        System.out.print("Enter Customer ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        Customer customer = findCustomerById(id);
        if (customer == null) {
            System.out.println("Customer not found.");
            return;
        }
        while (true) {
            System.out.println("\nManaging Customer: " + customer.getName());
            System.out.println("1. Update Name");
            System.out.println("2. Update Phone Number");
            System.out.println("3. Request Loan");
            System.out.println("4. Check Credit Limit & Loan Eligibility");
            System.out.println("5. Back to Main Menu");
            System.out.print("Enter choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            switch (choice) {
                case 1:
                    System.out.print("Enter new name: ");
                    String newName = scanner.nextLine();
                    customer.setName(newName);
                    System.out.println("Name updated successfully.");
                    break;
                case 2:
                    System.out.print("Enter new phone number: ");
                    String newPhone = scanner.nextLine();
                    customer.setPhoneNumber(newPhone);
                    System.out.println("Phone number updated successfully.");
                    break;
                case 3:
                    System.out.print("Enter loan amount: ");
                    double loanAmount = scanner.nextDouble();
                    if (customer.requestLoan(loanAmount)) {
                        System.out.println("Loan approved. Updated loan amount: " +
customer.getCurrentLoan());
                    } else {
                        System.out.println("Loan denied! Exceeds credit limit.");
                    }
                    break;
                case 4:
                    System.out.println("Credit Limit: " + customer.getCreditLimit());
                    System.out.println("Current Loan: " + customer.getCurrentLoan());
                    System.out.println("Eligible Loan Amount: " + customer.getEligibleLoanAmount());
                    break;
                case 5:
                    return;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        }
    }
    // Helper method to find a customer by ID
    private static Customer findCustomerById(int id) {
        for (Customer customer : customers) {
            if (customer.getCustomerId() == id) {
                return customer;
            }
        }
        return null;
    }
}
```

OUTPUT:

```
---- Bank System Menu ----
1. Add Customer
2. View All Customers
3. Manage Existing Customer
4. Exit
Enter choice: 1
Enter Customer ID: 101
Enter Name: Alice
Enter Phone Number: 9999988888
Is the customer privileged? (yes/no): yes
Customer added successfully.

---- Bank System Menu ----
1. Add Customer
...
```

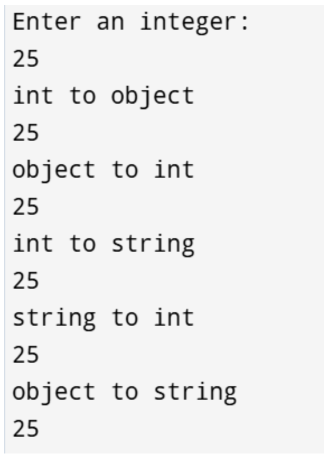
QUESTION 6:

Consider a wrapper class for a numeric basic type. Check the support for the following: conversion from i) basic type to object ii) object to basic type iii) basic type to String iv) String (holding numeric data) to numeric object v) object to String.

CODE:

```
import java.util.*;
public class program6 {
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    System.out.println("Enter an integer:");
    int n2 = in.nextInt();
    System.out.println("int to object");
    Integer n1 = n2;
    System.out.println(n1);
    System.out.println("object to int");
    int n3 = n1;
    System.out.println(n3);
    System.out.println("int to string");
    String s = Integer.toString(n2);
    System.out.println(s);
    System.out.println("string to int");
    int n4 = Integer.parseInt(s);
    System.out.println(n4);
    System.out.println("object to string");
    String s2 = n1.toString();
    System.out.println(s2);
    in.close();
  }
}
```

OUTPUT:



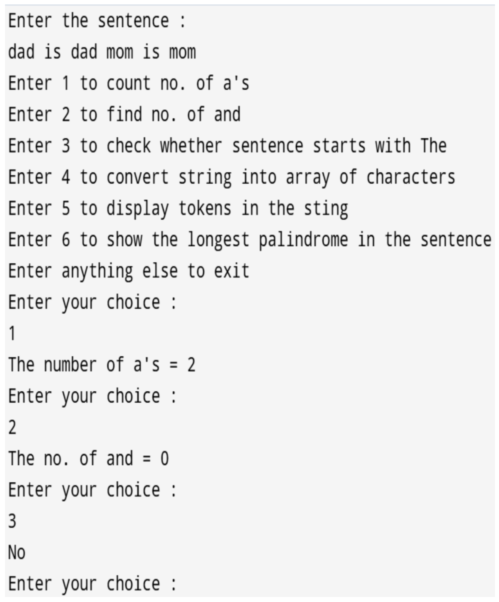
```
Enter an integer:
25
int to object
25
object to int
25
int to string
25
string to int
25
object to string
25
```

QUESTION 7:
Take a String input that contains multiple words. Do the following: i) number of times ‘a’ appears ii) number of times “and” appears iii) whether it starts with “The” or not iv) put the String into an array of characters v) display the tokens in the String (tokens are the substrings separated by space or @ or .) vi)Find the largest palindrome in a given input sentence after removing any non-alphanumeric character.

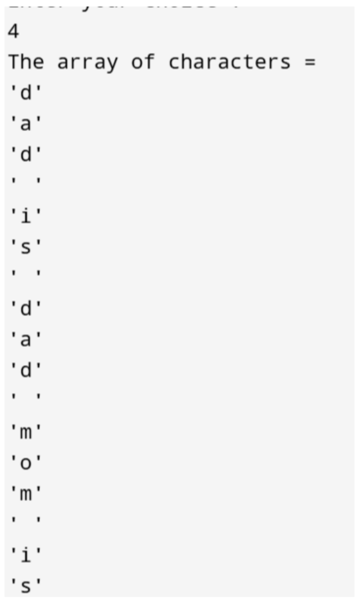
CODE:
```
import java.util.*;
public class program7{

  String findLargestPalindrome(String word) {
    if (word == null || word.length() < 2) {
      return word;
    }

    String longest = "";
    for (int i = 0; i < word.length(); i++) {
      String oddPalindrome = expandAroundCenter(word, i, i);
      String evenPalindrome = expandAroundCenter(word, i, i + 1);

      if (oddPalindrome.length() > longest.length()) {
        longest = oddPalindrome;
      }
      if (evenPalindrome.length() > longest.length()) {
        longest = evenPalindrome;
      }
    }
    return longest;
  }

  private static String expandAroundCenter(String word, int left, int right) {
    while (left >= 0 && right < word.length() && word.charAt(left) == word.charAt(right)) {
      left--;
      right++;
    }
    return word.substring(left + 1, right);
  }
  public static void main(String[] args)
  {
    Scanner in = new Scanner(System.in);
    String sentence;
    System.out.println("Enter the sentence : ");
    sentence = in.nextLine();
    int choice;
    System.out.println("Enter 1 to count no. of a's\nEnter 2 to find no. of and\nEnter 3 to check whether
sentence starts with The\nEnter 4 to convert string into array of characters\nEnter 5 to display tokens in
the sting\nEnter 6 to show the longest palindrome in the sentence\nEnter anything else to exit");
    while(true)
    {
      System.out.println("Enter your choice : ");
      choice = in.nextInt();
      switch(choice){
        case 1:{
          String temp = sentence;
          int count  = 0;
          for(int i = 0; i < temp.length(); i++){
            char ch = temp.charAt(i);
            if(ch == 'a' || ch == 'A'){
              count++;
            }
          }
          System.out.println("The number of a's = " + count);
          break;
        }
        case 2:{
          String temp = sentence;
          int count = 0;
          for(int i = 0; i < temp.length() - 2; i++){
            if (temp.substring(i, i + 3).equals("and")){
              count++;
            }
          }
          System.out.println("The no. of and = "+ count);
          break;
        }
        case 3:{
          String temp = sentence;
          if(temp.charAt(0) == 'T' && temp.charAt(1) == 'h' && temp.charAt(2) == 'e')
            System.out.println("Yes");
          else
            System.out.println("No");
          break;
        }
        case 4:{
          String temp = sentence;
          System.out.println("The array of characters = ");
          for(int i = 0; i < temp.length(); i++){
            System.out.println("'"+temp.charAt(i)+"'");
        }
        break;
    }
    case 5:{
        String temp = sentence;
        String word = "";
        for(int i = 0; i < temp.length(); i++){
            char ch = temp.charAt(i);
            if(ch == ' '){
                System.out.println(word);
                word = "";
            }
            else if(ch == '.'){
                System.out.println(word);
                word = "";
            }
            else if(ch == '@'){
                System.out.println(word);
                word = "";
            }
            else{
                word = word + ch;
            }
        }
        break;
    }
    case 6:{
        String temp = "";
        for(int i = 0; i < sentence.length(); i++){
            char ch = sentence.charAt(i);
            int value = (int)ch;
            if(value >= 65 && value <= 90){
                temp += ch;
            }
            else if(value >= 97 && value <= 122){
                temp += ch;
            }
            else if(value >= 48 && value <= 57){
                temp += ch;
            }
            else{
                continue;
            }
```

```
                    }
                    program7 ob = new program7();
                    System.out.println(ob.findLargestPalindrome(temp));
                    if(ob.findLargestPalindrome(temp).length() == 0)
                        System.out.println("No palindrome found");
                    break;
                }
                default:
                    in.close();
                    System.exit(0);
            }
        }
    }
}
```

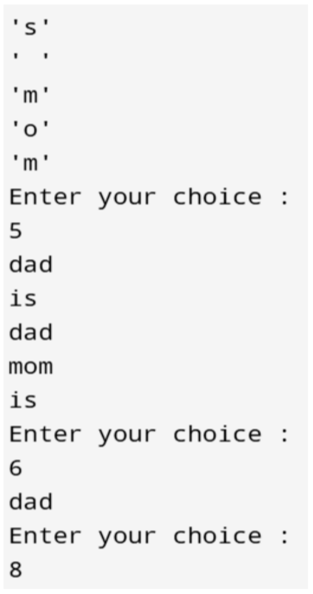
OUTPUT:



```
Enter the sentence :
dad is dad mom is mom
Enter 1 to count no. of a's
Enter 2 to find no. of and
Enter 3 to check whether sentence starts with The
Enter 4 to convert string into array of characters
Enter 5 to display tokens in the sting
Enter 6 to show the longest palindrome in the sentence
Enter anything else to exit
Enter your choice :
1
The number of a's = 2
Enter your choice :
2
The no. of and = 0
Enter your choice :
3
No
Enter your choice :
```



```
4
The array of characters =
'd'
'a'
'd'
' '
'i'
's'
' '
'd'
'a'
'd'
' '
'm'
'o'
'm'
' '
'i'
's'
```



```
's'
' '
'm'
'o'
'm'
Enter your choice :
5
dad
is
dad
mom
is
Enter your choice :
6
dad
Enter your choice :
8
```

QUESTION 8:

Write a class that implements the CharSequence interface found in the java.lang package. So, it would contain the following methods.

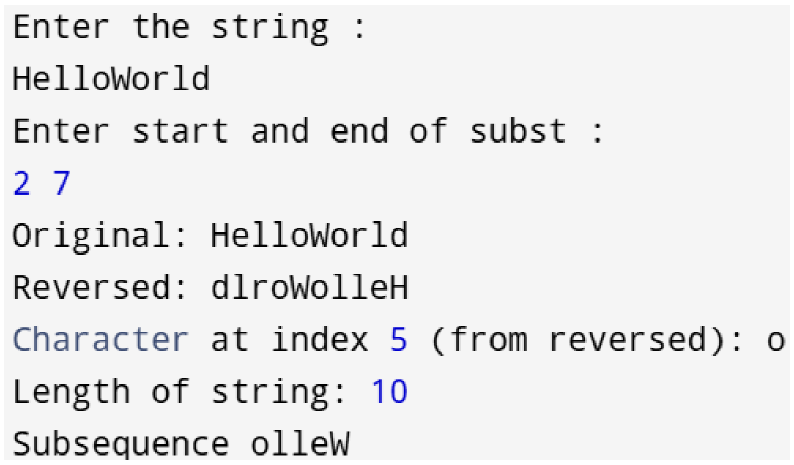
(i) char charAt(int index)
(ii) int length()
(iii)CharSequence subSequence(int start, int end)
(iv)Override toString()

Your implementation should return the string backwards. Select one of the sentences from the lecture slide to use as the data. Write a small main method to test your class; make sure to call all four methods.

CODE:

```
import java.util.*;
class ReverseCharSequence implements CharSequence {
   private String original;

   public ReverseCharSequence(String str) {
      this.original = str;
   }

   @Override
   public int length() {
      return original.length();
   }

   @Override
   public char charAt(int index) {
      return original.charAt(original.length() - 1 - index);
   }

   @Override
   public CharSequence subSequence(int start, int end) {
      String reversedSubstring = new StringBuilder(
            original.substring(original.length() - end, original.length() - start)).reverse().toString();
      return new ReverseCharSequence(reversedSubstring);
   }

   @Override
   public String toString() {
      return new StringBuilder(original).reverse().toString();
   }
}
public class program8{
   public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the string : ");
        String sentence = sc.nextLine();
        ReverseCharSequence reversedText = new ReverseCharSequence(sentence);
        System.out.println("Original: " + sentence);
        System.out.println("Reversed: " + reversedText.toString());
        System.out.println("Character at index 5 (from reversed): " + reversedText.charAt(5));
        System.out.println("Length of string: " + reversedText.length());
        System.out.println("Enter start and end of subst : ");
        int start = sc.nextInt();
        int end = sc.nextInt();
        System.out.println("Subsequence " + reversedText.subSequence(start, end));
        sc.close();
    }
}
```

OUTPUT:

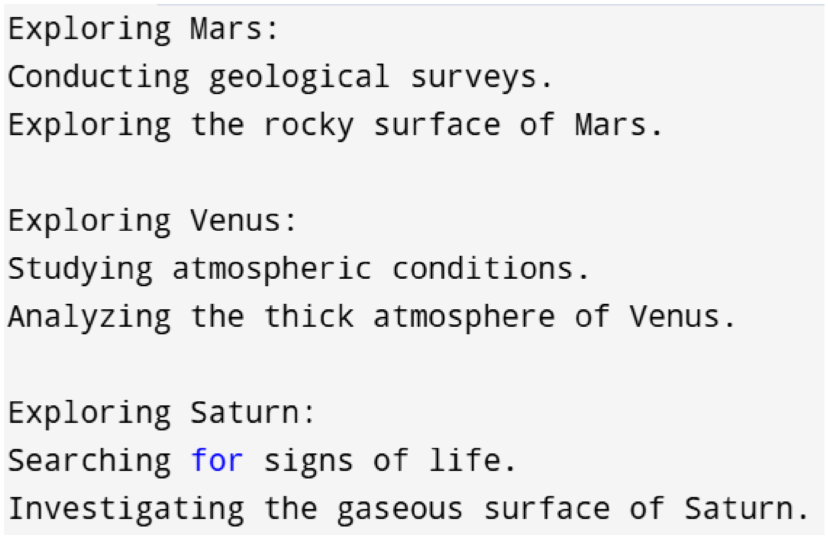


```
Enter the string :
HelloWorld
Enter start and end of subst :
2 7
Original: HelloWorld
Reversed: dlroWolleH
Character at index 5 (from reversed): o
Length of string: 10
Subsequence olleW
```

QUESTION 9:
A Planet Explorer routinely travels across the planets in the Solar System to discover life form, minerals available, etc. However, the method of exploring is different on each planet, due to the difference in atmosphere and surface composition. Every explorer should have an explore method that is defined based on the type of the explorer and the planet where (s)he is exploring. Consider three planets-Mars, Venus, and Saturn.
Implement it using interfaces, abstract class, inheritance.

CODE:
```java
interface Explorer {
    void explore();
}

abstract class Planet {
    protected String name;

    public Planet(String name) {
        this.name = name;
    }

    public abstract void accept(Explorer explorer);
}

class Mars extends Planet {
    public Mars() {
        super("Mars");
    }

    @Override
    public void accept(Explorer explorer) {
        explorer.explore();
        System.out.println("Exploring the rocky surface of Mars.");
    }
}

class Venus extends Planet {
    public Venus() {
        super("Venus");
    }

    @Override
    public void accept(Explorer explorer) {
        explorer.explore();
        System.out.println("Analyzing the thick atmosphere of Venus.");
    }
}

class Saturn extends Planet {
    public Saturn() {
        super("Saturn");
    }

    @Override
    public void accept(Explorer explorer) {
        explorer.explore();
        System.out.println("Investigating the gaseous surface of Saturn.");
    }
}

class GeologicalExplorer implements Explorer {
    @Override
    public void explore() {
        System.out.println("Conducting geological surveys.");
    }
}

class AtmosphericExplorer implements Explorer {
    @Override
    public void explore() {
        System.out.println("Studying atmospheric conditions.");
    }
}

class BiologicalExplorer implements Explorer {
    @Override
    public void explore() {
        System.out.println("Searching for signs of life.");
    }
}

public class program9 {
    public static void main(String[] args) {
        Planet mars = new Mars();
        Planet venus = new Venus();
        Planet saturn = new Saturn();

        Explorer geoExplorer = new GeologicalExplorer();
        Explorer atmoExplorer = new AtmosphericExplorer();
        Explorer bioExplorer = new BiologicalExplorer();

        System.out.println("Exploring Mars:");
        mars.accept(geoExplorer);

        System.out.println("\nExploring Venus:");
        venus.accept(atmoExplorer);

        System.out.println("\nExploring Saturn:");
        saturn.accept(bioExplorer);
    }
}
```

OUTPUT:



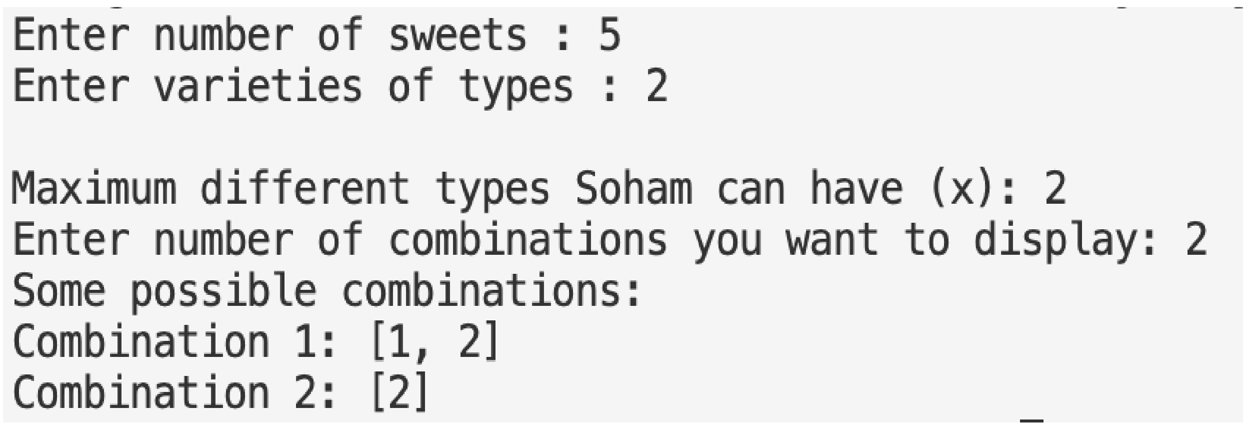
```
Exploring Mars:
Conducting geological surveys.
Exploring the rocky surface of Mars.

Exploring Venus:
Studying atmospheric conditions.
Analyzing the thick atmosphere of Venus.

Exploring Saturn:
Searching for signs of life.
Investigating the gaseous surface of Saturn.
```

Assignment 3

Q1. Soham has n sweets, where the ith sweet is of type “Sandesh” (say) while (i+1)th type is “Pithe” (say). So, the types need to be encoded by numerals for the assignment. He noticed that he started to gain weight. His classmate Arjeesh advised him to only eat n / 2 of the sweets and distribute the rest among friends. Soham still wishes to taste the maximum number of different types of
sweets while not ignoring his classmate’s suggestion.

```
import java.util.*;

class SweetDistribution {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter number of sweets : ");
      int n = sc.nextInt();
      System.out.print("Enter varieties of types : ");
      int k = sc.nextInt();

      List<Integer> sweets = new ArrayList<>();
      Random rand = new Random();
      for (int i = 0; i < n; i++) {
         sweets.add(rand.nextInt(k) + 1);
      }

      Set<Integer> set = new HashSet<>(sweets);
      int x = Math.min(set.size(), n / 2);

      System.out.println("\nMaximum different types Soham can have (x): " + x);
      System.out.print("Enter number of combinations you want to display: ");
      int y=sc.nextInt();
      List<Integer> typesList = new ArrayList<>(set);
```

```
        int max=typesList.size();

        if(y>max){
            System.err.println("Not possible");
            sc.close();
            System.exit(0);
        }
        System.out.println("Some possible combinations:");

        for (int i = 0; i < y; i++) {
            List<Integer> combination = new ArrayList<>();
            combination.add(typesList.get(i));
            for (int j = 1; j < x && j < typesList.size(); j++) {
                if (i != j) combination.add(typesList.get(j));
            }
            System.out.println("Combination " + (i+1) + ": " + combination);
        }
        sc.close();

    }
}
```

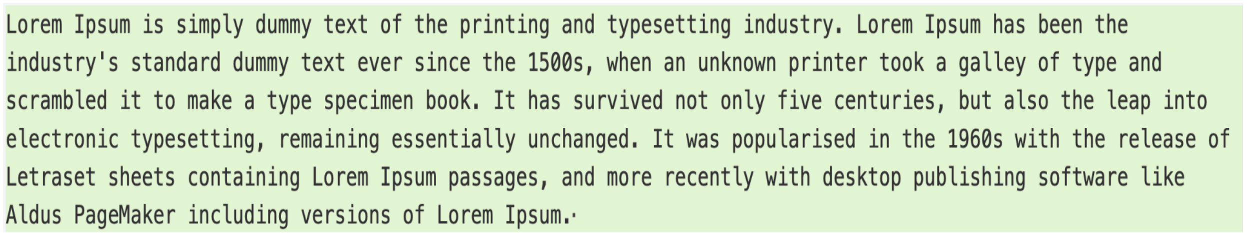


```
Enter number of sweets : 5
Enter varieties of types : 2

Maximum different types Soham can have (x): 2
Enter number of combinations you want to display: 2
Some possible combinations:
Combination 1: [1, 2]
Combination 2: [2]
```

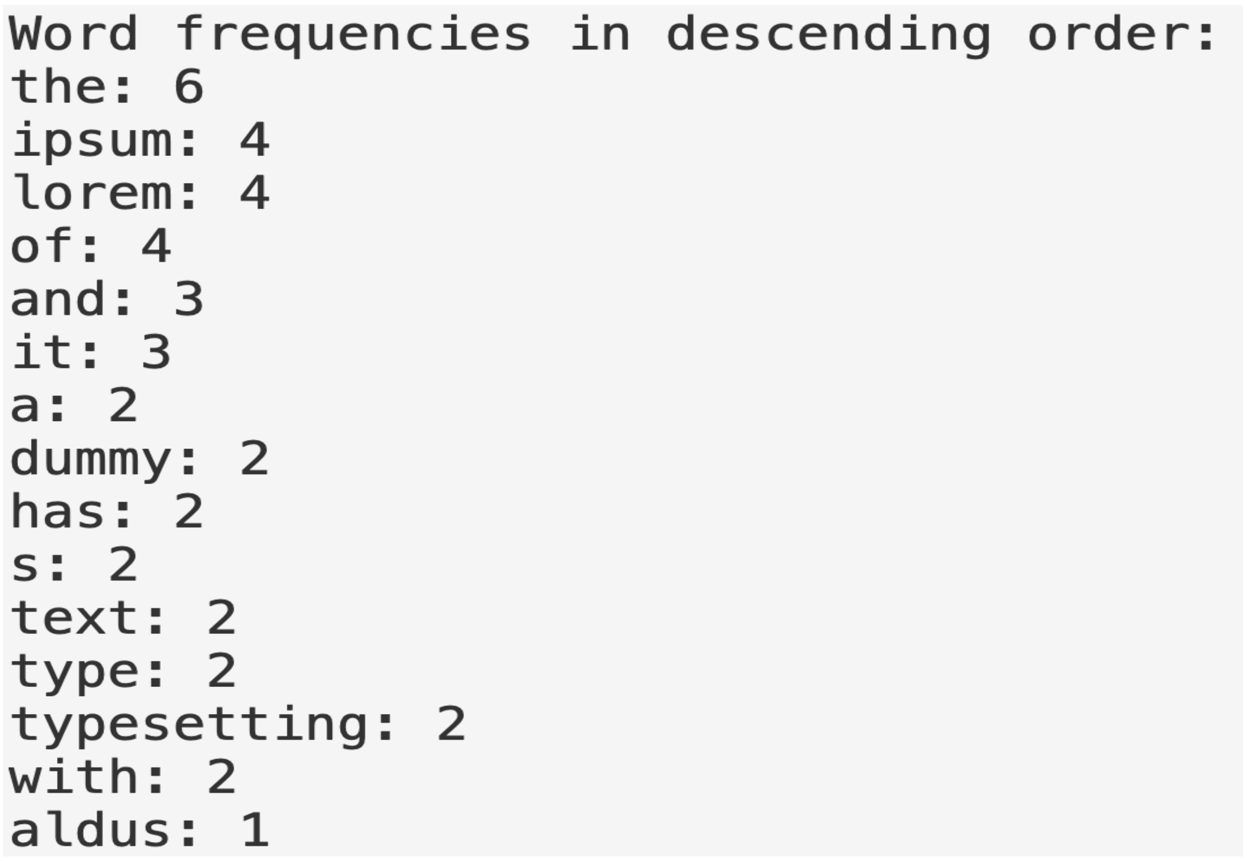
Q2. Extract the words and their frequencies from a text file. Then store them as key value pairs in a TreeMap. Sort them in *descending order using Comparator. Now, display the greatest and the least key value pairs from the collection.*

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

class WordFrequency {
    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(new FileReader("input.txt"));
        String line;
        Map<String, Integer> wordCount = new HashMap<>();

        while ((line = reader.readLine()) != null) {
            String[] words = line.split("\\s+");
            for (String word : words) {
                word = word.replaceAll("[^a-zA-Z]", "").toLowerCase();
                if (!word.isEmpty()) {
                    wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);
                }
            }
        }
        reader.close();

        TreeMap<String, Integer> sortedMap = new TreeMap<>(new
Comparator<String>() {
            @Override
            public int compare(String a, String b) {
                int freqCompare = wordCount.get(b).compareTo(wordCount.get(a));
                if (freqCompare != 0) return freqCompare;
                return a.compareTo(b);
            }
        });

        sortedMap.putAll(wordCount);

        System.out.println("Word frequencies in descending order:");
        for (Map.Entry<String, Integer> entry : sortedMap.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
        System.out.println("\nGreatest frequency: " + sortedMap.firstEntry());
        System.out.println("Least frequency: " + sortedMap.lastEntry());
    }
}
```

Input.txt:



```
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the
industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and
scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into
electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of
Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like
Aldus PageMaker including versions of Lorem Ipsum.
```

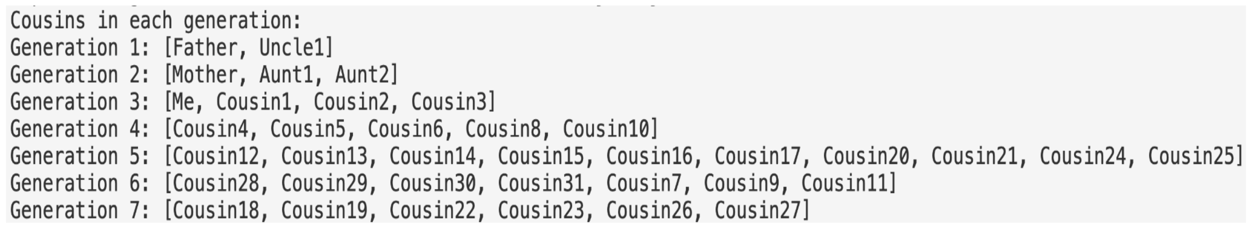


```
Word frequencies in descending order:
the: 6
ipsum: 4
lorem: 4
of: 4
and: 3
it: 3
a: 2
dummy: 2
has: 2
s: 2
text: 2
type: 2
typesetting: 2
with: 2
aldus: 1
```

Q3. Given preorder and inorder traversal outcomes of a family tree, construct and return the corresponding binary tree. If two nodes in the tree have the same depth then they are cousins. So, print the names of those who are cousins in generations of the family history. Try to use java collections classes and functions as much as you can.

```
import java.util.*;

class TreeNode {
   String name;
   TreeNode left, right;

   TreeNode(String name) {
      this.name = name;
   }
}

public class A3Q3 {
   private Map<Integer, List<String>> depthMap = new HashMap<>();

   public TreeNode buildTree(String[] preorder, String[] inorder) {
      return buildTreeHelper(preorder, 0, preorder.length - 1, inorder, 0, inorder.length -
1);
   }

   private TreeNode buildTreeHelper(String[] preorder, int preStart, int preEnd, String[]
inorder, int inStart, int inEnd) {
      if (preStart > preEnd || inStart > inEnd) return null;

      TreeNode root = new TreeNode(preorder[preStart]);
      int inIndex = 0;

      for (int i = inStart; i <= inEnd; i++) {
         if (inorder[i].equals(root.name)) {
            inIndex = i;
            break;
         }
      }
```

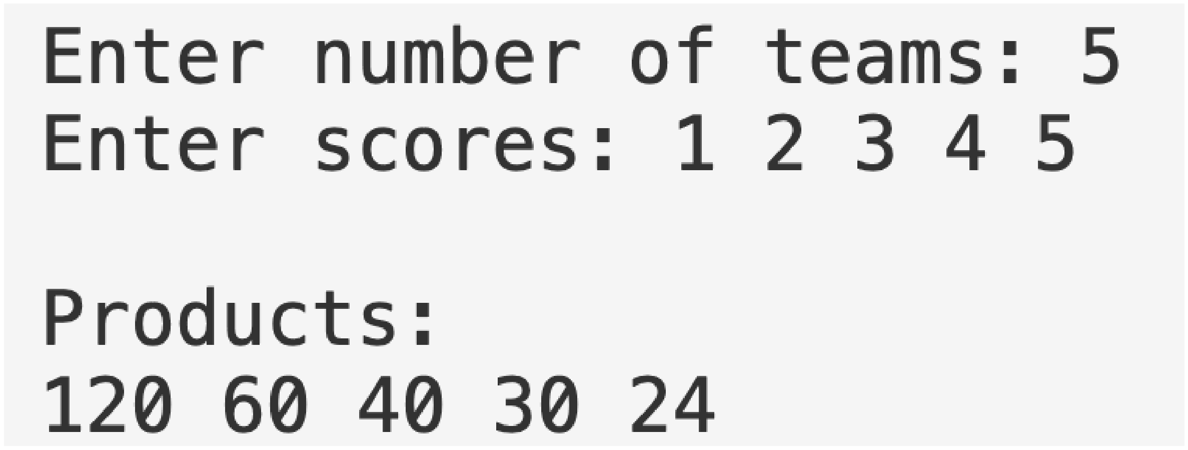
```
        root.left = buildTreeHelper(preorder, preStart + 1, preStart + (inIndex - inStart),
inorder, inStart, inIndex - 1);
        root.right = buildTreeHelper(preorder, preStart + (inIndex - inStart) + 1, preEnd,
inorder, inIndex + 1, inEnd);

        return root;
    }

    public void findCousins(TreeNode root) {
        traverseTree(root, 0);

        System.out.println("Cousins in each generation:");
        for (Map.Entry<Integer, List<String>> entry : depthMap.entrySet()) {
            if (entry.getValue().size() > 1) {
                System.out.println("Generation " + entry.getKey() + ": " + entry.getValue());
            }
        }
    }

    private void traverseTree(TreeNode node, int depth) {
        if (node == null) return;

        depthMap.putIfAbsent(depth, new ArrayList<>());
        depthMap.get(depth).add(node.name);

        traverseTree(node.left, depth + 1);
        traverseTree(node.right, depth + 1);
    }

    public static void main(String[] args) {
        String[] preorder = {"GrandFather", "Father", "Mother", "Me", "Cousin4",
"Cousin12", "Cousin28", "Cousin29", "Cousin13", "Cousin30", "Cousin31", "Cousin5",
"Cousin14", "Cousin15", "Aunt1", "Cousin1", "Cousin6", "Cousin16", "Cousin17",
"Cousin7", "Cousin18", "Cousin19", "Cousin2", "Cousin8", "Cousin20", "Cousin21",
"Cousin9", "Cousin22", "Cousin23", "Uncle1", "Aunt2", "Cousin3", "Cousin10",
"Cousin24", "Cousin25", "Cousin11", "Cousin26", "Cousin27"};

        String[] inorder = {"Cousin28", "Cousin12", "Cousin29", "Cousin4", "Cousin30",
"Cousin13", "Cousin31", "Me", "Cousin14", "Cousin5", "Cousin15", "Mother", "Father",
"Cousin16", "Cousin6", "Cousin17", "Cousin18", "Cousin7", "Cousin19", "Cousin1",
"Aunt1", "Cousin20", "Cousin8", "Cousin21", "Cousin22", "Cousin9", "Cousin23",
"Cousin2", "GrandFather", "Cousin24", "Cousin10", "Cousin25", "Cousin26", "Cousin11",
"Cousin27", "Cousin3", "Aunt2", "Uncle1"};

        A3Q3 tree = new A3Q3();
        TreeNode root = tree.buildTree(preorder, inorder);
        tree.findCousins(root);
    }
}
```



```
Cousins in each generation:
Generation 1: [Father, Uncle1]
Generation 2: [Mother, Aunt1, Aunt2]
Generation 3: [Me, Cousin1, Cousin2, Cousin3]
Generation 4: [Cousin4, Cousin5, Cousin6, Cousin8, Cousin10]
Generation 5: [Cousin12, Cousin13, Cousin14, Cousin15, Cousin16, Cousin17, Cousin20, Cousin21, Cousin24, Cousin25]
Generation 6: [Cousin28, Cousin29, Cousin30, Cousin31, Cousin7, Cousin9, Cousin11]
Generation 7: [Cousin18, Cousin19, Cousin22, Cousin23, Cousin26, Cousin27]
```

Q4. You have the number of goals scored by the football teams of the world cup for a year. Store them in a collection. Find the product of all the scores except itself. The product of any prefix or suffix is expected to fit in Java's int range. Otherwise, remove the very high scorers.
Write a program to display the products for the corresponding scores as input. The program should run in less than O(n^2) time and should not use '/'.

```
import java.util.*;

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

        System.out.print("\033[H\033[2J");
        System.out.flush();

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of teams: ");
        int n = sc.nextInt();
        int[] scores = new int[n];

        System.out.print("Enter scores: ");
        for (int i = 0; i < n; i++) {
            scores[i] = sc.nextInt();
        }

        sc.close();

        long[] result = new long[n];
        long left = 1;

        for (int i = 0; i < n; i++) {
            result[i] = left;
            left *= scores[i];
        }

        long right = 1;
        for (int i = n - 1; i >= 0; i--) {
            result[i] *= right;
            right *= scores[i];
        }

        System.out.println("\nProducts:");
        for (long num : result) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

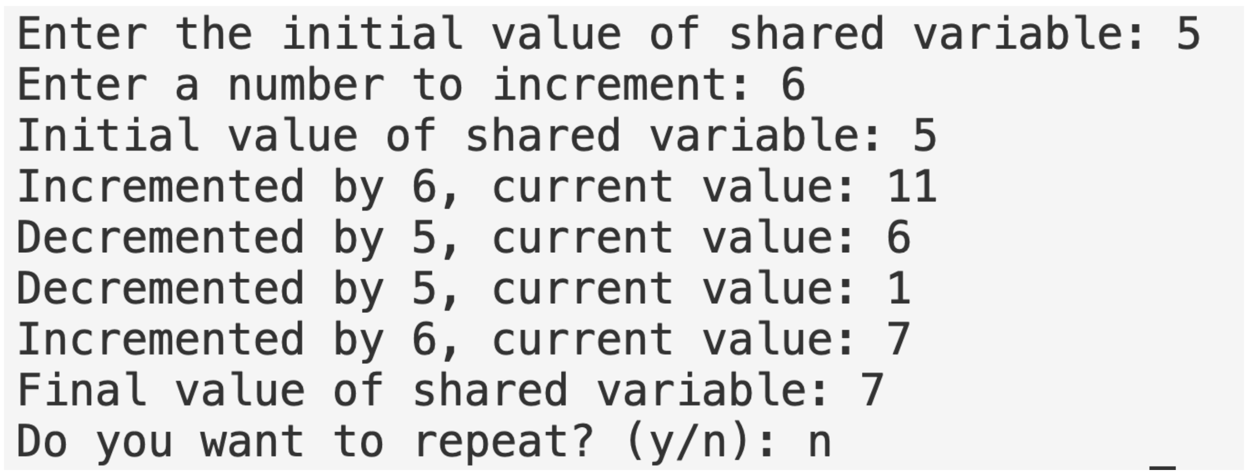


```
Enter number of teams: 5
Enter scores: 1 2 3 4 5

Products:
120 60 40 30 24
```

Q5. One thread takes an input from the user and increments an integer variable by that amount. Another thread reduces the variable by a fixed amount. Execute two versions of each thread simultaneously and all working on the same variable. Once all threads are over display the value of the variable. Repeat the threads unless different results emerge for repeated executions as a consequence of parallel programming. Don't use lambda functions.
Modify the problem stated in ensuring mutual exclusion on shared variables.

```
import java.util.Scanner;
public class A3Q5 {
   private static int sharedVariable;
   private static final Object lock = new Object();

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      while (true) {

         System.out.print("\033[H\033[2J");
         System.out.flush();
         System.out.print("Enter the initial value of shared variable: ");
         sharedVariable = scanner.nextInt();
```

```
            System.out.print("Enter a number to increment: ");
            int incrementValue = scanner.nextInt();
            System.out.println("Initial value of shared variable: " + sharedVariable);

            Thread incrementThread1 = new Thread(new IncrementTask(incrementValue));
            Thread incrementThread2 = new Thread(new IncrementTask(incrementValue));
            Thread decrementThread1 = new Thread(new DecrementTask());
            Thread decrementThread2 = new Thread(new DecrementTask());

            incrementThread1.start();
            incrementThread2.start();
            decrementThread1.start();
            decrementThread2.start();

            try {
                incrementThread1.join();
                incrementThread2.join();
                decrementThread1.join();
                decrementThread2.join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            System.out.println("Final value of shared variable: " + sharedVariable);

            System.out.print("Do you want to repeat? (y/n): ");
            String response = scanner.next();
            if (!response.equalsIgnoreCase("y")) {
                break;
            }
        }

        scanner.close();
    }

    static class IncrementTask implements Runnable {
        private final int incrementValue;

        public IncrementTask(int incrementValue) {
            this.incrementValue = incrementValue;
        }

        @Override
        public void run() {
            synchronized (lock) {
                sharedVariable += incrementValue;
                System.out.println("Incremented by " + incrementValue + ", current value: " +
sharedVariable);
            }
        }
    }

    static class DecrementTask implements Runnable {
        private static final int DECREMENT_VALUE = 5;

        @Override
        public void run() {
            synchronized (lock) {
                sharedVariable -= DECREMENT_VALUE;
                System.out.println("Decremented by " + DECREMENT_VALUE + ", current
value: " + sharedVariable);
            }
        }
    }
}
```

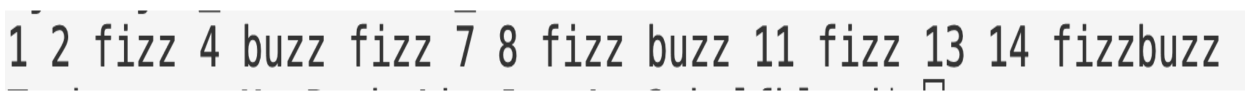


```
Enter the initial value of shared variable: 5
Enter a number to increment: 6
Initial value of shared variable: 5
Incremented by 6, current value: 11
Decremented by 5, current value: 6
Decremented by 5, current value: 1
Incremented by 6, current value: 7
Final value of shared variable: 7
Do you want to repeat? (y/n): n
```

Q6. You have the four functions:
• printFizz that prints the word "fizz" to the console,
• printBuzz that prints the word "buzz" to the console,
• printFizzBuzz that prints the word "fizzbuzz" to the console, and
• printNumber that prints a given integer to the console.
You are given an instance of the class FizzBuzz that has four functions: fizz, buzz, fizzbuzz and number. The same instance of FizzBuzz will be passed to
four different threads:
• Thread A: calls fizz() that should output the word "fizz".
• Thread B: calls buzz() that should output the word "buzz".
• Thread C: calls fizzbuzz() that should output the word "fizzbuzz".
• Thread D: calls number() that should only output the integers.
Repeat the threads unless different results emerge for repeated executions as a consequence of parallel programming. You may use lambda function for only creating Runnable objects.

```
import java.util.function.IntConsumer;

class FizzBuzz {
    private int n;
    private int current = 1;

    public FizzBuzz(int n) {
        this.n = n;
    }

    public synchronized void fizz(Runnable printFizz) throws InterruptedException {
        while (current <= n) {
            if (current % 3 == 0 && current % 5 != 0) {
                printFizz.run();
                current++;
                notifyAll();
            } else {
```

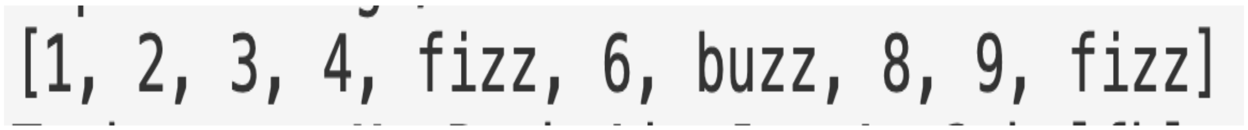
```
            wait();
        }
    }
}

public synchronized void buzz(Runnable printBuzz) throws InterruptedException {
    while (current <= n) {
        if (current % 5 == 0 && current % 3 != 0) {
            printBuzz.run();
            current++;
            notifyAll();
        } else {
            wait();
        }
    }
}

public synchronized void fizzbuzz(Runnable printFizzBuzz) throws
InterruptedException {
    while (current <= n) {
        if (current % 15 == 0) {
            printFizzBuzz.run();
            current++;
            notifyAll();
        } else {
            wait();
        }
    }
}

public synchronized void number(IntConsumer printNumber) throws
InterruptedException {
    while (current <= n) {
        if (current % 3 != 0 && current % 5 != 0) {
            printNumber.accept(current);
            current++;
            notifyAll();
        } else {
            wait();
        }
    }
}
}

class FizzBuzzThreads {
public static void main(String[] args) {
    FizzBuzz fizzBuzz = new FizzBuzz(15);

    Thread threadA = new Thread(() -> {
        try {
            fizzBuzz.fizz(() -> System.out.print("fizz "));
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    });

    Thread threadB = new Thread(() -> {
        try {
            fizzBuzz.buzz(() -> System.out.print("buzz "));
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    });

    Thread threadC = new Thread(() -> {
        try {
            fizzBuzz.fizzbuzz(() -> System.out.print("fizzbuzz "));
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    });

    Thread threadD = new Thread(() -> {
        try {
            fizzBuzz.number(number -> System.out.print(number + " "));
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    });
    threadA.start();
    threadB.start();
    threadC.start();
    threadD.start();

    try {
        threadA.join();
        threadB.join();
        threadC.join();
        threadD.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println();
}
}
```



```
1 2 fizz 4 buzz fizz 7 8 fizz buzz 11 fizz 13 14 fizzbuzz
```

Q7. Modify the given class so that the ith token (1-indexed) of the series is:
• "fizzbuzz" if i is divisible by 5 and 7,
• "fizz" if i is divisible by 5 and not 9,
• "buzz" if i is divisible by 7 and not 3, or
• i if i is not divisible by 7 or 5.
If n=10 then the output will be [1,2,3,4,fizz,6,buzz,8,9,fizz]. You may use lambda function for only creating Runnable objects.

```
import java.util.*;

class ModifiedFizzBuzz {
    public static void main(String[] args) {
        int n = 10;
        String[] result = new String[n];
        Runnable fizzBuzzLogic = () -> {
            for (int i = 1; i <= n; i++) {
                if (i % 5 == 0 && i % 7 == 0) {
                    result[i - 1] = "fizzbuzz";
                } else if (i % 5 == 0 && i % 9 != 0) {
                    result[i - 1] = "fizz";
                } else if (i % 7 == 0 && i % 3 != 0) {
                    result[i - 1] = "buzz";
                } else {
                    result[i - 1] = "" + i;
                }
            }
        };

        fizzBuzzLogic.run();
        System.out.println(Arrays.toString(result));
    }
}
```

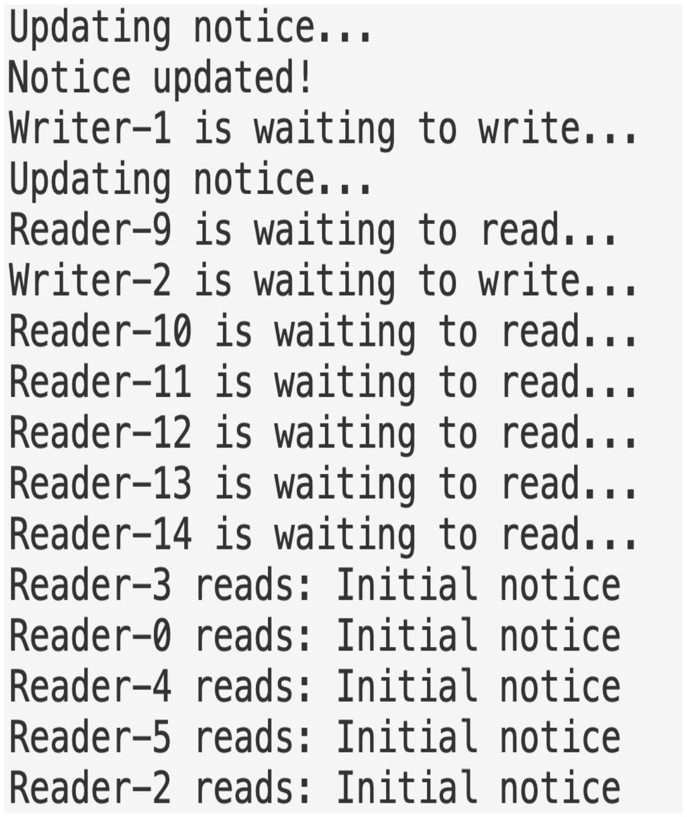


```
[1, 2, 3, 4, fizz, 6, buzz, 8, 9, fizz]
```

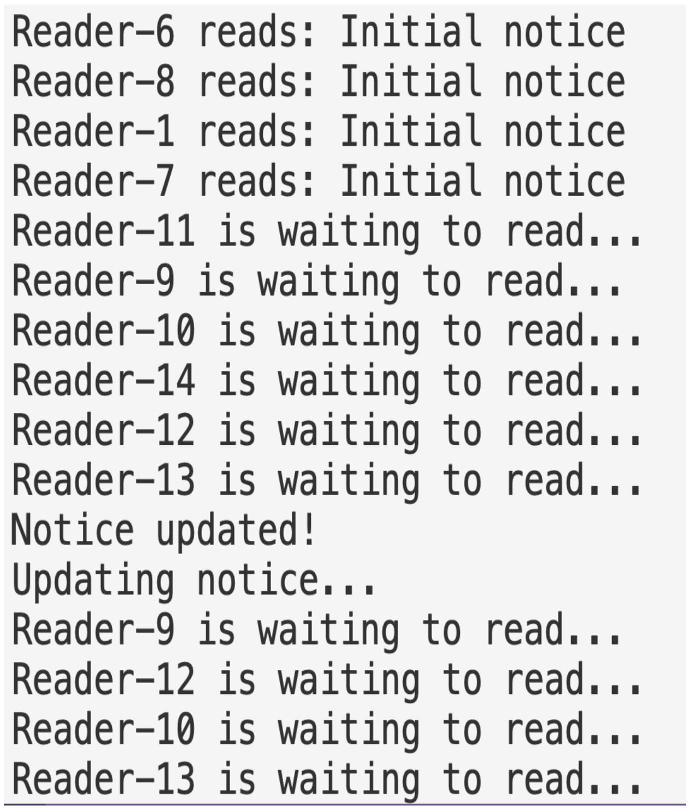
*Q8. You are asked to create a notice board. Anybody can read the notice board parallelly but when any one is writing on it, all others should wait for the modification to finish. So, the waiting threads would display a suitable message to indicate that it is waiting for the update
to finish.*

```
import java.util.concurrent.locks.*;

class NoticeBoard {
    private String notice;
    private ReadWriteLock lock = new ReentrantReadWriteLock();

    public String readNotice() {
        while (!lock.readLock().tryLock()) {
            System.out.println(Thread.currentThread().getName() + " is waiting to read...");
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        try {
            return notice;
        } finally {
            lock.readLock().unlock();
        }
    }

    public void writeNotice(String newNotice) {
        lock.writeLock().lock();
        try {
            System.out.println("Updating notice...");
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            notice = newNotice;
            System.out.println("Notice updated!");
        } finally {
            lock.writeLock().unlock();
        }
    }
}

class Reader implements Runnable {
    private NoticeBoard board;

    public Reader(NoticeBoard board) {
        this.board = board;
    }

    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName() + " reads: " +
board.readNotice());
    }
}

class Writer implements Runnable {
    private NoticeBoard board;
    private String notice;

    public Writer(NoticeBoard board, String notice) {
        this.board = board;
        this.notice = notice;
    }

    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName() + " is waiting to write...");
        board.writeNotice(notice);
    }
}

class NoticeBoardDemo {
    public static void main(String[] args) {
        NoticeBoard board = new NoticeBoard();
        board.writeNotice("Initial notice");

        for (int i = 0; i < 5; i++) {
            new Thread(new Reader(board), "Reader-" + i).start();
        }

        new Thread(new Writer(board, "Updated notice 1"), "Writer-1").start();

        for (int i = 5; i < 10; i++) {
            new Thread(new Reader(board), "Reader-" + i).start();
        }
```

```java
        new Thread(new Writer(board, "Updated notice 2"), "Writer-2").start();

        for (int i = 10; i < 15; i++) {
            new Thread(new Reader(board), "Reader-" + i).start();
        }
    }
}
```



```
Updating notice...
Notice updated!
Writer-1 is waiting to write...
Updating notice...
Reader-9 is waiting to read...
Writer-2 is waiting to write...
Reader-10 is waiting to read...
Reader-11 is waiting to read...
Reader-12 is waiting to read...
Reader-13 is waiting to read...
Reader-14 is waiting to read...
Reader-3 reads: Initial notice
Reader-0 reads: Initial notice
Reader-4 reads: Initial notice
Reader-5 reads: Initial notice
Reader-2 reads: Initial notice
```



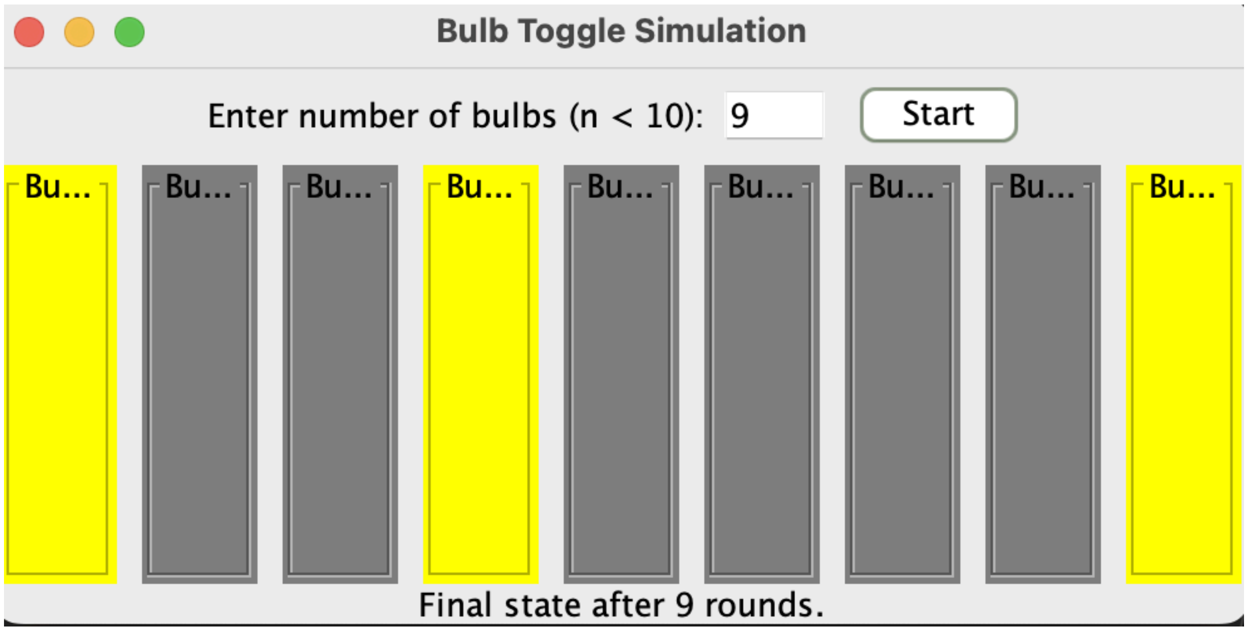
```
Reader-6 reads: Initial notice
Reader-8 reads: Initial notice
Reader-1 reads: Initial notice
Reader-7 reads: Initial notice
Reader-11 is waiting to read...
Reader-9 is waiting to read...
Reader-10 is waiting to read...
Reader-14 is waiting to read...
Reader-12 is waiting to read...
Reader-13 is waiting to read...
Notice updated!
Updating notice...
Reader-9 is waiting to read...
Reader-12 is waiting to read...
Reader-10 is waiting to read...
Reader-13 is waiting to read...
```

Assignment 4

Q1. There are n bulbs that are initially off. You first turn on all the bulbs, then you turn off every second bulb. On the third round, you toggle every third bulb (turning on if it's off or turning off if it's on). For the i-th round, you toggle every i bulb. For the nth round, you only
toggle the last bulb. Design a Java program using Swing to show the result of every round where n is given by the
user. You may limit it to n<10 for better clarity.

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

public class A4Q1 extends JFrame {
    private JPanel bulbsPanel;
    private JTextField inputField;
    private JButton startButton;
    private JLabel roundLabel;
    private int n;
    private boolean[] bulbs;
    private int round;

    public A4Q1() {
        setTitle("Bulb Toggle Simulation");
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        setLayout(new BorderLayout());

        JPanel topPanel = new JPanel();
        topPanel.add(new JLabel("Enter number of bulbs (n < 10):"));
        inputField = new JTextField(3);
        topPanel.add(inputField);
        startButton = new JButton("Start");
        topPanel.add(startButton);
        add(topPanel, BorderLayout.NORTH);
    bulbsPanel = new JPanel();
    add(bulbsPanel, BorderLayout.CENTER);

    roundLabel = new JLabel(" ");
    roundLabel.setHorizontalAlignment(SwingConstants.CENTER);
    add(roundLabel, BorderLayout.SOUTH);

    startButton.addActionListener(e -> startSimulation());

    setSize(500, 250);
    setLocationRelativeTo(null);
    setVisible(true);
}

private void startSimulation() {
    try {
        n = Integer.parseInt(inputField.getText());
        if (n < 1 || n >= 10) {
            JOptionPane.showMessageDialog(this, "Please enter n between 1 and 9.");
            return;
        }
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(this, "Invalid input.");
        return;
    }
    bulbs = new boolean[n];
    round = 0;
    for (int i = 0; i < n; i++) bulbs[i] = false;
    nextRound();
}

private void nextRound() {
    if (round == 0) {
        for (int i = 0; i < n; i++) bulbs[i] = true;
    } else {
        int step = round + 1;
        for (int i = step - 1; i < n; i += step) {
            bulbs[i] = !bulbs[i];
        }
    }
```

```
        updateBulbsPanel();
        roundLabel.setText("After round " + (round + 1));
        round++;
        if (round < n) {
            Timer timer = new Timer(1200, new ActionListener() {
                public void actionPerformed(ActionEvent evt) {
                    ((Timer) evt.getSource()).stop();
                    nextRound();
                }
            });
            timer.setRepeats(false);
            timer.start();
        } else {
            roundLabel.setText("Final state after " + n + " rounds.");
        }
    }

    private void updateBulbsPanel() {
        bulbsPanel.removeAll();
        bulbsPanel.setLayout(new GridLayout(1, n, 10, 10));
        for (int i = 0; i < n; i++) {
            JPanel bulb = new JPanel();
            bulb.setPreferredSize(new Dimension(40, 40));
            bulb.setBackground(bulbs[i] ? Color.YELLOW : Color.GRAY);
            bulb.setBorder(BorderFactory.createTitledBorder("Bulb " + (i + 1)));
            bulbsPanel.add(bulb);
        }
        bulbsPanel.revalidate();
        bulbsPanel.repaint();
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(A4Q1::new);
    }
}
```

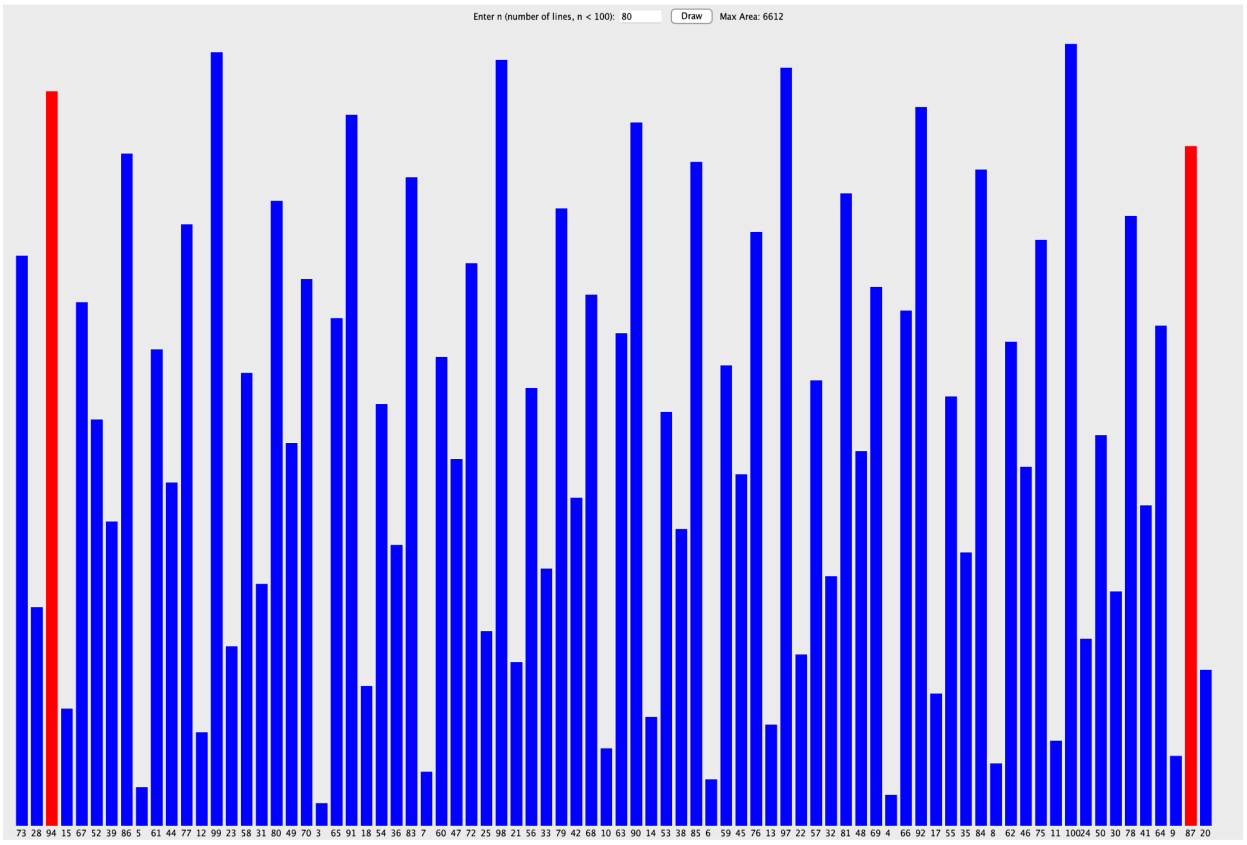


Q2. Consider the problem from Assignment 2 again.
An integer array representing height of length n is given as input. There are n vertical lines drawn from the array such that the two endpoints of the ith line are (i, 0) and (i, height[i]).
Any two lines along with the x-axis form a container that can hold water. Find the largest container that can hold maximum water.
Write a Java program using Swing where the no. of lines(n) to consider are taken through a textbox or any suitable GUI widget and the final result will be shown graphically. The heights of the lines could be read from a file containing say, 100 different heights where n<100.

```
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.util.*;

public class A4Q2 extends JFrame {
```

```
private int[] heights = new int[100];
private int n = 0;
private int maxArea = 0;
private int leftIndex = 0, rightIndex = 0;

private JTextField nField = new JTextField(5);
private JButton drawButton = new JButton("Draw");
private JLabel areaLabel = new JLabel("Max Area: ");
private JPanel drawPanel = new JPanel() {
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (n == 0) return;
        int width = getWidth();
        int height = getHeight();
        int barWidth = width / (n + 2);
        int maxH = Arrays.stream(heights).limit(n).max().orElse(1);

        for (int i = 0; i < n; i++) {
            int barHeight = (int)((heights[i] / (double)maxH) * (height - 40));
            if (i == leftIndex || i == rightIndex) {
                g.setColor(Color.RED);
            } else {
                g.setColor(Color.BLUE);
            }
            g.fillRect(20 + i * barWidth, height - barHeight - 20, barWidth - 5, barHeight);
            g.setColor(Color.BLACK);
            g.drawString(String.valueOf(heights[i]), 20 + i * barWidth, height - 5);
        }
    }
};

public A4Q2() {
    setTitle("Max Water Container");
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setSize(800, 400);
    setLayout(new BorderLayout());
    setExtendedState(JFrame.MAXIMIZED_BOTH);
    setUndecorated(true);
    try (BufferedReader br = new BufferedReader(new FileReader("heights.txt"))) {
        String line;
```

```
            int idx = 0;
            while ((line = br.readLine()) != null && idx < 100) {
                heights[idx++] = Integer.parseInt(line.trim());
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error reading heights.txt", "Error",
JOptionPane.ERROR_MESSAGE);
        }

        JPanel topPanel = new JPanel();
        topPanel.add(new JLabel("Enter n (number of lines, n < 100):"));
        topPanel.add(nField);
        topPanel.add(drawButton);
        topPanel.add(areaLabel);

        add(topPanel, BorderLayout.NORTH);
        add(drawPanel, BorderLayout.CENTER);

        drawButton.addActionListener(e -> {
            try {
                n = Integer.parseInt(nField.getText());
                if (n <= 1 || n > 100) throw new NumberFormatException();
                findMaxArea();
                areaLabel.setText("Max Area: " + maxArea);
                drawPanel.repaint();
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(this, "Please enter a valid n (2 <= n <=
100)", "Input Error", JOptionPane.ERROR_MESSAGE);
            }
        });
    }

    private void findMaxArea() {
        maxArea = 0;
        leftIndex = 0;
        rightIndex = 0;
        int l = 0, r = n - 1;
        while (l < r) {
            int area = Math.min(heights[l], heights[r]) * (r - l);
            if (area > maxArea) {
                maxArea = area;
                leftIndex = l;
                rightIndex = r;
            }
            if (heights[l] < heights[r]) l++;
            else r--;
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new A4Q2().setVisible(true));
    }
}
```

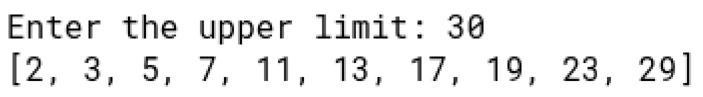


Python Assignment 1

1. **Write a prime generator program using only primes and using python loops.**

```
def generate_primes(limit):
   primes = []
   for num in range(2, limit + 1):
      is_prime = True
      for prime in primes:
         if prime * prime > num:
            break
         if num % prime == 0:
            is_prime = False
            break
      if is_prime:
         primes.append(num)
   return primes


n = int(input("Enter the upper limit: "))
prime_list = generate_primes(n)
print(prime_list)
```

Output:

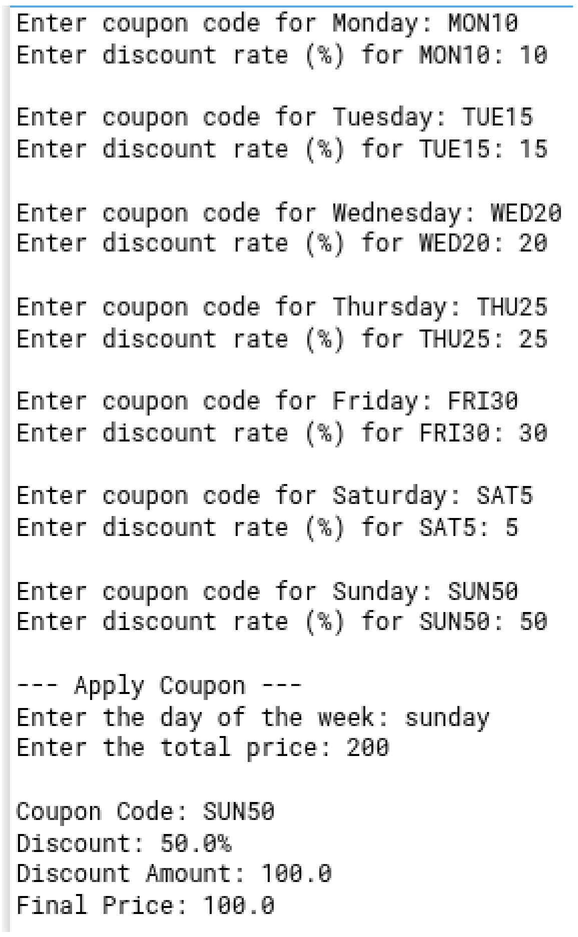


```
Enter the upper limit: 30
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

2. **Write a discount coupon code using dictionary in Python with different rate coupons for each day of the week.**

```
coupons = {}
rates = {}

days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"]

for day in days:
    code = input(f"Enter coupon code for {day}: ")
    rate = float(input(f"Enter discount rate (%) for {code}: "))
    coupons[day] = code
    rates[code] = rate

day = input("Enter the day of the week: ").capitalize()
price = float(input("Enter the total price: "))

if day in coupons:
    code = coupons[day]
    discount_percent = rates[code]
    discount_amount = (discount_percent / 100) * price
    final_price = price - discount_amount
    print(f"Coupon Code: {code}")
    print(f"Discount: {discount_percent}%")
    print(f"Discount Amount: {discount_amount}")
    print(f"Final Price: {final_price}")
else:
    print("Invalid day entered.")
```

OUTPUT:

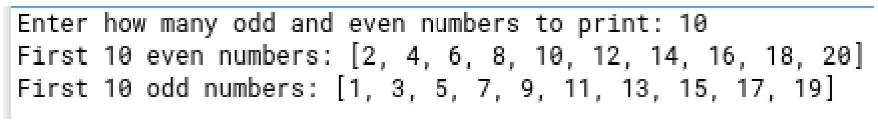

```
Enter coupon code for Monday: MON10
Enter discount rate (%) for MON10: 10

Enter coupon code for Tuesday: TUE15
Enter discount rate (%) for TUE15: 15

Enter coupon code for Wednesday: WED20
Enter discount rate (%) for WED20: 20

Enter coupon code for Thursday: THU25
Enter discount rate (%) for THU25: 25

Enter coupon code for Friday: FRI30
Enter discount rate (%) for FRI30: 30

Enter coupon code for Saturday: SAT5
Enter discount rate (%) for SAT5: 5

Enter coupon code for Sunday: SUN50
Enter discount rate (%) for SUN50: 50

--- Apply Coupon ---
Enter the day of the week: sunday
Enter the total price: 200

Coupon Code: SUN50
Discount: 50.0%
Discount Amount: 100.0
Final Price: 100.0
```

3. **Print first 10 odd and even numbers using iterators and compress. You can use duck typing.**

```
from itertools import compress


class NumIterator:
    def __init__(self, start=1):
        self.num = start


    def __iter__(self):
        return self


    def __next__(self):
        current = self.num
        self.num += 1
        return current
```

```
n = int(input("Enter how many odd and even numbers to print: "))


iterator = NumIterator()
numbers = []
while len(numbers) < 2 * n:
   numbers.append(next(iterator))


# Create selectors for even and odd
selectors_even = [(x % 2 == 0) for x in numbers]
selectors_odd = [(x % 2 != 0) for x in numbers]


even_numbers = list(compress(numbers, selectors_even))[:n]
odd_numbers = list(compress(numbers, selectors_odd))[:n]


print("First", n, "even numbers:", even_numbers)
print("First", n, "odd numbers:", odd_numbers)
```

OUTPUT:

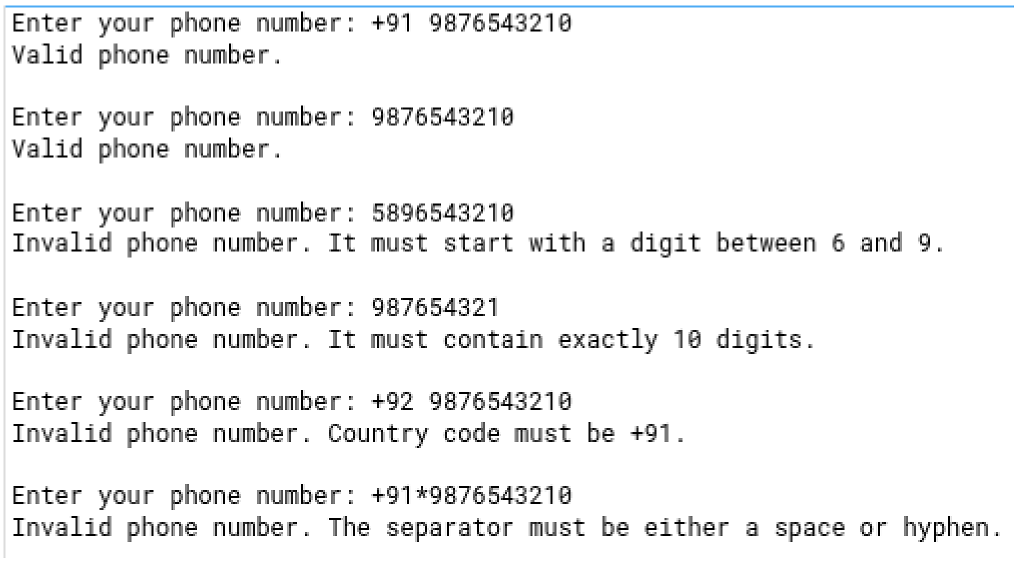


```
Enter how many odd and even numbers to print: 10
First 10 even numbers: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
First 10 odd numbers: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

4. **Write a regular expression to validate a phone number.**

```
import re

def validate_phone_number(phone):
   pattern = r'^(?:\+91|91)?[-\s]?([6-9]\d{9})$'

   if not re.fullmatch(pattern, phone):
      if not re.match(r'^\d{10}$', phone):
            print("Invalid phone number. It must contain exactly 10 digits.")
        elif not re.match(r'^[6-9]', phone):
            print("Invalid phone number. It must start with a digit between 6 and 9.")
        elif "+" in phone and not phone.startswith("+91"):
            print("Invalid phone number. Country code must be +91.")
        elif "-" in phone or " " in phone:
            print("Invalid phone number. The separator must be either a space or
hyphen, but it is used incorrectly.")
        else:
            print("Invalid phone number. Please check the format.")
    else:
        print("Valid phone number.")


phone = input("Enter your phone number: ")
validate_phone_number(phone)
```

OUTPUT:

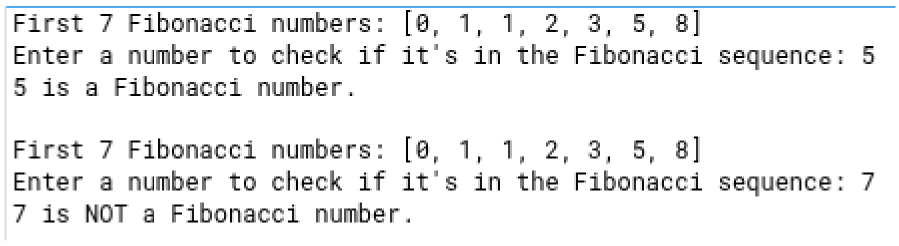


```
Enter your phone number: +91 9876543210
Valid phone number.

Enter your phone number: 9876543210
Valid phone number.

Enter your phone number: 5896543210
Invalid phone number. It must start with a digit between 6 and 9.

Enter your phone number: 987654321
Invalid phone number. It must contain exactly 10 digits.

Enter your phone number: +92 9876543210
Invalid phone number. Country code must be +91.

Enter your phone number: +91*9876543210
Invalid phone number. The separator must be either a space or hyphen.
```

5. **Write first seven Fibonacci numbers using generator next function/ yield in python. Trace and memorize the function. Also check whether a user given number is Fibinacci or not.**

```
def fibonacci_generator():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b


def is_fibonacci(n):
    if n < 0:
        return False
    fib_gen = fibonacci_generator()
    fib_number = next(fib_gen)
    while fib_number < n:
        fib_number = next(fib_gen)
    return fib_number == n




fib_gen = fibonacci_generator()
first_seven_fib = [next(fib_gen) for _ in range(7)]
print("First 7 Fibonacci numbers:", first_seven_fib)


user_number = int(input("Enter a number to check if it's in the Fibonacci
sequence: "))
if is_fibonacci(user_number):
    print(f"{user_number} is a Fibonacci number.")
else:
    print(f"{user_number} is NOT a Fibonacci number.")
```

OUTPUT:



```
First 7 Fibonacci numbers: [0, 1, 1, 2, 3, 5, 8]
Enter a number to check if it's in the Fibonacci sequence: 5
5 is a Fibonacci number.

First 7 Fibonacci numbers: [0, 1, 1, 2, 3, 5, 8]
Enter a number to check if it's in the Fibonacci sequence: 7
7 is NOT a Fibonacci number.
```

6. **Write a simple program which loops over a list of user data (tuples containing a username, email and age) and adds each user to a directory if the user is at least 16 years old. You do not need to store the age. Write a simple exception hierarchy which defines a different exception for each of these error conditions:**

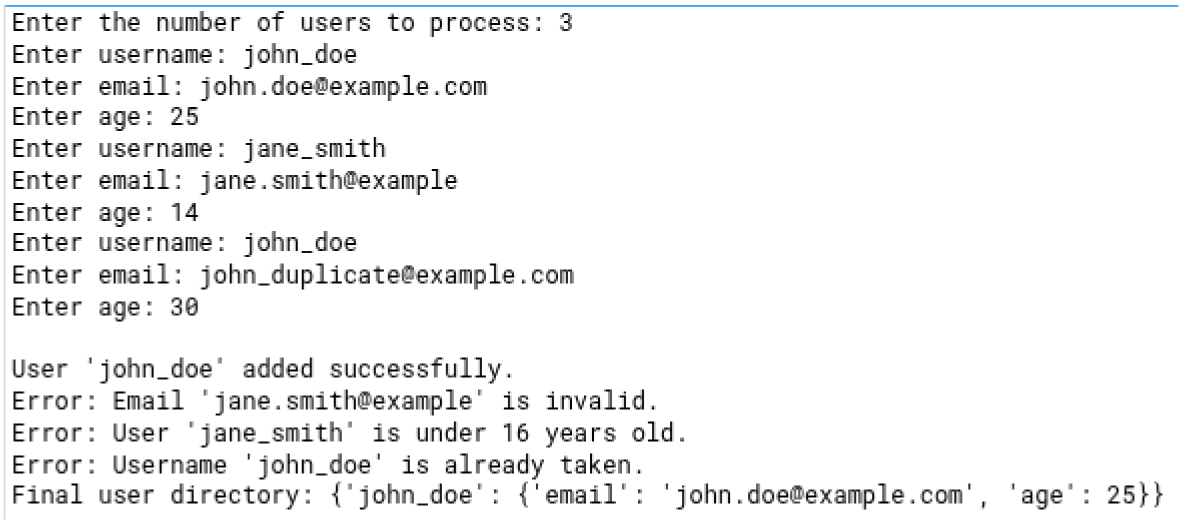
- **the username is not unique**
- **the age is not a positive integer**
- **the user is under 16**
- **the email address is not valid (a simple check for a username, the @ symbol and a domain name is sufficient)**

Raise these exceptions in your program where appropriate. Whenever an exception occurs, your program should move onto the next set of data in the list. Print a different error message for each different kind of exception.

```
import re

class UserException(Exception):
    """Base class for other exceptions."""
    pass


class DuplicateUsernameError(UserException):
    """Raised when the username is not unique."""
    pass


class InvalidAgeError(UserException):
    """Raised when the age is not a positive integer or the user is under 16."""
    pass


class InvalidEmailError(UserException):
    """Raised when the email is not valid."""
    pass
```

```
def is_valid_email_format(email):
    """Validates email format using regex."""
    email_regex = r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'
    return re.match(email_regex, email) is not None


def register_user(user_info, user_directory):
    username, email, age = user_info


    if username in user_directory:
        raise DuplicateUsernameError(f"Error: Username '{username}' is already
taken.")


    if not isinstance(age, int) or age <= 0:
        raise InvalidAgeError(f"Error: Age '{age}' is not a valid positive integer.")
    if age < 16:
        raise InvalidAgeError(f"Error: User '{username}' is under 16 years old.")


    if not is_valid_email_format(email):
        raise InvalidEmailError(f"Error: Email '{email}' is invalid.")


    user_directory[username] = {'email': email, 'age': age}


user_directory = {}


total_users = int(input("Enter the number of users to process: "))


for _ in range(total_users):
    input_username = input("Enter username: ")
    input_email = input("Enter email: ")
    input_age = int(input("Enter age: "))
    user_info = (input_username, input_email, input_age)


    try:
        register_user(user_info, user_directory)
        print(f"User '{input_username}' added successfully.")
    except UserException as error:
        print(error)


print("\nFinal user directory:", user_directory)
```

OUTPUT:

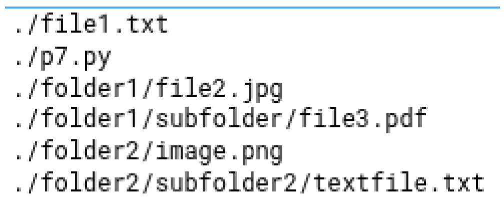


```
Enter the number of users to process: 3
Enter username: john_doe
Enter email: john.doe@example.com
Enter age: 25
Enter username: jane_smith
Enter email: jane.smith@example
Enter age: 14
Enter username: john_doe
Enter email: john_duplicate@example.com
Enter age: 30

User 'john_doe' added successfully.
Error: Email 'jane.smith@example' is invalid.
Error: User 'jane_smith' is under 16 years old.
Error: Username 'john_doe' is already taken.
Final user directory: {'john_doe': {'email': 'john.doe@example.com', 'age': 25}}
```

7. **Write a function *findfiles* that recursively descends the directory tree for the specified directory and generates paths of all the files in the tree.**

```
import os
def findfiles(directory):
    for dirpath, dirnames, filenames in os.walk(directory):
        for filename in filenames:
            file_path = os.path.join(dirpath, filename)
            yield file_path

directory = input("Enter the directory path to search for files: ")

for file in findfiles(directory):
    print(file)
```

OUTPUT:

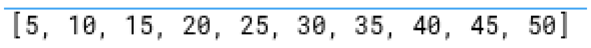


```
./file1.txt
./p7.py
./folder1/file2.jpg
./folder1/subfolder/file3.pdf
./folder2/image.png
./folder2/subfolder2/textfile.txt
```

8. **Create a list of all the numbers up to N=50 which are multiples of five using anonymous function.**

```
N = 50
multiples_of_five = list(filter(lambda x: x % 5 == 0, range(1, N+1)))
print(multiples_of_five)
```

OUTPUT:



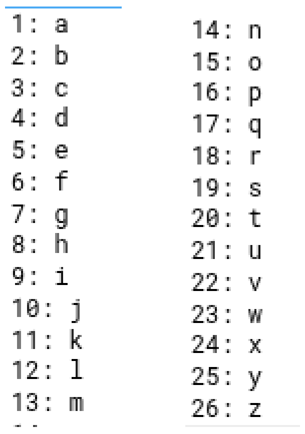
```
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
```

9. **Enumerate the sequence of all lowercase ASCII letters, starting from 1, using**

   enumerate.

```
lowercase_letters = 'abcdefghijklmnopqrstuvwxyz'
for index, letter in enumerate(lowercase_letters, start=1):
    print(f"{index}: {letter}")
```

OUTPUT:



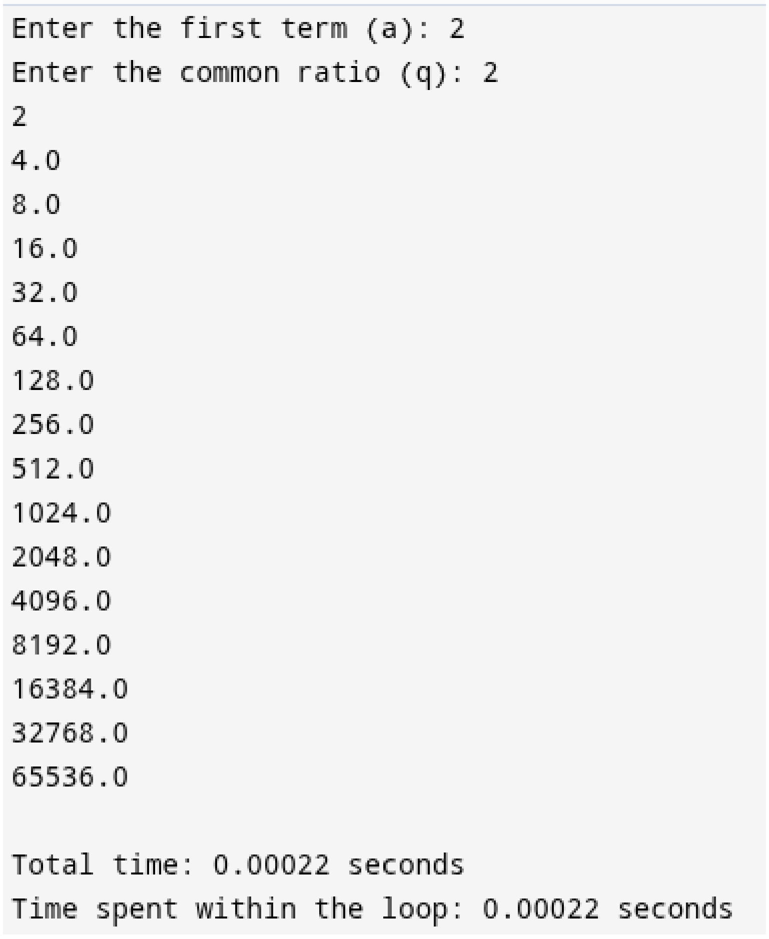
```
1: a        14: n
2: b        15: o
3: c        16: p
4: d        17: q
5: e        18: r
6: f        19: s
7: g        20: t
8: h        21: u
9: i        22: v
10: j       23: w
11: k       24: x
12: l       25: y
13: m       26: z
```

10. **Write a code which yields all terms of the geometric progression a, aq, aq2 , aq3 , ...**

When the progression produces a term that is greater than 100,000, the generator stops (with a return statement). Compute total time and time within the loop.

```
import time

def geometric_progression(a, q):
    term = a
    index = 0
    while term <= 100000:
        yield term
        index += 1
        term = a * q ** index
    return "Stopped as term exceeded 100,000"


def compute_time():
    a = int(input("Enter the first term (a): "))
    q = float(input("Enter the common ratio (q): "))
    start_time = time.time()


    total_loop_time_start = time.time()

    gen = geometric_progression(a, q)

    for term in gen:
        print(term)

    total_loop_time_end = time.time()
    total_time_end = time.time()

    total_time = total_time_end - start_time
    loop_time = total_loop_time_end - total_loop_time_start


    print(f"\nTotal time: {total_time:.5f} seconds")
    print(f"Time spent within the loop: {loop_time:.5f} seconds")


compute_time()
```

OUTPUT:



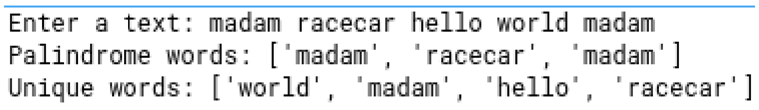
```
Enter the first term (a): 2
Enter the common ratio (q): 2
2
4.0
8.0
16.0
32.0
64.0
128.0
256.0
512.0
1024.0
2048.0
4096.0
8192.0
16384.0
32768.0
65536.0

Total time: 0.00022 seconds
Time spent within the loop: 0.00022 seconds
```

11. **Search for palindrome and unique words in a text using class method and string method.**

```
class TextProcessor:

  @staticmethod
  def is_palindrome(word):
    return word == word[::-1]


  @staticmethod
  def find_palindromes(text):
    words = text.split()
    return [word for word in words if TextProcessor.is_palindrome(word)]


  @staticmethod
```

```
    def find_unique_words(text):
        words = text.split()
        return list(set(words))


# Example usage
text = input("Enter a text: ")


palindromes = TextProcessor.find_palindromes(text)
unique_words = TextProcessor.find_unique_words(text)


print(f"Palindrome words: {palindromes}")
print(f"Unique words: {unique_words}")
```

OUTPUT:



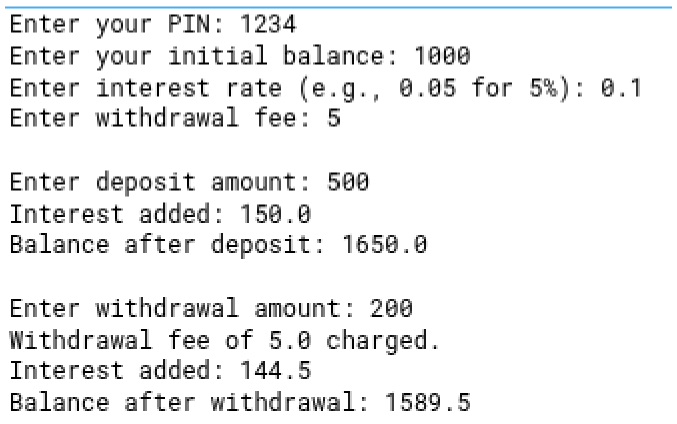
```
Enter a text: madam racecar hello world madam
Palindrome words: ['madam', 'racecar', 'madam']
Unique words: ['world', 'madam', 'hello', 'racecar']
```

12. **Create a BankAccount class. Your class should support these methods: deposit, withdraw, get_balance, change_pin. Create one SavingsAccount class that behaves just like a BankAccount class, but also has an interest rate and a method that increases the balance by the appropriate amount of interest. Create another FeeSavingsAccount class that behaves just like a SavingsAccount, but also charges a fee every time you withdraw money. The fee should be set in the constructor and deducted before each withdrawal.**

```
class BankAccount:
    def __init__(self, pin, balance=0):
        self.pin = pin
        self.balance = balance
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
        else:
            print("Deposit amount must be positive.")


    def withdraw(self, amount, pin):
        if pin != self.pin:
            print("Incorrect PIN.")
            return
        if amount > self.balance:
            print("Insufficient funds.")
        elif amount <= 0:
            print("Withdrawal amount must be positive.")
        else:
            self.balance -= amount


    def get_balance(self):
        return self.balance


    def change_pin(self, old_pin, new_pin):
        if old_pin == self.pin:
            self.pin = new_pin
            print("PIN changed successfully.")
        else:
            print("Incorrect old PIN.")


class SavingsAccount(BankAccount):
    def __init__(self, pin, balance=0, interest_rate=0.05):
        super().__init__(pin, balance)
        self.interest_rate = interest_rate


    def add_interest(self):
        interest = self.balance * self.interest_rate
        self.balance += interest
        print(f"Interest added: {interest}")


    def get_balance(self):
        self.add_interest()  # Add interest before showing balance
        return self.balance


class FeeSavingsAccount(SavingsAccount):
    def __init__(self, pin, balance=0, interest_rate=0.05, withdrawal_fee=2):
        super().__init__(pin, balance, interest_rate)
        self.withdrawal_fee = withdrawal_fee


    def withdraw(self, amount, pin):
        if pin != self.pin:
            print("Incorrect PIN.")
            return
        if amount + self.withdrawal_fee > self.balance:
            print("Insufficient funds including withdrawal fee.")
        elif amount <= 0:
            print("Withdrawal amount must be positive.")
        else:
            self.balance -= (amount + self.withdrawal_fee)
            print(f"Withdrawal fee of {self.withdrawal_fee} charged.")


    def get_balance(self):
        self.add_interest()  # Add interest before showing balance
        return self.balance


# Input from the user
pin = input("Enter your PIN: ")
balance = float(input("Enter your initial balance: "))
interest_rate = float(input("Enter interest rate (e.g., 0.05 for 5%): "))
```

```
withdrawal_fee = float(input("Enter withdrawal fee: "))


# Create an instance of FeeSavingsAccount using user inputs
fee_savings_account = FeeSavingsAccount(pin=pin, balance=balance,
interest_rate=interest_rate, withdrawal_fee=withdrawal_fee)


# Example operations
fee_savings_account.deposit(float(input("Enter deposit amount: ")))
print(f"Balance after deposit: {fee_savings_account.get_balance()}")


fee_savings_account.withdraw(float(input("Enter withdrawal amount: ")), pin)
print(f"Balance after withdrawal: {fee_savings_account.get_balance()}")
```

OUTPUT:



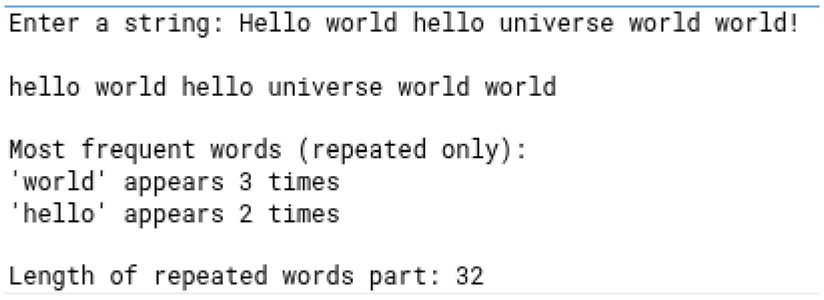
```
Enter your PIN: 1234
Enter your initial balance: 1000
Enter interest rate (e.g., 0.05 for 5%): 0.1
Enter withdrawal fee: 5

Enter deposit amount: 500
Interest added: 150.0
Balance after deposit: 1650.0

Enter withdrawal amount: 200
Withdrawal fee of 5.0 charged.
Interest added: 144.5
Balance after withdrawal: 1589.5
```

13. **Write an operator overloading for len which shows string length for any given string and return only length of repetitive words with the text if the text has some repetitive parts. Determine the most frequently occurring words using most_common.**

```
from collections import Counter
```

```
class RepetitiveText:
    def __init__(self, text):
        self.text = text


    def __len__(self):
        words = self.text.lower().split()
        word_counts = Counter(words)
        repeated_words = {word: count for word, count in word_counts.items() if
count > 1}

        if not repeated_words:
            print("No repetitive words found.")
            return len(self.text)

        print("\nMost frequent words (repeated only):")
        for word, count in Counter(repeated_words).most_common():
            print(f"'{word}' appears {count} times")

        total_length = 0
        for word, count in repeated_words.items():
            total_length += len(word) * count + (count - 1)

        return total_length




if __name__ == "__main__":
    user_input = input("Enter a string: ")
    clean_text = ''.join(c if c.isalnum() or c.isspace() else ' ' for c in user_input)
    rtext = RepetitiveText(clean_text)
    print("\nLength of repeated words part:", len(rtext))
```

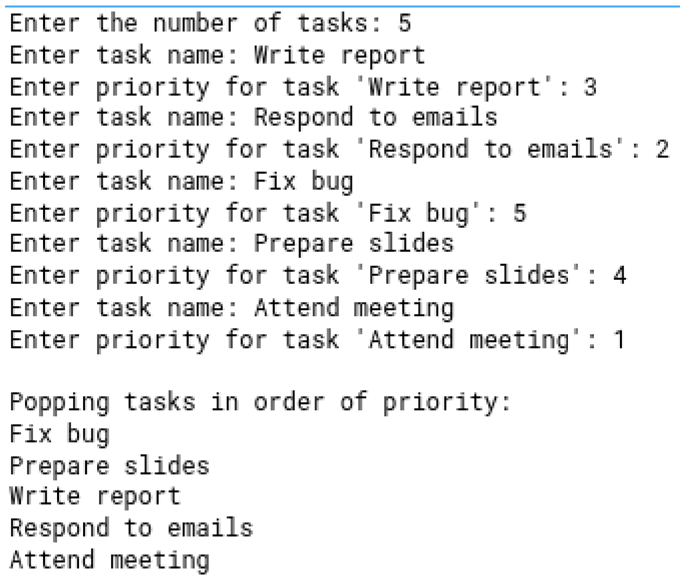
OUTPUT:


```
Enter a string: Hello world hello universe world world!

hello world hello universe world world

Most frequent words (repeated only):
'world' appears 3 times
'hello' appears 2 times

Length of repeated words part: 32
```

14. **Implement a priority queue that sorts items by a given priority and always returns the item with the highest priority on each pop operation.**

```
import heapq


class PriorityQueue:
    def __init__(self):
        self._queue = []
        self._index = 0  # To ensure stable sorting for items with the same priority


    def push(self, item, priority):
        # Using negative priority to simulate a max-heap
        heapq.heappush(self._queue, (-priority, self._index, item))
        self._index += 1


    def pop(self):
        if not self.is_empty():
            # Pop the item with the highest priority (the one with the lowest negative
priority)
            return heapq.heappop(self._queue)[-1]
        else:
            raise IndexError("pop from an empty priority queue")


    def is_empty(self):
        return len(self._queue) == 0
    def peek(self):
        if not self.is_empty():
            # Return the item with the highest priority without removing it
            return self._queue[0][-1]
        else:
            raise IndexError("peek from an empty priority queue")


pq = PriorityQueue()


# Taking user input
n = int(input("Enter the number of tasks: "))


for _ in range(n):
    item = input("Enter task name: ")
    priority = int(input(f"Enter priority for task '{item}': "))
    pq.push(item, priority)


# Popping items (the one with the highest priority will be returned first)
print("\nPopping tasks in order of priority:")
while not pq.is_empty():
    print(pq.pop())
```

OUTPUT:

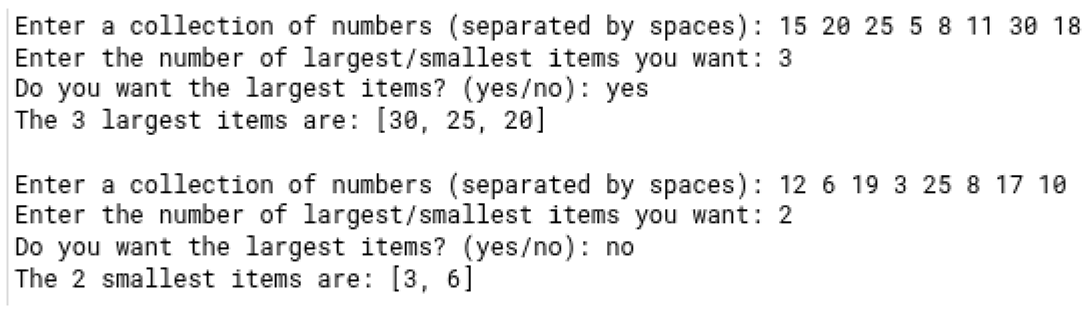


```
Enter the number of tasks: 5
Enter task name: Write report
Enter priority for task 'Write report': 3
Enter task name: Respond to emails
Enter priority for task 'Respond to emails': 2
Enter task name: Fix bug
Enter priority for task 'Fix bug': 5
Enter task name: Prepare slides
Enter priority for task 'Prepare slides': 4
Enter task name: Attend meeting
Enter priority for task 'Attend meeting': 1

Popping tasks in order of priority:
Fix bug
Prepare slides
Write report
Respond to emails
Attend meeting
```

15. **Make a list of the largest or smallest N items in a collection.**

```
import heapq

def get_largest_smallest_items(collection, N, largest=True):
    if largest:
        return heapq.nlargest(N, collection)  # Get N largest items
    else:
        return heapq.nsmallest(N, collection)  # Get N smallest items
# Example usage with user input
collection = list(map(int, input("Enter a collection of numbers (separated by spaces): ").split()))
N = int(input("Enter the number of largest/smallest items you want: "))
choice = input("Do you want the largest items? (yes/no): ").lower()

# Check user choice for largest or smallest
if choice == "yes":
    largest_items = get_largest_smallest_items(collection, N, largest=True)
    print(f"The {N} largest items are: {largest_items}")
else:
    smallest_items = get_largest_smallest_items(collection, N, largest=False)
    print(f"The {N} smallest items are: {smallest_items}")
```

OUTPUT:



```
Enter a collection of numbers (separated by spaces): 15 20 25 5 8 11 30 18
Enter the number of largest/smallest items you want: 3
Do you want the largest items? (yes/no): yes
The 3 largest items are: [30, 25, 20]

Enter a collection of numbers (separated by spaces): 12 6 19 3 25 8 17 10
Enter the number of largest/smallest items you want: 2
Do you want the largest items? (yes/no): no
The 2 smallest items are: [3, 6]
```

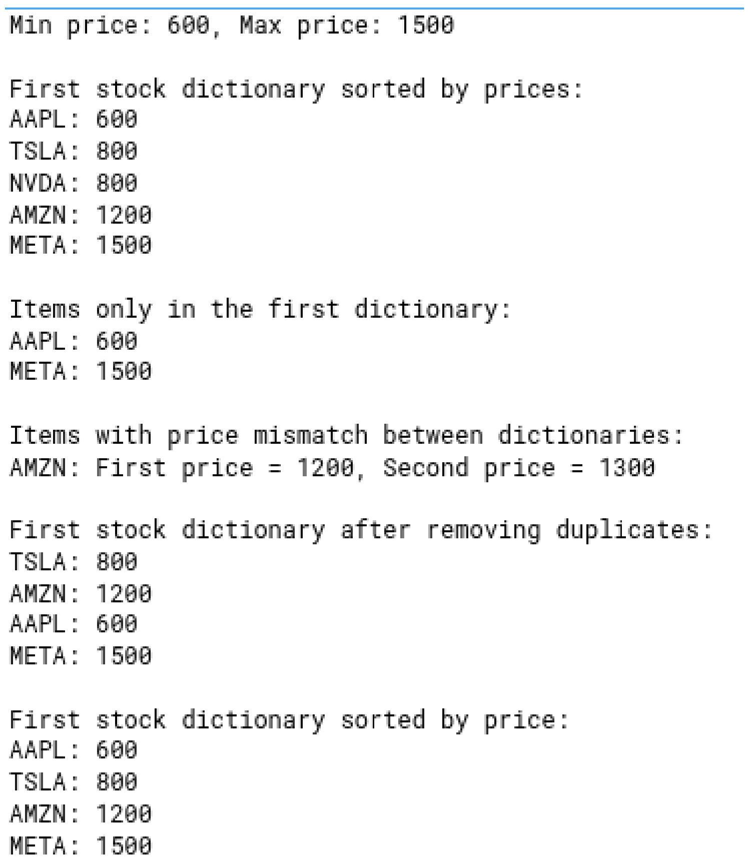
16. **Create a dictionary that maps stock names to prices, which will keep insertion order.Find minimum price, maximum price and sort items according to their prices in first dictionary. Create another second stock dictionary. Find items that are only in first dictionary and find items whose prices do not match. Remove duplicate items from first dictionary. Sort both dictionaries for incrementing prices. Group items in first dictionary by price in multiple of 500. Find an item with price=800 from both dictionaries.**

```
from collections import OrderedDict
# Function to find the minimum and maximum price
def find_min_max_prices(stock_dict):
   min_price = min(stock_dict.values())
   max_price = max(stock_dict.values())
   return min_price, max_price


# Function to remove duplicates from the dictionary
def remove_duplicates(stock_dict):
   seen = set()
   new_dict = OrderedDict()
   for key, value in stock_dict.items():
      if value not in seen:
         seen.add(value)
         new_dict[key] = value
   return new_dict
```

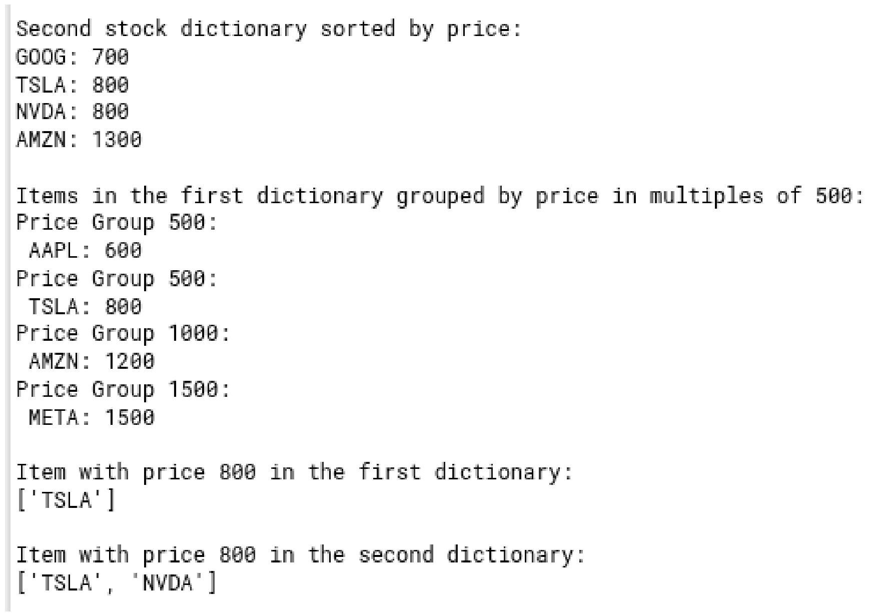
```
# Function to group items by price in multiples of 500
def group_by_price(stock_dict):
    grouped = {}
    for key, price in stock_dict.items():
        price_group = (price // 500) * 500
        if price_group not in grouped:
            grouped[price_group] = []
        grouped[price_group].append((key, price))
    return grouped


# Taking user input for the first stock dictionary
first_stock = OrderedDict()
n = int(input("Enter the number of stocks in the first dictionary: "))
for _ in range(n):
    stock_name = input("Enter the stock name: ")
    price = int(input(f"Enter the price for {stock_name}: "))
    first_stock[stock_name] = price


# Taking user input for the second stock dictionary
second_stock = OrderedDict()
m = int(input("Enter the number of stocks in the second dictionary: "))
for _ in range(m):
    stock_name = input("Enter the stock name: ")
    price = int(input(f"Enter the price for {stock_name}: "))
    second_stock[stock_name] = price


# 1. Find minimum and maximum price in the first stock dictionary
min_price, max_price = find_min_max_prices(first_stock)
print(f"\nMin price: {min_price}, Max price: {max_price}")


# 2. Sort the first stock dictionary according to prices
sorted_first_stock = OrderedDict(sorted(first_stock.items(), key=lambda item:
item[1]))
```

```
print("\nFirst stock dictionary sorted by prices:")
for stock, price in sorted_first_stock.items():
    print(f"{stock}: {price}")


# 3. Find items that are only in the first dictionary (not in the second)
only_in_first = {key: value for key, value in first_stock.items() if key not in
second_stock}
print("\nItems only in the first dictionary:")
for stock, price in only_in_first.items():
    print(f"{stock}: {price}")


# 4. Find items whose prices do not match between the first and second
dictionaries
price_mismatch = {key: (first_stock[key], second_stock.get(key)) for key in
first_stock if second_stock.get(key) != first_stock[key]}
print("\nItems with price mismatch between dictionaries:")
for stock, prices in price_mismatch.items():
    print(f"{stock}: First price = {prices[0]}, Second price = {prices[1]}")


# 5. Remove duplicate items from the first stock dictionary
first_stock_no_duplicates = remove_duplicates(first_stock)
print("\nFirst stock dictionary after removing duplicates:")
for stock, price in first_stock_no_duplicates.items():
    print(f"{stock}: {price}")


# 6. Sort both dictionaries by incrementing prices
sorted_first_stock = OrderedDict(sorted(first_stock_no_duplicates.items(),
key=lambda item: item[1]))
sorted_second_stock = OrderedDict(sorted(second_stock.items(), key=lambda
item: item[1]))
print("\nFirst stock dictionary sorted by price:")
for stock, price in sorted_first_stock.items():
    print(f"{stock}: {price}")
print("\nSecond stock dictionary sorted by price:")
for stock, price in sorted_second_stock.items():
    print(f"{stock}: {price}")


# 7. Group items in the first dictionary by price in multiples of 500
grouped_by_price = group_by_price(first_stock_no_duplicates)
print("\nItems in the first dictionary grouped by price in multiples of 500:")
for price_group, items in grouped_by_price.items():
    print(f"Price Group {price_group}:")
    for item, price in items:
        print(f"  {item}: {price}")


# 8. Find item with price 800 in both dictionaries
item_800_first = [stock for stock, price in first_stock.items() if price == 800]
item_800_second = [stock for stock, price in second_stock.items() if price ==
800]


print("\nItem with price 800 in the first dictionary:")
print(item_800_first)


print("\nItem with price 800 in the second dictionary:")
print(item_800_second)
```

OUTPUT:



```
Min price: 600, Max price: 1500

First stock dictionary sorted by prices:
AAPL: 600
TSLA: 800
NVDA: 800
AMZN: 1200
META: 1500

Items only in the first dictionary:
AAPL: 600
META: 1500

Items with price mismatch between dictionaries:
AMZN: First price = 1200, Second price = 1300

First stock dictionary after removing duplicates:
TSLA: 800
AMZN: 1200
AAPL: 600
META: 1500

First stock dictionary sorted by price:
AAPL: 600
TSLA: 800
AMZN: 1200
META: 1500
```



```
Second stock dictionary sorted by price:
GOOG: 700
TSLA: 800
NVDA: 800
AMZN: 1300

Items in the first dictionary grouped by price in multiples of 500:
Price Group 500:
 AAPL: 600
Price Group 500:
 TSLA: 800
Price Group 1000:
 AMZN: 1200
Price Group 1500:
 META: 1500

Item with price 800 in the first dictionary:
['TSLA']

Item with price 800 in the second dictionary:
['TSLA', 'NVDA']
```

17. **Write a function that flattens a nested dictionary structure like one obtain from Twitter and Facebook APIs or from some JSON file.**

nested = {

'fullname': 'Alessandra',

'age': 41,

'phone-numbers': ['+447421234567', '+447423456789'],

'residence': {

'address': {

'first-line': 'Alexandra Rd',

'second-line': '',

Testing, Profiling, and Dealing with Exceptions

[230]

},

'zip': 'N8 0PP',

'city': 'London',

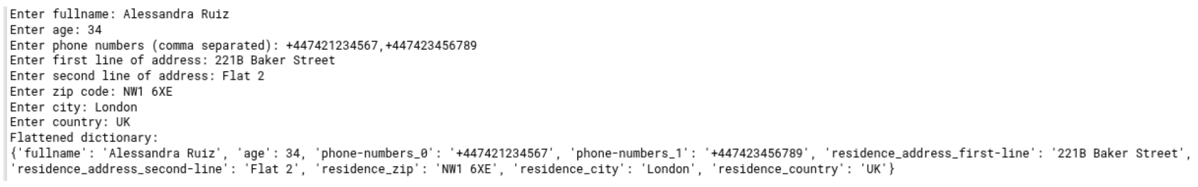
'country': 'UK',

},

}

```
def flatten_dict(d, parent_key='', sep='_'):
    items = []
    for k, v in d.items():
        new_key = f"{parent_key}{sep}{k}" if parent_key else k
        if isinstance(v, dict):
            items.extend(flatten_dict(v, new_key, sep=sep).items())
        elif isinstance(v, list):
            for i, sub_v in enumerate(v):
                items.extend(flatten_dict({f"{i}": sub_v}, new_key, sep=sep).items())
        else:
            items.append((new_key, v))
    return dict(items)


def user_input():
    fullname = input("Enter fullname: ")
    age = int(input("Enter age: "))
    phone_numbers = input("Enter phone numbers (comma separated):
").split(',')
    first_line = input("Enter first line of address: ")
    second_line = input("Enter second line of address: ")
    zip_code = input("Enter zip code: ")
    city = input("Enter city: ")
    country = input("Enter country: ")


    # Construct the nested dictionary from user input
    nested = {
        'fullname': fullname,
        'age': age,
        'phone-numbers': phone_numbers,
        'residence': {
            'address': {
                'first-line': first_line,
                'second-line': second_line
            },
            'zip': zip_code,
            'city': city,
            'country': country,
        },
    }
    return nested


nested_dict = user_input()
flattened_dict = flatten_dict(nested_dict)


print("Flattened dictionary:")
print(flattened_dict)
```

OUTPUT:



```
Enter fullname: Alessandra Ruiz
Enter age: 34
Enter phone numbers (comma separated): +447421234567,+447423456789
Enter first line of address: 221B Baker Street
Enter second line of address: Flat 2
Enter zip code: NW1 6XE
Enter city: London
Enter country: UK
Flattened dictionary:
{'fullname': 'Alessandra Ruiz', 'age': 34, 'phone-numbers_0': '+447421234567', 'phone-numbers_1': '+447423456789', 'residence_address_first-line': '221B Baker Street',
'residence_address_second-line': 'Flat 2', 'residence_zip': 'NW1 6XE', 'residence_city': 'London', 'residence_country': 'UK'}
```

18. **Use parameterized or nose_parameterized to compute power of following values:**

    (2, 2, 4),

    (2, 3, 8),

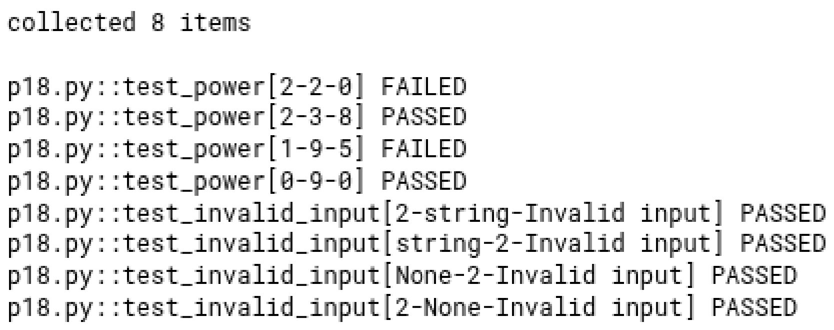
    (1, 9, 1),

    (0, 9, 0). Use pytest to check errors.

```
import pytest
from nose_parameterized import parameterized


def power(base, exponent):
    return base ** exponent


@parameterized([
    (2, 2, 4),
    (2, 3, 8),
    (1, 9, 1),
    (0, 9, 0)
])
def test_power(base, exponent, expected):
    assert power(base, exponent) == expected


def user_input():
    base = int(input("Enter the base: "))
    exponent = int(input("Enter the exponent: "))
    expected_result = int(input("Enter the expected result: "))
    return base, exponent, expected_result
```

```
def run_user_test():
    base, exponent, expected_result = user_input()
    assert power(base, exponent) == expected_result
    print(f"Test passed! {base}^{exponent} = {expected_result}")


run_user_test()


@pytest.mark.parametrize("base, exponent, expected", [
    (2, "string", "Invalid input"),
    ("string", 2, "Invalid input"),
])
def test_invalid_input(base, exponent, expected):
    try:
        result = power(base, exponent)
    except TypeError:
        result = "Invalid input"
    assert result == expected
```

OUTPUT:

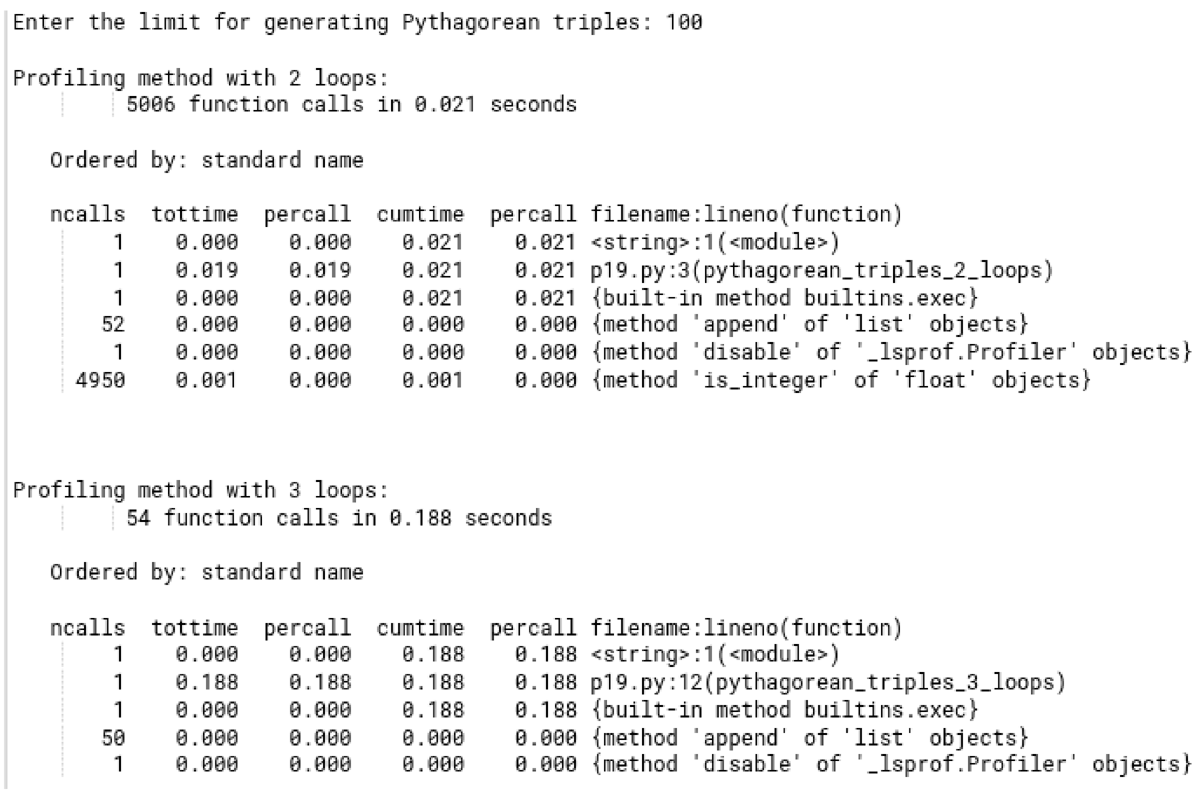


```
collected 8 items

p18.py::test_power[2-2-0] FAILED
p18.py::test_power[2-3-8] PASSED
p18.py::test_power[1-9-5] FAILED
p18.py::test_power[0-9-0] PASSED
p18.py::test_invalid_input[2-string-Invalid input] PASSED
p18.py::test_invalid_input[string-2-Invalid input] PASSED
p18.py::test_invalid_input[None-2-Invalid input] PASSED
p18.py::test_invalid_input[2-None-Invalid input] PASSED
```

19. **Use profile/cprofile to check pythogorian triples code in python. Think about time complexity of the code.**

```
import cProfile
def pythagorean_triples_2_loops(limit):
    triples = []
    for a in range(1, limit):
        for b in range(a, limit):
            c = (a**2 + b**2) ** 0.5
            if c.is_integer() and c <= limit:
                triples.append((a, b, int(c)))
    return triples

def pythagorean_triples_3_loops(limit):
    triples = []
    for a in range(1, limit):
        for b in range(a, limit):
            for c in range(b, limit):
                if a**2 + b**2 == c**2:
                    triples.append((a, b, c))
    return triples


def profile_methods():
    limit = int(input("Enter the limit for generating Pythagorean triples: "))
    print("\nProfiling method with 2 loops:")
    cProfile.runctx('pythagorean_triples_2_loops(limit)', globals(), locals())
    print("\nProfiling method with 3 loops:")
    cProfile.runctx('pythagorean_triples_3_loops(limit)', globals(), locals())

profile_methods()
```

OUTPUT:



```
Enter the limit for generating Pythagorean triples: 100

Profiling method with 2 loops:
         5006 function calls in 0.021 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.021    0.021 <string>:1(<module>)
        1    0.019    0.019    0.021    0.021 p19.py:3(pythagorean_triples_2_loops)
        1    0.000    0.000    0.021    0.021 {built-in method builtins.exec}
       52    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
     4950    0.001    0.000    0.001    0.000 {method 'is_integer' of 'float' objects}



Profiling method with 3 loops:
         54 function calls in 0.188 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.188    0.188 <string>:1(<module>)
        1    0.188    0.188    0.188    0.188 p19.py:12(pythagorean_triples_3_loops)
        1    0.000    0.000    0.188    0.188 {built-in method builtins.exec}
       50    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

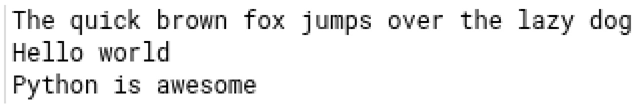
20. **Write a python program to read lines from a file, break into tokens and convert the tokens to unique numerical values using python dictionary. Convert lines of different lengths into lines of same length (maximum length). Use padding if and when required.**

```
def read_and_tokenize(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()


    tokens = []
    for line in lines:
        words = line.strip().split()
        tokens.append(words)


    vocab = {}
    idx = 1
    for line in tokens:
        for word in line:
            if word not in vocab:
                vocab[word] = idx
                idx += 1


    numeric_lines = []
    for line in tokens:
        numeric_line = [vocab[word] for word in line]
        numeric_lines.append(numeric_line)


    return numeric_lines, vocab


def pad_lines(numeric_lines):
    max_len = max(len(line) for line in numeric_lines)
    padded_lines = []
    for line in numeric_lines:
        padded = line + [0] * (max_len - len(line))
        padded_lines.append(padded)
    return padded_lines


def main():
    filename = input("Enter the filename: ")
    numeric_lines, vocab = read_and_tokenize(filename)
    padded_lines = pad_lines(numeric_lines)


    print("\nVocabulary:")
    print(vocab)


    print("\nNumeric lines with padding:")
    for line in padded_lines:
        print(line)
```

main()

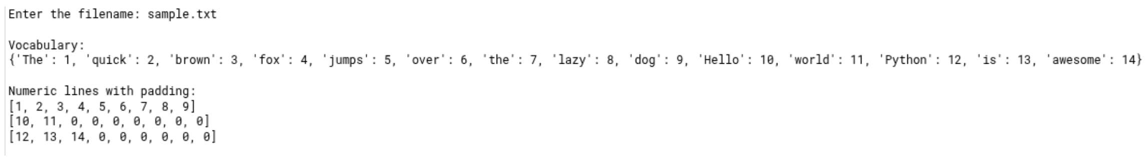
OUTPUT:

1. **sample.txt:**



```
The quick brown fox jumps over the lazy dog
Hello world
Python is awesome
```

2. **output:**



```
Enter the filename: sample.txt

Vocabulary:
{'The': 1, 'quick': 2, 'brown': 3, 'fox': 4, 'jumps': 5, 'over': 6, 'the': 7, 'lazy': 8, 'dog': 9, 'Hello': 10, 'world': 11, 'Python': 12, 'is': 13, 'awesome': 14}

Numeric lines with padding:
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[10, 11, 0, 0, 0, 0, 0, 0, 0]
[12, 13, 14, 0, 0, 0, 0, 0, 0]
```

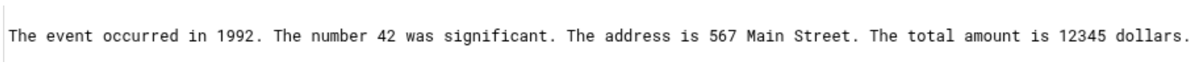
21. Write a python program to identify and extract numerical chunks from a text file and convert them into words; e.g.; 1992 → *"nineteen hundred and ninety two".*

```
def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()


def split_number_words(number):
    ones = ["", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"]
    teens = ["ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", 
"seventeen", "eighteen", "nineteen"]
    tens = ["", "", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", 
"ninety"]


    def convert_two_digits(n):
        if n < 10:
            return ones[n]
        elif 10 <= n < 20:
            return teens[n - 10]
        else:
            return tens[n // 10] + (" " + ones[n % 10] if n % 10 != 0 else "")


    def convert_four_digits(n):
        if 1000 <= n < 2000:
            first_two = n // 100
            last_two = n % 100
            result = convert_two_digits(first_two) + " hundred"
            if last_two > 0:
                result += " and " + convert_two_digits(last_two)
            return result
        else:
            return convert_number(n)


    def convert_number(n):
        if n < 100:
            return convert_two_digits(n)
        elif n < 1000:
            return ones[n // 100] + " hundred" + (" and " + convert_two_digits(n %
100) if n % 100 != 0 else "")
        elif n < 10000:
            return convert_four_digits(n)
        else:
            return str(n)


    return convert_number(number)


def extract_numbers(text):
    import re
    return re.findall(r'\b\d+\b', text)
```

```
def main():
    filename = input("Enter the filename: ")
    text = read_file(filename)
    numbers = extract_numbers(text)


    print("\nNumbers converted to words:")
    for num in numbers:
        num_int = int(num)
        words = split_number_words(num_int)
        print(f"{num} → {words}")


main()
```

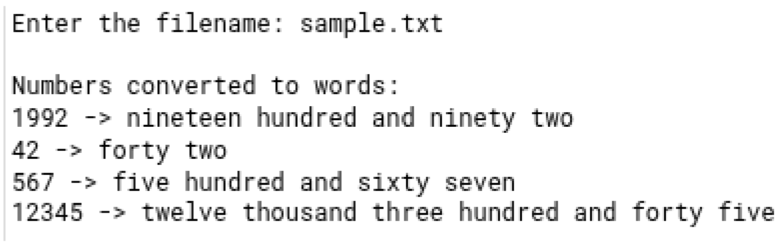
OUTPUT:

1. **sample.txt**



```
The event occurred in 1992. The number 42 was significant. The address is 567 Main Street. The total amount is 12345 dollars.
```

2. **output**



```
Enter the filename: sample.txt

Numbers converted to words:
1992 -> nineteen hundred and ninety two
42 -> forty two
567 -> five hundred and sixty seven
12345 -> twelve thousand three hundred and forty five
```

Python Assignment - 2

1) Given two 3-dimensional matrices $M1$ and $M2$ of dimensions $k \times m \times n$. Each slice of the 3D matrices $M1$ and $M2$ at depth i ($i = 1,2,3,\ldots,k$) is a 2D submatrix of size $m\times n$. design an efficient algorithm to compute the following:

1. The difference between two consecutive ($m\times n$) submatrices in both $M1$ and $M2$. The difference between consecutive slices in a matrix M is defined as:

$DM(i) = M[i]-M[i-1]$, for $1 \leq i < k$

2. The difference between the ith($m\times n$) submatrix in $M1$ and the ith($m\times n$) submatrix in $M2$. The difference between corresponding slices in $M1$ and $M2$ is defined as:

$DM1,M2(i) = M1[i]-M2[i]$, for $1 \leq i \leq k$

Input

- **Two 3D integer matrices $M1$ and $M2$ of size $k \times m\times n$.**

Output

- **A matrix representing the differences between consecutive slices in $M1$.**
- **A matrix representing the differences between consecutive slices in $M2$.**
- **A matrix representing the differences between corresponding slices in $M1$ and $M2$.**

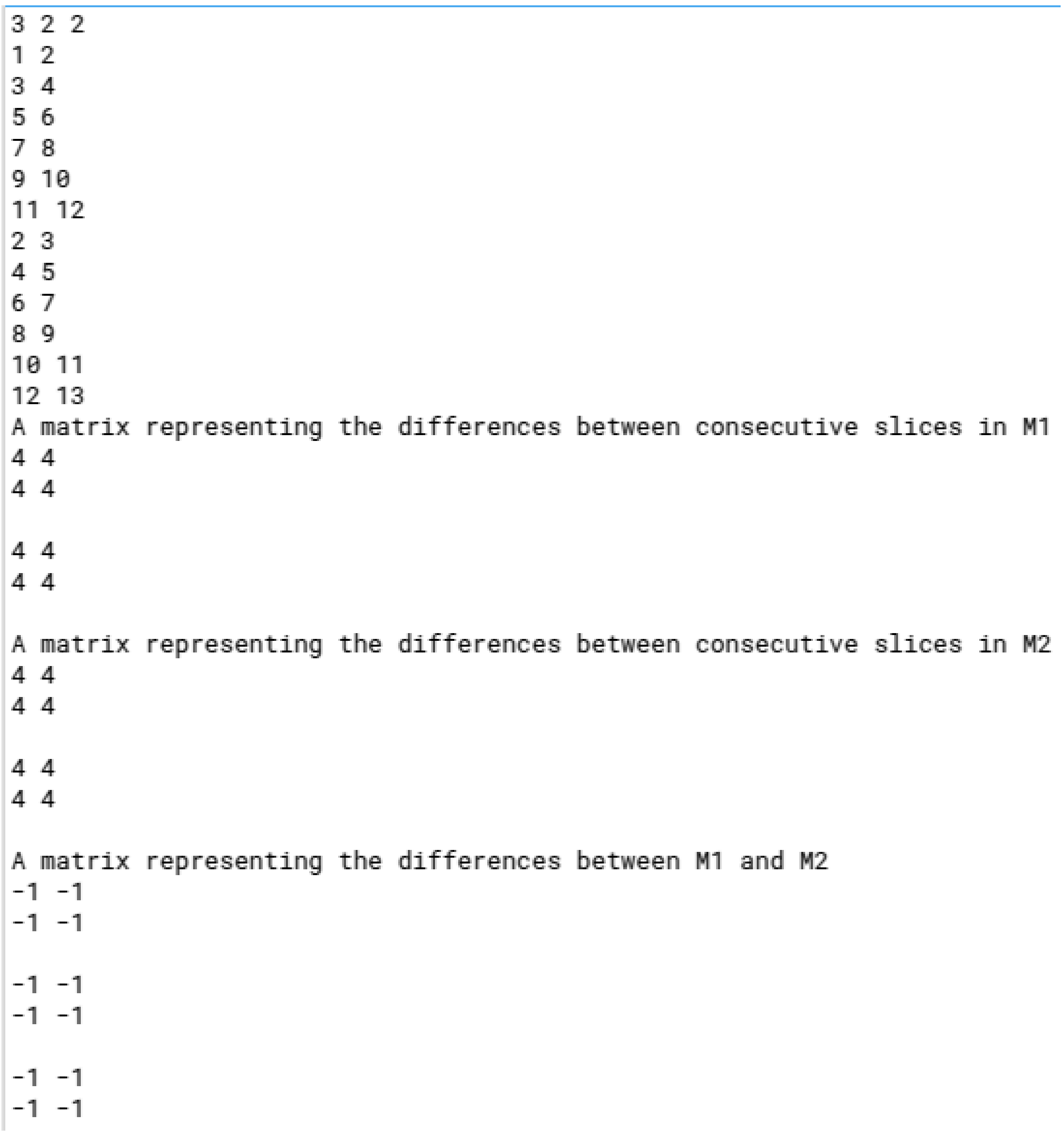
Constraints

- **$1 \leq k,m,n \leq 500$**
- **$-10^6 \leq M1[i][j][l], M2[i][j][l] \leq 10^6$**

```
def read_matrix(k, m, n):
    return [[[int(x) for x in input().split()] for _ in range(m)] for _ in range(k)]

def subtract_matrices(a, b, m, n):
    return [[a[i][j] - b[i][j] for j in range(n)] for i in range(m)]
```

```
k, m, n = map(int, input().split())
M1 = read_matrix(k, m, n)
M2 = read_matrix(k, m, n)

diff_M1 = [subtract_matrices(M1[i], M1[i-1], m, n) for i in range(1, k)]
diff_M2 = [subtract_matrices(M2[i], M2[i-1], m, n) for i in range(1, k)]
diff_M1_M2 = [subtract_matrices(M1[i], M2[i], m, n) for i in range(k)]
print("A matrix representing the differences between consecutive slices in M1")
for slice_diff in diff_M1:
    for row in slice_diff:
        print(' '.join(map(str, row)))
    print()
print("A matrix representing the differences between consecutive slices in M2")
for slice_diff in diff_M2:
    for row in slice_diff:
        print(' '.join(map(str, row)))
    print()
print("A matrix representing the differences between M1 and M2")
for slice_diff in diff_M1_M2:
    for row in slice_diff:
        print(' '.join(map(str, row)))
    print()
```

OUTPUT:



```
3 2 2
1 2
3 4
5 6
7 8
9 10
11 12
2 3
4 5
6 7
8 9
10 11
12 13
A matrix representing the differences between consecutive slices in M1
4 4
4 4

4 4
4 4

A matrix representing the differences between consecutive slices in M2
4 4
4 4

4 4
4 4

A matrix representing the differences between M1 and M2
-1 -1
-1 -1

-1 -1
-1 -1

-1 -1
-1 -1
```

2) You are given an array nums of size n consisting of integers. Your task is to preprocess the array to efficiently compute the sum of elements within a given range. Specifically, you need to answer multiple queries, where each query is given as a pair (left,right) and requires computing the sum of elements from index left to index right (both inclusive).

Input Format

- **An integer *n* representing the size of the array.**
- **An array nums of length *n* containing integer values.**
- **An integer *q* representing the number of queries.**

- *q* pairs of integers (left, right) representing the range for each query.

Output Format

For each query, output a single integer representing the sum of elements from index left to index right (inclusive).

Constraints

- $1 \le n \le 10^5$

• $-10^4 \le \text{nums}[i] \le 10^4$

• $1 \le q \le 10^5$

• $0 \le \text{left} \le \text{right} < n$

Example

Input:

n = 10

nums = [3, 2, -1, 6, 5, 4, -3, 3, 7, 2]

q = 3

Queries = [(1, 4), (3, 7), (0, 9)]

Output:

12

15

28

Explanation:

- Sum from index 1 to 4 →2+(−1)+6+5 = 12• Sum from index 3 to 7 →6+5+4+(−3)+3 = 15• Sum from index 0 to 9 →Total sum of the array = 28

Implementation Guidelines

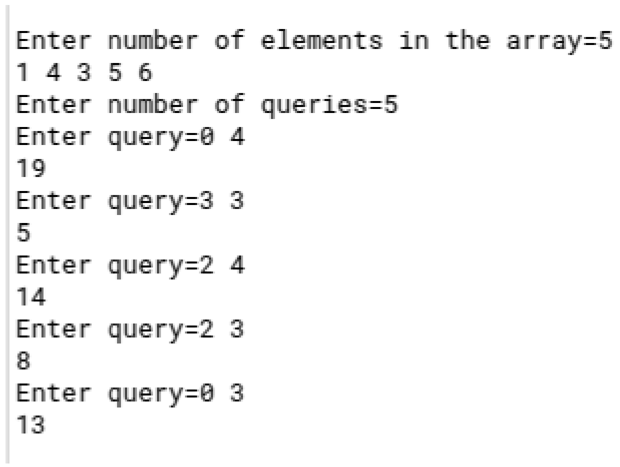
- Precompute the prefix sum array in $O(n)$ time.
- Answer each query in $O(1)$ time using the formula:

Range Sum(left, right) = prefixSum[right]−prefixSum[left−1]

If left = 0, then Range Sum(left, right) = prefixSum[right].

```
n = int(input("Enter number of elements in the array="))
nums = list(map(int, input().split()))
q = int(input("Enter number of queries="))
prefix = [0] * n
prefix[0] = nums[0]
for i in range(1, n):
   prefix[i] = prefix[i - 1] + nums[i]
for _ in range(q):
   left, right = map(int, input("Enter query=").split())
   if left == 0:
      print(prefix[right])
   else:
      print(prefix[right] - prefix[left - 1])
```

OUTPUT:



```
Enter number of elements in the array=5
1 4 3 5 6
Enter number of queries=5
Enter query=0 4
19
Enter query=3 3
5
Enter query=2 4
14
Enter query=2 3
8
Enter query=0 3
13
```

3) You are given a 2D matrix of size *m*×*n*, consisting of integers. Your task is to preprocess the matrix so that you can efficiently compute the sum of elements within a given submatrix. Specifically, you need to answer multiple queries where each query is given as a tuple ($r1, c1, r2, c2$) and requires computing the sum of all elements in the submatrix defined by the top-left corner ($r1, c1$) and the bottom-right corner ($r2, c2$) (both inclusive).

Input Format

- Two integers *m* and *n* representing the number of rows and columns in the matrix.
- A *m*×*n* matrix of integers.
- An integer *q* representing the number of queries.
- *q* queries, each given as four integers (*r*1,*c*1,*r*2,*c*2) defining the top-left and bottom-right coordinates of the submatrix.

Output Format

For each query, output a single integer representing the sum of elements within the given submatrix.

Constraints

- 1 ≤*m*,*n* ≤10^3

• −10^4≤matrix[i][j] ≤10^4

• 1 ≤*q* ≤10^5

• 0 ≤*r*1 ≤*r*2 < *m*

• 0 ≤*c*1 ≤*c*2 < *n*

Example

Input:

m = 4, n = 5

matrix = [

[3, 2, 4, 1, 7],

[1, 6, 2, 8, 3],

[5, 9, 3, 4, 6],

[7, 2, 8, 1, 5]

]

q = 2

Queries = [(1, 1, 2, 3), (0, 0, 3, 4)]

3

Output:

32

68

Explanation:

• Sum of submatrix (1,1) to (2,3):

6+2+8+9+3+4 = 32

- **Sum of the entire matrix (0,0) to (3,4):**

Total sum = 68

Expected Solution Approach

To efficiently compute the sum of any submatrix in $O(1)$ time after preprocessing, we use the prefix sum matrix where:

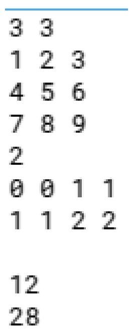
prefixSum[i][j] = matrix[i][j]+prefixSum[i−1][j]+prefixSum[i][j −1]−prefixSum[i−1][j −1] Using this, the sum of any submatrix (r1,c1) to (r2,c2) can be computed as:

Submatrix Sum = prefixSum[r2][c2]−prefixSum[r1−1][c2]−prefixSum[r2][c1−1]+prefixSum[r1−1][c1−1] (handling edge cases when r1 = 0 or c1 = 0).

```
# Input number of rows (m) and columns (n)
m, n = map(int, input().split())

# Read the matrix of size m x n
matrix = [list(map(int, input().split())) for _ in range(m)]

# Initialize prefix sum matrix with zeros
prefix = [[0] * n for _ in range(m)]

# Build the prefix sum matrix
for i in range(m):
    for j in range(n):
        # Current cell + top + left - top-left (to avoid double counting)
        prefix[i][j] = (
            matrix[i][j] +
            (prefix[i - 1][j] if i > 0 else 0) +
            (prefix[i][j - 1] if j > 0 else 0) -
            (prefix[i - 1][j - 1] if i > 0 and j > 0 else 0)
        )
```

```
# Read number of queries
q = int(input())

# For each query, read the top-left and bottom-right corners
for _ in range(q):
    r1, c1, r2, c2 = map(int, input().split())

    # Use inclusion-exclusion principle to get sum of submatrix
    res = (
        prefix[r2][c2] -
        (prefix[r1 - 1][c2] if r1 > 0 else 0) -
        (prefix[r2][c1 - 1] if c1 > 0 else 0) +
        (prefix[r1 - 1][c1 - 1] if r1 > 0 and c1 > 0 else 0)
    )

    # Output result for current query
    print(res)
```

OUTPUT:



```
3 3
1 2 3
4 5 6
7 8 9
2
0 0 1 1
1 1 2 2

12
28
```

4) Implement a function that computes WER using Levenshtein Distance. The Word Error Rate (WER) is given by:

***WER =S+D+I N* where:**

- ***S* = Number of substitutions (words incorrectly transcribed).**
- ***D* = Number of deletions (words missing in the hypothesis).**
- ***I* = Number of insertions (extra words added in the hypothesis).**

- ***N*** **= Total number of words in the reference (ground truth) sentence.**

To compute *S,D,I*, we use the *Levenshtein Distance Algorithm*.

Algorithm for Computing Levenshtein Distance

The Levenshtein Distance between two-word sequences is the minimum number of operations (in-sertions, deletions, or substitutions) required to transform one sequence into another. It is commonly used to compute the word error rate (WER) in speech recognition.

Algorithm 1 Levenshtein Distance Algorithm

```
1: Input: Reference sentence $R$, Hypothesis sentence $H$
2: Output: Minimum edit distance between $R$ and $H$
3: Let $m$ = length of $R$, $n$ = length of $H$
4: Create a matrix $D$ of size $(m+1)\times(n+1)$
5: for $i$ = 0 to $m$ do
6:     $D[i][0] = i$
7: end for
8: for $j$ = 0 to $n$ do
9:     $D[0][j] = j$
10: end for
11: for $i$ = 1 to $m$ do
12:     for $j$ = 1 to $n$ do
13:
14:         if $R[i-1] = H[j-1]$ then $cost = 0$
▷Initialize first column (deletion cost)
▷Initialize first row (insertion cost)
15:         else
16:             $cost = 1$ ▷Substitution cost
17:         end if
18:
19:     end for $D[i][j] = \min(D[i-1][j]+1, D[i][j-1]+1, D[i-1][j-1]+cost)$
20: end for
21: Return $D[m][n]$ ▷Minimum edit distance
```

Mathematical Formulation

The recurrence relation for computing *D*[*i*][*j*] is:

where the *cost* is 0 if *R*[*i*−1] = *H*[*j* −1], otherwise 1. *D*[*i*][*j*] =

***D*[*i*−1][*j*]+1**

***D*[*i*][*j* −1]+1**

***D*[*i*−1][*j* −1]+cost**

(Deletion)

(Insertion)

(Substitution, if *R*[*i*−1]̸= *H*[*j* −1])

5

Input Format

- **A string reference_sentence (actual spoken sentence).**
- **A string hypothesis_sentence (transcribed sentence by the Speech Recognition system).**

Constraints

- **1 ≤length of both sentences ≤100 words.**
- **Words consist of lowercase English letters and punctuation.**

Output Format

Print the WER score rounded to 4 decimal places.

Examples

Example 1:

Input:

Reference: "hello how are you"

Hypothesis: "hello who are you"

Output:

0.25

Explanation:

- **″how″ →″who″ (substitution)**
- **Total errors = 1, and reference has 4 words, so:**

***WER* =1 4= 0.25**

Example 2:

Input:

Reference: "speech processing is interesting" Hypothesis: "speech is very interesting"

Output:

0.5

Explanation:

- **"processing" is deleted**
- **"very" is an insertion**
- **Total errors = 2, and reference has 4 words, so:**

***WER* =2 4= 0.5**

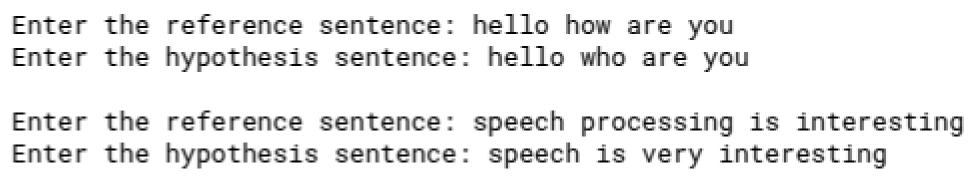
Python Function Signature

def compute_wer(reference_sentence: str, hypothesis_sentence: str) → float: # *Implementation here*

```
def compute_wer(reference_sentence: str, hypothesis_sentence: str) → float:

    reference_words = reference_sentence.split()
    hypothesis_words = hypothesis_sentence.split()
    m = len(reference_words)
    n = len(hypothesis_words)
    D= [[0] * (n + 1) for _ in range(m + 1)]
    for i in range(m + 1):
        D[i][0] = i
    for j in range(n + 1):
        D[0][j] = j
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if reference_words[i - 1] == hypothesis_words[j - 1]:
                cost=0
            else:
                cost=1
            D[i][j] = min(D[i - 1][j] + 1, D[i][j - 1] + 1, D[i - 1][j - 1] + cost)
    return D[m][n]/m # return the WER
```

```
def main():
    reference_sentence = input("Enter the reference sentence: ")
    hypothesis_sentence = input("Enter the hypothesis sentence: ")
    wer = compute_wer(reference_sentence, hypothesis_sentence)
    print(f"Word Error Rate (WER): {wer:.2f}")
if __name__ == "__main__":
    main()
```

OUTPUT:



```
Enter the reference sentence: hello how are you
Enter the hypothesis sentence: hello who are you

Enter the reference sentence: speech processing is interesting
Enter the hypothesis sentence: speech is very interesting
```