1A. Program :

```
import java.util.*;
   public class SquareRootNewton {
      public static void main(String[] args) {
         Scanner sc=new Scanner(System.in);
         int num = sc.nextInt();
         double sqrt = num / 2.0;
         double t;
         double epsilon = 1e-10;

         do {
            t = sqrt;
            sqrt = (t + (num / t)) / 2;
         } while (Math.abs(t - sqrt) > epsilon);

         System.out.println("Square root of " + num + " is: " + sqrt);
      }
   }
```

1A.Output :

```
25
Square root of 25 is: 5.0

=== Code Execution Successful ===
```

1B.Program :

```
import java.util.*;
public class UglyNumber {
        public static boolean isUgly(int num) {
            if (num <= 0) {
               return false;
            }
             while (num % 2 == 0) {
               num /= 2;
            }
            while (num % 3 == 0) {
               num /= 3;
            }
            while (num % 5 == 0) {
               num /= 5;
            }
             return num == 1;
        }
        public static void main(String[] args) {
            Scanner sc=new Scanner(System.in);
            int number = sc.nextInt();
            if (isUgly(number)) {
               System.out.println(number + " is an ugly number.");
            } else {
               System.out.println(number + " is not an ugly number.");
            }
        }
    }
```

1B.Output :

```
30
30 is an ugly number.

=== Code Execution Successful ===
```

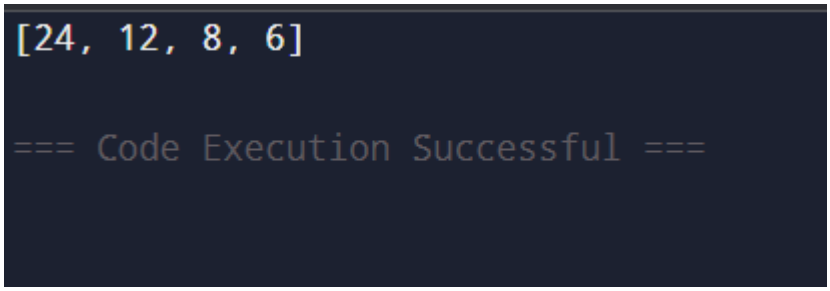
```
26
26 is not an ugly number.

=== Code Execution Successful ===
```

1C.Program :

```
import java.util.Arrays;
public class ProductExceptSelf {
    public static int[] productExceptSelf(int[] nums) {
        int n = nums.length;
        int[] left = new int[n];
        int[] right = new int[n];
        int[] output = new int[n];
        left[0] = 1;
        for (int i = 1; i < n; i++) {
            left[i] = left[i - 1] * nums[i - 1];
        }
        right[n - 1] = 1;
        for (int i = n - 2; i >= 0; i--) {
            right[i] = right[i + 1] * nums[i + 1];
        }
        for (int i = 0; i < n; i++) {
            output[i] = left[i] * right[i];
        }
        return output;
    }
    public static void main(String[] args) {
        int[] nums = {1, 2, 3, 4};
        int[] result = productExceptSelf(nums);
        System.out.println(Arrays.toString(result));
    }
}
```

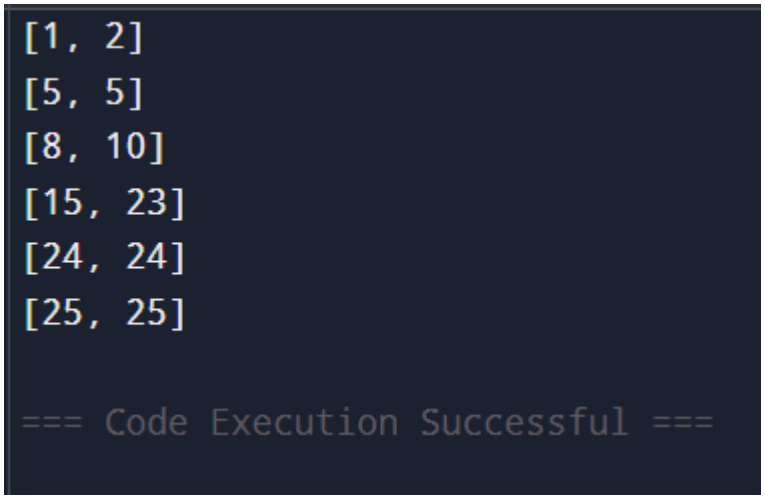
1C.Output :



2A.Program :

```
import java.util.*;
public class IntervalListIntersections {
    public static int[][] intervalIntersection(int[][] A, int[][] B) {
        List<int[]> result = new ArrayList<>();
        int i = 0, j = 0;
        while (i < A.length && j < B.length) {
            int startMax = Math.max(A[i][0], B[j][0]);
            int endMin = Math.min(A[i][1], B[j][1]);
            if (startMax <= endMin) {
                result.add(new int[]{startMax, endMin});
            }
            if (A[i][1] < B[j][1]) {
                i++;
            } else {
                j++;
            }
        }
        return result.toArray(new int[result.size()][]);
    }
    public static void main(String[] args) {
        int[][] A = {{0,2},{5,10},{13,23},{24,25}};
        int[][] B = {{1,5},{8,12},{15,24},{25,26}};
        int[][] intersections = intervalIntersection(A, B);
        for (int[] interval : intersections) {
            System.out.println(Arrays.toString(interval));
        }
    }
}
```

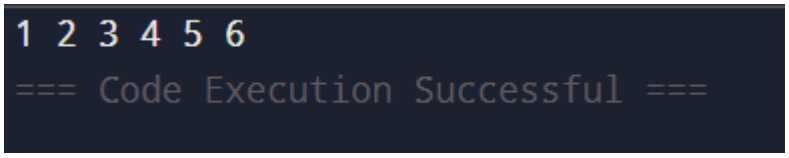
2A.Output :



2B.Program :

```
public class MergeSortedArray {
    public static void merge(int[] nums1, int m, int[] nums2, int n) {
        int i = m - 1;
        int j = n - 1;
        int k = m + n - 1;
        while (i >= 0 && j >= 0) {
            if (nums1[i] > nums2[j]) {
                nums1[k--] = nums1[i--];
            } else {
                nums1[k--] = nums2[j--];
            }
        }
        while (j >= 0) {
            nums1[k--] = nums2[j--];
        }
    }
    public static void main(String[] args) {
        int[] nums1 = {1, 3, 5, 0, 0, 0};
        int m = 3;
        int[] nums2 = {2, 4, 6};
        int n = 3;
        merge(nums1, m, nums2, n);
        for (int num : nums1) {
            System.out.print(num + " ");
        }
    }
}
```

2B.Output :



2C.Program :

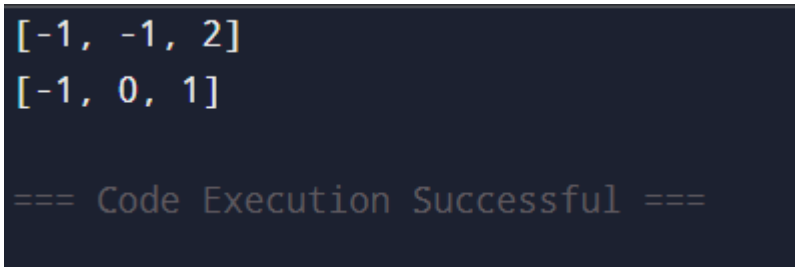
```
import java.util.*;
public class ThreeSum {
    public static List<List<Integer>> threeSum(int[] nums) {
        List<List<Integer>> result = new ArrayList<>();
        Arrays.sort(nums);
        for (int i = 0; i < nums.length - 2; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) {
            continue;
            }
            int left = i + 1, right = nums.length - 1;
            while (left < right) {
                int sum = nums[i] + nums[left] + nums[right];
                if (sum == 0) {
                    result.add(Arrays.asList(nums[i], nums[left], nums[right]));
                    while (left < right && nums[left] == nums[left + 1]) {
                        left++;
                        }
                    while (left < right && nums[right] == nums[right - 1]) {
                        right--;
                        }
                    left++;
                    right--;
                } else if (sum < 0) {
                    left++;
                } else {
                    right--;
                }
            }
```

```
        }
        return result;
    }

    public static void main(String[] args) {
        int[] nums = {-1, 0, 1, 2, -1, -4};
        List<List<Integer>> triplets = threeSum(nums);

        for (List<Integer> triplet : triplets) {
            System.out.println(triplet);
        }
    }
}
```

2C.Output :



3A.Program :

```
import java.util.Scanner;
public class PatternFinding {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of rows for triangle pattern: ");
        int rows = sc.nextInt();
        System.out.println("\nTriangle Pattern:");
        for (int i = 1; i <= rows; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
        sc.nextLine();
        System.out.print("\nEnter a text: ");
        String text = sc.nextLine();
        System.out.print("Enter a pattern to search: ");
        String pattern = sc.nextLine();
        if (text.contains(pattern)) {
            System.out.println("Pattern FOUND in the text.");
        } else {
            System.out.println(" Pattern NOT found in the text.");
        }
        sc.close();
    }
}
```

3A.Output :

```
Enter number of rows for triangle pattern: 5

Triangle Pattern:
*
* *
* * *
* * * *
* * * * *

Enter a text: Hello world,this is java
Enter a pattern to search: java
Pattern FOUND in the text.

=== Code Execution Successful ===
```

3B.Program :

```
import java.util.Scanner;
class Palindrome {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string or number: ");
        String input = sc.nextLine();
        String reversed = new StringBuilder(input).reverse().toString();
        if (input.equalsIgnoreCase(reversed)) {
            System.out.println("It is a palindrome");
        } else {
            System.out.println("It is not a palindrome");
        }
        sc.close();
    }
}
```

3B.Output :

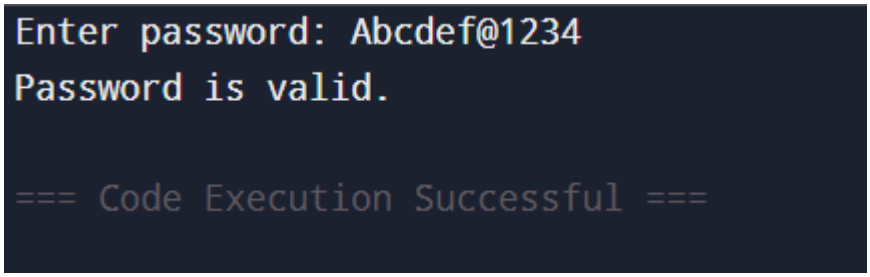
```
Enter a string or number: madam
It is a palindrome

=== Code Execution Successful ===
```

3C.Program :

```
import java.util.Scanner;
public class PasswordValidator {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter password: ");
        String password = sc.nextLine();
        if (validatePassword(password)) {
            System.out.println("Password is valid.");
        } else {
            System.out.println("Password is invalid.");
        }
    }
    public static boolean validatePassword(String password) {
        if (password.length() < 8) {
            System.out.println(" - Must be at least 8 characters long.");
            return false;
        }
        boolean hasUpper = false, hasLower = false, hasDigit = false, hasSpecial = false;
        String specialChars = "@#!$%^&*()-+";
        for (char c : password.toCharArray()) {
            if (Character.isUpperCase(c)) hasUpper = true;
            else if (Character.isLowerCase(c)) hasLower = true;
            else if (Character.isDigit(c)) hasDigit = true;
            else if (specialChars.contains(String.valueOf(c))) hasSpecial = true;
        }
        return hasUpper && hasLower && hasDigit && hasSpecial;
    }
}
```

3C.Output :

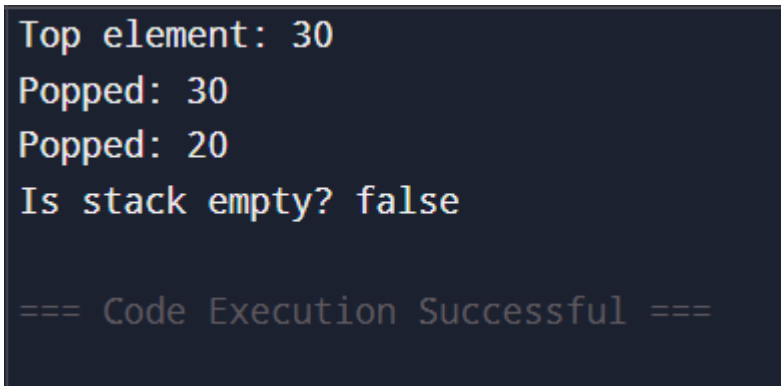

Enter password: Abcdef@1234
Password is valid.
=== Code Execution Successful ===

4A.Program :

```
import java.util.LinkedList;
import java.util.Queue;
class StackUsingTwoQueues {
    Queue<Integer> q1 = new LinkedList<>();
    Queue<Integer> q2 = new LinkedList<>();
    public void push(int x) {
        q2.add(x);
        while (!q1.isEmpty()) {
            q2.add(q1.poll());
        }
        Queue<Integer> temp = q1;
        q1 = q2;
        q2 = temp;
    }
    public int pop() {
        if (q1.isEmpty()) {
            System.out.println("Stack is empty");
            return -1;
        }
        return q1.poll();
    }
    public int peek() {
        if (q1.isEmpty()) {
            System.out.println("Stack is empty");
            return -1;
        }
        return q1.peek();
```

```
    }
    public boolean isEmpty() {
        return q1.isEmpty();
    }
    public static void main(String[] args) {
        StackUsingTwoQueues stack = new StackUsingTwoQueues();
        stack.push(10);
        stack.push(20);
        stack.push(30);
        System.out.println("Top element: " + stack.peek());
        System.out.println("Popped: " + stack.pop());
        System.out.println("Popped: " + stack.pop());
        System.out.println("Is stack empty? " + stack.isEmpty());
    }
}
```

4A.Output :


Top element: 30
Popped: 30
Popped: 20
Is stack empty? false
=== Code Execution Successful ===

4B.Program :

```
import java.util.ArrayList;
class BagOfNumbers {
    private ArrayList<Integer> bag = new ArrayList<>();
    public void add(int num) {
        bag.add(num);
    }
    public boolean remove(int num) {
        return bag.remove((Integer) num);
    }
    public boolean contains(int num) {
        return bag.contains(num);
    }
    public int size() {
        return bag.size();
    }
    public boolean isEmpty() {
        return bag.isEmpty();
    }
    public void display() {
        System.out.println("Bag: " + bag);
    }
    public static void main(String[] args) {
        BagOfNumbers bag = new BagOfNumbers();
        bag.add(5);
        bag.add(10);
        bag.add(5);
        bag.add(20);
```

```
        bag.display();
        bag.remove(5);
        bag.display();
        System.out.println("Contains 10? " + bag.contains(10));
        System.out.println("Bag size: " + bag.size());
        System.out.println("Is empty? " + bag.isEmpty());
    }
}
```

4B.Output :

```
Bag: [5, 10, 5, 20]
Bag: [10, 5, 20]
Contains 10? true
Bag size: 3
Is empty? false

=== Code Execution Successful ===
```

4C.Program :

```
import java.util.Scanner;

class DiskTower {
    private int[] disks;
    private int day = 0;
    private int expected;
    public DiskTower(int n) {
        disks = new int[n];
        expected = n;
    }
    public void placeDisk(int size) {
        disks[day++] = size;
        System.out.print("Day " + day + ": ");
        printTower(day - 1);
        System.out.println();
    }
    private void printTower(int upto) {
        if (upto < 0) return;
        if (disks[upto] == expected) {
            System.out.print(disks[upto] + " ");
            expected--;
            printTower(upto - 1);
        } else {
            printTower(upto - 1);
        }
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of disks: ");
        int n = sc.nextInt();
        DiskTower tower = new DiskTower(n);
        for (int i = 0; i < n; i++) {
            System.out.print("Enter disk size for Day " + (i + 1) + ": ");
            tower.placeDisk(sc.nextInt());
        }
        sc.close();
    }
}
```

4C.Output :

```
Enter number of disks: 4
Enter disk size for Day 1: 4
Day 1: 4
Enter disk size for Day 2: 2
Day 2:
Enter disk size for Day 3: 3
Day 3: 3 2
Enter disk size for Day 4: 1
Day 4: 1

=== Code Execution Successful ===
```

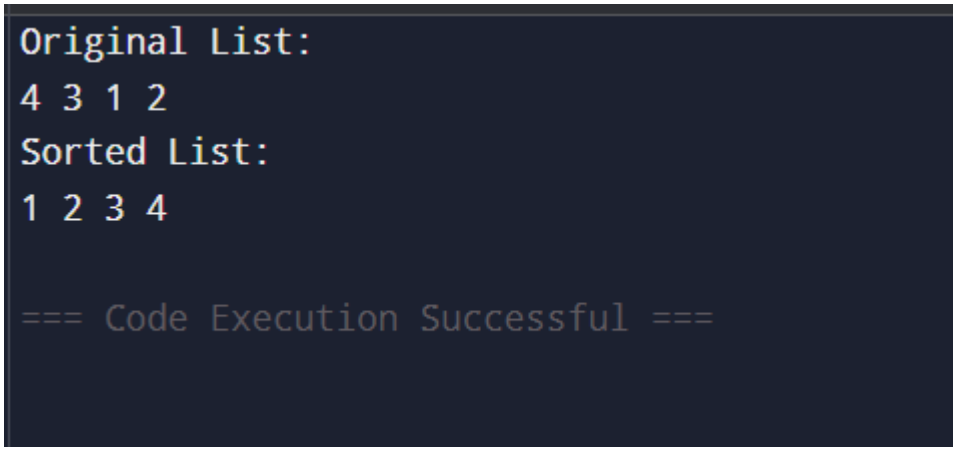
5A.Program :

```
class Node {
   int data;
   Node next;
   Node(int data) { this.data = data; }
}
class LinkedList {
   Node head;
   void insert(int data) {
      Node newNode = new Node(data);
      if (head == null) {
         head = newNode;
         return;
      }
      Node temp = head;
      while (temp.next != null) temp = temp.next;
      temp.next = newNode;
   }
   void insertionSort() {
      Node sorted = null;
      Node current = head;
      while (current != null) {
         Node next = current.next;
         sorted = sortedInsert(sorted, current);
         current = next;
      }
      head = sorted;
   }
```

```
    Node sortedInsert(Node sorted, Node newNode) {
        if (sorted == null || newNode.data < sorted.data) {
            newNode.next = sorted;
            return newNode;
        }
        Node temp = sorted;
        while (temp.next != null && temp.next.data < newNode.data) {
            temp = temp.next;
        }
        newNode.next = temp.next;
        temp.next = newNode;
        return sorted;
    }
    void printList() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}
public class InsertionSortLinkedList {
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.insert(4);
        list.insert(3);
        list.insert(1);
        list.insert(2);
```

```
        System.out.println("Original List:");
        list.printList();
        list.insertionSort();
        System.out.println("Sorted List:");
        list.printList();
    }
}
```

5A.Output :


Original List:
4 3 1 2
Sorted List:
1 2 3 4
=== Code Execution Successful ===

5B.Program :

```
class Node {
    int data;
    Node next;
    Node(int data) { this.data = data; }
}
class LinkedList {
    Node head;
    void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node temp = head;
        while (temp.next != null) temp = temp.next;
        temp.next = newNode;
    }
    void remove(int key) {
        if (head == null) return;
        if (head.data == key) {
            head = head.next;
            return;
        }
        Node temp = head;
        while (temp.next != null && temp.next.data != key) {
            temp = temp.next;
        }
```

```
        if (temp.next != null) {
            temp.next = temp.next.next;
        }
    }
    void printList() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}
public class RemoveElementLinkedList {
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.insert(10);
        list.insert(20);
        list.insert(30);
        list.insert(20);

        System.out.println("Original List:");
        list.printList();
        list.remove(20);
        System.out.println("After removing first occurrence of 20:");
        list.printList();
    }
}
```

5B.Output :

```
Original List:
10 20 30 20
After removing first occurrence of 20:
10 30 20

=== Code Execution Successful ===
```

5C.Program :

```
import java.util.HashSet;
class Node {
    int data;
    Node next;
    Node(int data) { this.data = data; }
}
class LinkedList {
    Node head;
    void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node temp = head;
        while (temp.next != null) temp = temp.next;
        temp.next = newNode;
    }
    void removeDuplicates() {
        HashSet<Integer> set = new HashSet<>();
        Node current = head;
        Node prev = null;
        while (current != null) {
            if (set.contains(current.data)) {
                prev.next = current.next;
            } else {
                set.add(current.data);
```

```
                prev = current;
            }
            current = current.next;
        }
    }
    void printList() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}
public class RemoveDuplicatesLinkedList {
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.insert(1);
        list.insert(2);
        list.insert(2);
        list.insert(3);
        list.insert(1);
        System.out.println("Original List:");
        list.printList();
        list.removeDuplicates();
        System.out.println("After removing duplicates:");
        list.printList();
    }
}
```

5C.Output :

```
Original List:
1 2 2 3 1
After removing duplicates:
1 2 3

=== Code Execution Successful ===
```