

ASSIGNMENT-14

1. Generate a code to non-negative integer's num1 and num2 represented as strings; return the product of num1 and num2, also represented as a string.

```
public class Main {
    public static void main(String[] args) {
        String num1 = "123";
        String num2 = "456";
        String product = multiply(num1, num2);
        System.out.println(product);
    }

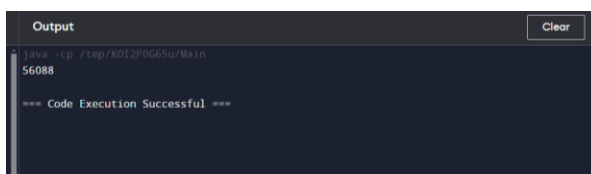
    public static String multiply(String num1, String num2) {
        int m = num1.length(), n = num2.length();
        int[] pos = new int[m + n];

        for (int i = m - 1; i >= 0; i--) {
            for (int j = n - 1; j >= 0; j--) {
                int mul = (num1.charAt(i) - '0') * (num2.charAt(j) - '0');
                int p1 = i + j, p2 = i + j + 1;
                int sum = mul + pos[p2];

                pos[p1] += sum / 10;
                pos[p2] = sum % 10;
            }
        }

        StringBuilder sb = new StringBuilder();
        for (int p : pos) {
            if (!(sb.length() == 0 && p == 0)) {
                sb.append(p);
            }
        }

        return sb.length() == 0 ? "0" : sb.toString();
    }
}
```



The screenshot shows a dark-themed output window with a 'Clear' button in the top right corner. The text inside the window displays the command 'java -cp /tmp/R012P066Su/Main', the output '56088', and a status message '=== Code Execution Successful ==='.

```
Output
Clear
java -cp /tmp/R012P066Su/Main
56088
=== Code Execution Successful ===
```

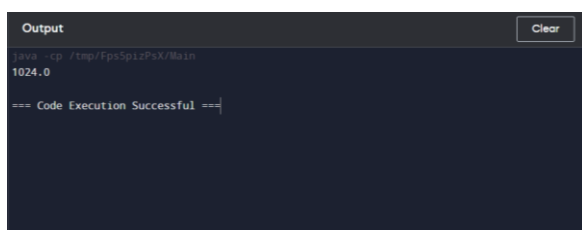
1. Implement `pow(x, n)`, which calculates `x` raised to the power `n`

Input: `x = 2.00000`, `n = 10`

Output: `1024.00000`

```
class Solution {
    public double myPow(double x, int n) {
        if (n == 0) return 1;
        if (n < 0) {
            x = 1 / x;
            n = -n;
        }
        double result = 1;
        while (n > 0) {
            if (n % 2 == 1) {
                result *= x;
            }
            x *= x;
            n /= 2;
        }
        return result;
    }
}

public class Main {
    public static void main(String[] args) {
        Solution solution = new Solution();
        double x = 2.00000;
        int n = 10;
        System.out.println(solution.myPow(x, n));
    }
}
```

A screenshot of a code execution environment. At the top, there's a title bar with the word "Output" and a "Clear" button. Below the title bar, the output text shows the command `java -cp ./tmp/FpsSpizPsX/Main` followed by the result `1024.0`. At the bottom, a status message reads `=== Code Execution Successful ===`.

2. Given an integer array `nums`, find the subarray with the largest sum, and return its sum. Input:

`nums = [-2,1,-3,4,-1,2,1,-5,4]`

Output: `6`

```
public class Main {
    public static int maxSubArray(int[] nums) {
        int maxSum = nums[0];
        int currentSum = nums[0];
```

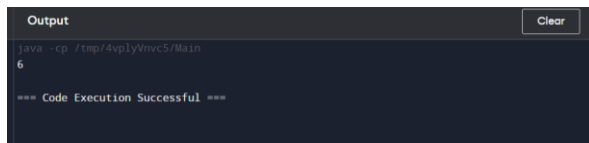
```

        for (int i = 1; i < nums.length; i++) {
            currentSum = Math.max(nums[i], currentSum + nums[i]);
            maxSum = Math.max(maxSum, currentSum);
        }

        return maxSum;
    }

    public static void main(String[] args) {
        int[] nums = {-2, 1, -3, 4, -1, 2, 1, -5, 4};
        System.out.println(maxSubArray(nums)); // Output: 6
    }
}

```



```

Output
java -cp /tmp/4uplymcc5/Main
6
=== Code Execution Successful ===

```

3. Write a Java program which creates only one object. If user attempts to create second object, he should not be able to create it. (Using Exception Handling).
4. There is an exam room with n seats in a single row labeled from 0 to $n - 1$. When a student enters the room, they must sit in the seat that maximizes the distance to the closest person. If there are multiple such seats, they sit in the seat with the lowest number. If no one is in the room, then the student sits at seat number 0. Design a class that simulates the mentioned exam room. Implement the ExamRoom class: ExamRoom (int n) Initializes the object of the exam room with the number of the seats n . int seat () Returns the label of the seat at which the next student will sit. Void leave (int p) indicates that the student sitting at seat p will leave the room. It is guaranteed that there will be a student sitting at seat p .

Input["ExamRoom", "seat", "seat", "seat", "seat", "leave", "seat"]

[[10], [], [], [], [], [4], []]

Output

[null, 0, 9, 4, 2, null, 5]

```
import java.util.TreeSet;
```

```

class ExamRoom {
    private int n;
    private TreeSet<Integer> students;

    public ExamRoom(int n) {
        this.n = n;
        students = new TreeSet<>();
    }
}

```

```

    }

    public int seat() {
        if (students.isEmpty()) {
            students.add(0);
            return 0;
        }

        int prev = -1;
        int maxDist = students.first();
        int seat = 0;

        for (int s : students) {
            if (prev != -1) {
                int dist = (s - prev) / 2;
                if (dist > maxDist) {
                    maxDist = dist;
                    seat = prev + dist;
                }
            }
            prev = s;
        }

        if (n - 1 - students.last() > maxDist) {
            seat = n - 1;
        }

        students.add(seat);
        return seat;
    }

    public void leave(int p) {
        students.remove(p);
    }
}

public class Main {
    public static void main(String[] args) {
        ExamRoom examRoom = new ExamRoom(10);
        System.out.println(examRoom.seat());
        System.out.println(examRoom.seat());
        System.out.println(examRoom.seat());
        System.out.println(examRoom.seat());
        examRoom.leave(4);
        System.out.println(examRoom.seat());
    }
}

```

OutputClear

```
java -cp ./tmp/y1Fn3fDEK3/Main  
0  
9  
4  
2  
5  
  
=== Code Execution Successful ===
```