```
import java.util.Scanner;

public class WorstFit {

    // Function to implement Worst Fit algorithm
    static void worstFit(int blocks[], int m, int processes[], int n) {

        // Array to store the allocation of processes
        int[] allocation = new int[n];

        // Initialize all allocations as -1 (meaning no allocation)
        for (int i = 0; i < n; i++) {
            allocation[i] = -1;
        }

        // Traverse through all processes and find the worst block for each process
        for (int i = 0; i < n; i++) {
            // Find the block with the maximum size that can accommodate the current process
            int maxSize = -1;
            int worstBlock = -1;

            for (int j = 0; j < m; j++) {
                if (blocks[j] >= processes[i] && blocks[j] > maxSize) {
                    maxSize = blocks[j];
                    worstBlock = j;
                }
            }

            // If we found a block for the process, allocate it
```

```
        if (worstBlock != -1) {
            allocation[i] = worstBlock;
            blocks[worstBlock] -= processes[i]; // Reduce the available memory in the block
        }
    }

    // Display the results
    System.out.println("Process No. | Process Size | Block No. | Remaining Block Size");
    for (int i = 0; i < n; i++) {
        if (allocation[i] != -1) {
            System.out.println((i + 1) + "\t\t" + processes[i] + "\t\t" + (allocation[i] + 1) + "\t\t" +
blocks[allocation[i]]);
        } else {
            System.out.println((i + 1) + "\t\t" + processes[i] + "\t\t" + "Not Allocated");
        }
    }
  }

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

    // Input the number of blocks
    System.out.print("Enter the number of memory blocks: ");
    int m = scanner.nextInt();

    // Input the sizes of the blocks
    int[] blocks = new int[m];
    System.out.println("Enter the sizes of the memory blocks:");
    for (int i = 0; i < m; i++) {
```

```
            blocks[i] = scanner.nextInt();
        }

        // Input the number of processes
        System.out.print("Enter the number of processes: ");
        int n = scanner.nextInt();

        // Input the sizes of the processes
        int[] processes = new int[n];
        System.out.println("Enter the sizes of the processes:");
        for (int i = 0; i < n; i++) {
            processes[i] = scanner.nextInt();
        }

        // Call the Worst Fit allocation function
        worstFit(blocks, m, processes, n);

        scanner.close();
    }
}
```

```
java -cp /tmp/HYjzC7E2iY/WorstFit
Enter the number of memory blocks: 4
Enter the sizes of the memory blocks:
6
4
8
5
Enter the number of processes: 3
Enter the sizes of the processes:
3
7
3
Process No. | Process Size | Block No. | Remaining Block Size
1       3       3       5
2       7       Not Allocated
3       3       1       3

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