MemoryAllo-Worst Fit – 10

```
import java.util.*;

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

        System.out.print("Enter number of memory blocks: ");
        int nb = sc.nextInt();
        int block[] = new int[nb], rem[] = new int[nb];

        System.out.println("Enter block sizes:");
        for(int i=0;i<nb;i++){
            block[i] = sc.nextInt();
            rem[i] = block[i];
        }

        System.out.print("Enter number of processes: ");
        int np = sc.nextInt();
        int process[] = new int[np], alloc[] = new int[np];

        System.out.println("Enter process sizes:");
        for(int i=0;i<np;i++){
            process[i] = sc.nextInt();
            alloc[i] = -1;
        }

        // Worst Fit Allocation: choose largest available block
        for(int i = 0; i < np; i++) {
            int worstIndex = -1;
            for(int j = 0; j < nb; j++) {
```

```
            if(rem[j] >= process[i]) {
                if(worstIndex == -1 || rem[j] > rem[worstIndex])
                    worstIndex = j;
            }
        }
        if(worstIndex != -1) {
            alloc[i] = worstIndex;
            rem[worstIndex] -= process[i];
        }
    }

    // Display Allocation Results
    System.out.println("\nProcess\tSize\tBlock Allocated");
    for(int i = 0; i < np; i++) {
        if(alloc[i] != -1)
            System.out.println("P" + (i+1) + "\t" + process[i] + "\tBlock " + (alloc[i]+1));
        else
            System.out.println("P" + (i+1) + "\t" + process[i] + "\tNot Allocated");
    }

    // Display Fragmentation / unused space
    System.out.println("\nBlock\tInitial Size\tRemaining Space (Unused)");
    int totalUnused = 0;
    for(int i = 0; i < nb; i++) {
        System.out.println("B" + (i+1) + "\t" + block[i] + "\t\t" + rem[i]);
        totalUnused += rem[i];
    }

    System.out.println("\nTotal Unused Memory: " + totalUnused);
  }
}
```