Practical No : 06
Name :Kiran Dadarao Janjal
Roll No : 13216
Batch : B1

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

class MemoryPlacement {
    static void firstFit(int blockSize[], int processSize[]) {
        int m = blockSize.length;
        int n = processSize.length;
        int allocation[] = new int[n];
        Arrays.fill(allocation, -1);

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                if (blockSize[j] >= processSize[i]) {
                    allocation[i] = j;
                    blockSize[j] -= processSize[i];
                    break;
                }
            }
        }

        System.out.println("First Fit Allocation:");
        printAllocation(processSize, allocation);
    }

    static void bestFit(int blockSize[], int processSize[]) {
        int m = blockSize.length;
        int n = processSize.length;
        int allocation[] = new int[n];
        Arrays.fill(allocation, -1);

        for (int i = 0; i < n; i++) {
            int bestIdx = -1;
            for (int j = 0; j < m; j++) {
                if (blockSize[j] >= processSize[i]) {
                    if (bestIdx == -1 || blockSize[j] < blockSize[bestIdx])
                        bestIdx = j;
                }
            }
            if (bestIdx != -1) {
                allocation[i] = bestIdx;
                blockSize[bestIdx] -= processSize[i];
            }
        }

        System.out.println("Best Fit Allocation:");
        printAllocation(processSize, allocation);
    }
    static void worstFit(int blockSize[], int processSize[]) {
        int m = blockSize.length;
        int n = processSize.length;
        int allocation[] = new int[n];
        Arrays.fill(allocation, -1);

        for (int i = 0; i < n; i++) {
            int worstIdx = -1;
            for (int j = 0; j < m; j++) {
                if (blockSize[j] >= processSize[i]) {
                    if (worstIdx == -1 || blockSize[j] > blockSize[worstIdx])
                        worstIdx = j;
                }
            }
            if (worstIdx != -1) {
                allocation[i] = worstIdx;
                blockSize[worstIdx] -= processSize[i];
            }
        }

        System.out.println("Worst Fit Allocation:");
        printAllocation(processSize, allocation);
    }

    static void nextFit(int blockSize[], int processSize[]) {
        int m = blockSize.length;
        int n = processSize.length;
        int allocation[] = new int[n];
        Arrays.fill(allocation, -1);
        int j = 0;

        for (int i = 0; i < n; i++) {
            int count = 0;
            while (count < m) {
                if (blockSize[j] >= processSize[i]) {
                    allocation[i] = j;
                    blockSize[j] -= processSize[i];
                    break;
                }
                j = (j + 1) % m;
                count++;
            }
        }

        System.out.println("Next Fit Allocation:");
        printAllocation(processSize, allocation);
    }

    static void printAllocation(int processSize[], int allocation[]) {
        System.out.println("Process No.\tProcess Size\tBlock No.");
        for (int i = 0; i < processSize.length; i++) {
            System.out.print(" " + (i + 1) + "\t\t" + processSize[i] + "\t\t");
            if (allocation[i] != -1)
```

```
                System.out.println(allocation[i] + 1);
            else
                System.out.println("Not Allocated");
        }
        System.out.println();
    }

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

        System.out.print("Enter number of memory blocks: ");
        int m = sc.nextInt();
        int[] blockSize = new int[m];
        System.out.println("Enter block sizes:");
        for (int i = 0; i < m; i++) {
            blockSize[i] = sc.nextInt();
        }

        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();
        int[] processSize = new int[n];
        System.out.println("Enter process sizes:");
        for (int i = 0; i < n; i++) {
            processSize[i] = sc.nextInt();
        }

        firstFit(blockSize.clone(), processSize);
        bestFit(blockSize.clone(), processSize);
        worstFit(blockSize.clone(), processSize);
        nextFit(blockSize.clone(), processSize);

        sc.close();
    }
}
```

```
OUTPUT :=== Enter block sizes:
100 200 300 400 230
Enter number of processes: 5
Enter process sizes:
200 400 340 500 400
First Fit Allocation:
Process No.     Process Size    Block No.
 1              200             2
 2              400             4
 3              340             Not Allocated
 4              500             Not Allocated
 5              400             Not Allocated

Best Fit Allocation:
Process No.     Process Size    Block No.
 1              200             2
 2               400             4
 3               340             Not Allocated
 4               500             Not Allocated
 5               400             Not Allocated

Worst Fit Allocation:
Process No.     Process Size    Block No.
 1               200             4
 2               400             Not Allocated
 3               340             Not Allocated
 4               500             Not Allocated
 5               400             Not Allocated

Next Fit Allocation:
Process No.     Process Size    Block No.
 1               200             2
 2               400             4
 3               340             Not Allocated
 4               500             Not Allocated
 5               400             Not Allocated
```