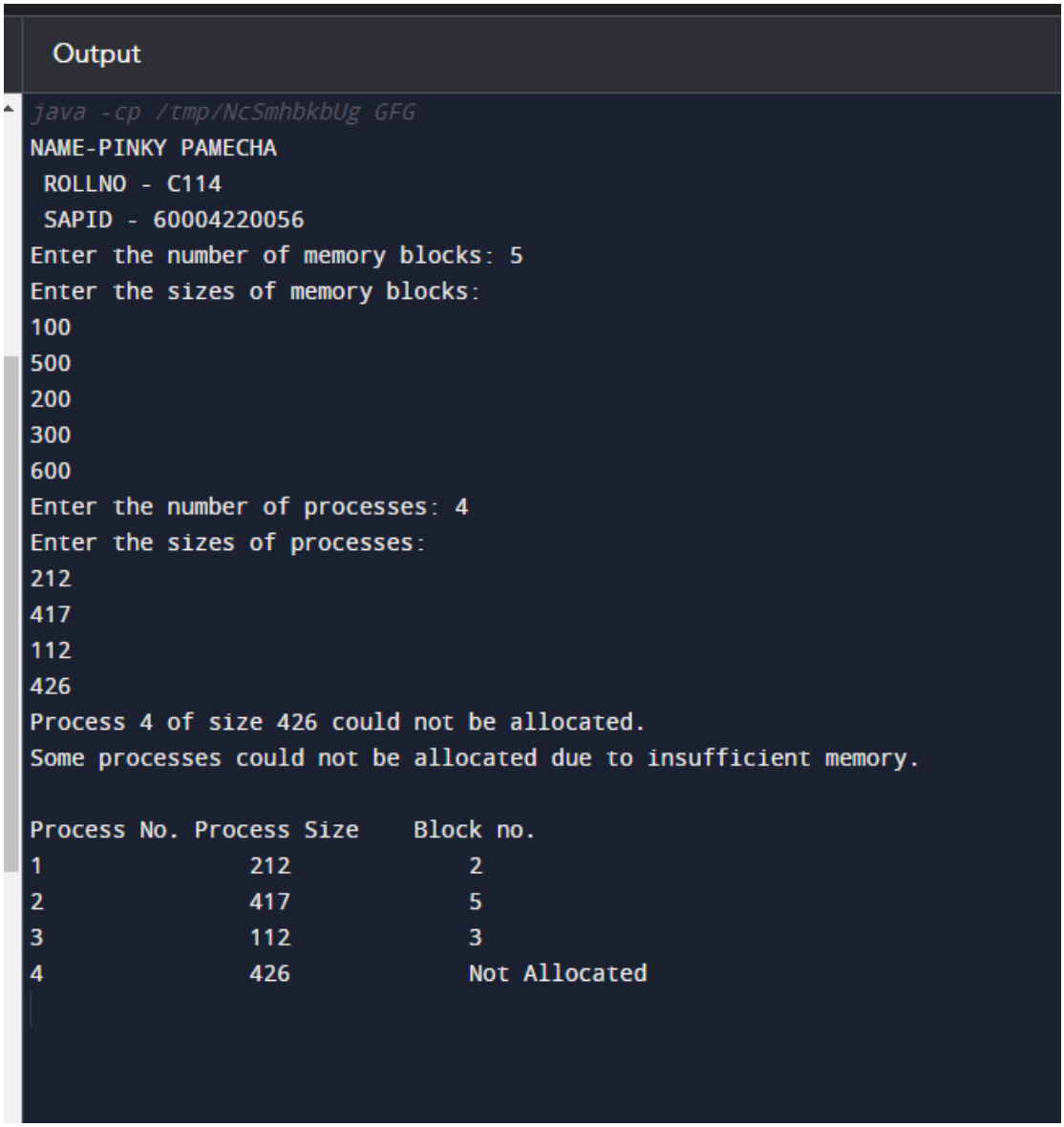
NAME – PINKY PAMECHA
ROLL NO – C114
SAPID – 60004220056

BEST FIT:
Code:

```
import java.util.*;

public class GFG {

   // Function to allocate memory to blocks as per First fit algorithm
   static void FirstFit(int blockSize[], int m, int processSize[], int n) {
      int allocation[] = new int[n];
      Arrays.fill(allocation, -1);

      // Flag to mark blocks as used
      boolean[] blockUsed = new boolean[m];

      for (int i = 0; i < n; i++) {
         for (int j = 0; j < m; j++) {
            if (!blockUsed[j] && blockSize[j] >= processSize[i]) {
               allocation[i] = j;
               blockSize[j] -= processSize[i];
               blockUsed[j] = true;
               break; // Break the loop after allocation
            }
         }
      }

      // Check if any process remains unallocated due to insufficient memory blocks
      boolean allAllocated = true;
      for (int i = 0; i < n; i++) {
         if (allocation[i] == -1) {
            System.out.println("Process " + (i + 1) + " of size " + processSize[i] + " could not be
allocated.");
            allAllocated = false;
         }
      }

      // Check for duplicate block allocations
      for (int i = 0; i < n; i++) {
         for (int j = i + 1; j < n; j++) {
            if (allocation[i] != -1 && allocation[j] != -1 && allocation[i] == allocation[j]) {
               System.out.println("Error: Process " + (i + 1) + " and Process " + (j + 1) + " are both
allocated to block " + (allocation[i] + 1));
            }
         }
      }

      if (allAllocated) {
         System.out.println("All processes are successfully allocated.");
```

```
        } else {
            System.out.println("Some processes could not be allocated due to insufficient memory.");
        }

        System.out.print("\nProcess No.\tProcess Size\tBlock no.\n");
        for (int i = 0; i < n; i++) {
            System.out.print(i + 1 + "\t\t\t\t" + processSize[i] + "\t\t\t\t");
            if (allocation[i] != -1) {
                System.out.print(allocation[i] + 1);
            } else {
                System.out.print("Not Allocated");
            }
            System.out.println("");
        }
    }

    // Driver program
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("NAME-PINKY PAMECHA \n ROLLNO - C114 \n SAPID - 60004220056");
        System.out.print("Enter the number of memory blocks: ");
        int m = scanner.nextInt();
        int blockSize[] = new int[m];
        System.out.println("Enter the sizes of memory blocks:");
        for (int i = 0; i < m; i++) {
            blockSize[i] = scanner.nextInt();
        }

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

        FirstFit(blockSize, m, processSize, n);
        scanner.close();
    }
}
```

Output:



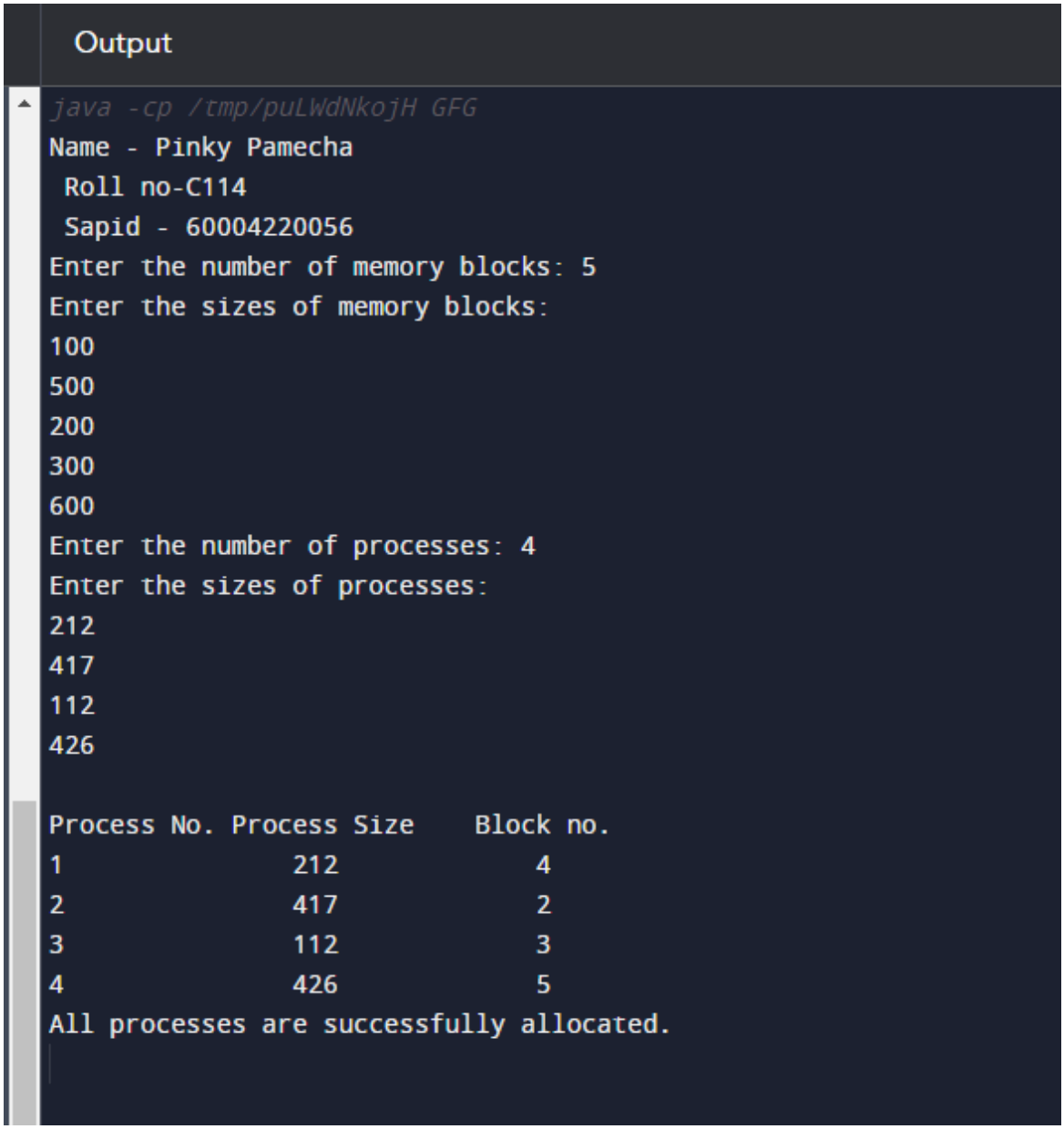
BEST FIT:
Code:

```
import java.util.*;

public class GFG {

    // Function to allocate memory to blocks as per Best fit algorithm
    static void BestFit(int blockSize[], int m, int processSize[], int n) {
        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];
        }
    }

    // Check if any process remains unallocated due to insufficient memory blocks
    boolean allAllocated = true;
    for (int i = 0; i < n; i++) {
        if (allocation[i] == -1) {
            System.out.println("Process " + (i + 1) + " of size " + processSize[i] + " could not be
allocated.");
            allAllocated = false;
        }
    }

    System.out.print("\nProcess No.\tProcess Size\tBlock no.\n");
    for (int i = 0; i < n; i++) {
        System.out.print(i + 1 + "\t\t\t\t" + processSize[i] + "\t\t\t\t");
        if (allocation[i] != -1) {
            System.out.print(allocation[i] + 1);
        } else {
            System.out.print("Not Allocated");
        }
        System.out.println("");
    }

    if (allAllocated) {
        System.out.println("All processes are successfully allocated.");
    } else {
        System.out.println("Some processes could not be allocated due to insufficient memory.");
    }
  }

  // Driver program
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Name - Pinky Pamecha \n Roll no-C114 \n Sapid - 60004220056");
    System.out.print("Enter the number of memory blocks: ");
    int m = scanner.nextInt();
    int blockSize[] = new int[m];
    System.out.println("Enter the sizes of memory blocks:");
    for (int i = 0; i < m; i++) {
        blockSize[i] = scanner.nextInt();
    }

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

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

        BestFit(blockSize, m, processSize, n);
        scanner.close();
    }
}
```

Output:

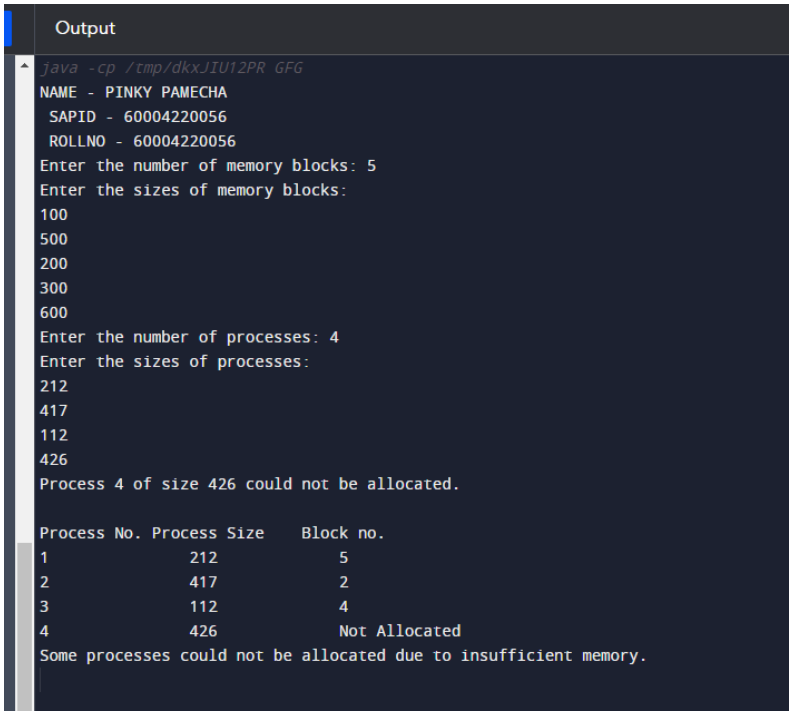


WORST FIT:
Code:

```
import java.util.*;

public class GFG {

   // Function to allocate memory to blocks as per Worst fit algorithm
   static void WorstFit(int blockSize[], int m, int processSize[], int n) {
      int allocation[] = new int[n];
      Arrays.fill(allocation, -1);

      // Flag to mark blocks as used
      boolean[] blockUsed = new boolean[m];

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

      // Check if any process remains unallocated due to insufficient memory blocks
      boolean allAllocated = true;
      for (int i = 0; i < n; i++) {
         if (allocation[i] == -1) {
            System.out.println("Process " + (i + 1) + " of size " + processSize[i] + " could not be
allocated.");
            allAllocated = false;
         }
      }

      System.out.print("\nProcess No.\tProcess Size\tBlock no.\n");
      for (int i = 0; i < n; i++) {
         System.out.print(i + 1 + "\t\t\t\t" + processSize[i] + "\t\t\t\t");
         if (allocation[i] != -1) {
            System.out.print(allocation[i] + 1);
         } else {
            System.out.print("Not Allocated");
         }
         System.out.println("");
      }

      if (allAllocated) {
         System.out.println("All processes are successfully allocated.");
```

```
        } else {
            System.out.println("Some processes could not be allocated due to insufficient memory.");
        }
    }

    // Driver program
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("NAME - PINKY PAMECHA \n SAPID - 60004220056 \n ROLLNO - 60004220056");
        System.out.print("Enter the number of memory blocks: ");
        int m = scanner.nextInt();
        int blockSize[] = new int[m];
        System.out.println("Enter the sizes of memory blocks:");
        for (int i = 0; i < m; i++) {
            blockSize[i] = scanner.nextInt();
        }

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

        WorstFit(blockSize, m, processSize, n);
        scanner.close();
    }
}
```

Output:



NEXT FIT:
Code:

```
import java.util.*;

public class GFG {

   // Function to allocate memory to blocks as per Next fit algorithm
   static void NextFit(int blockSize[], int m, int processSize[], int n) {
      int allocation[] = new int[n];
      Arrays.fill(allocation, -1);

      int j = 0; // Start searching from the beginning

      for (int i = 0; i < n; i++) {
         while (true) {
            if (blockSize[j] >= processSize[i]) {
               allocation[i] = j;
               blockSize[j] -= processSize[i];
               break;
            }
            j = (j + 1) % m; // Move to the next block
            // If we come back to the starting block, break the loop
            if (j == 0) break;
         }
      }

      System.out.print("\nProcess No.\tProcess Size\tBlock no.\n");
      for (int i = 0; i < n; i++) {
         System.out.print(i + 1 + "\t\t\t\t" + processSize[i] + "\t\t\t\t");
         if (allocation[i] != -1) {
            System.out.print(allocation[i] + 1);
         } else {
            System.out.print("Not Allocated");
         }
         System.out.println("");
      }
   }

   // Driver program
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Name - Pinky Pamecha \n Roll no - C114 \n Sap id - 60004220056");
      System.out.print("Enter the number of memory blocks: ");
      int m = scanner.nextInt();
      int blockSize[] = new int[m];
      System.out.println("Enter the sizes of memory blocks:");
      for (int i = 0; i < m; i++) {
         blockSize[i] = scanner.nextInt();
      }

      System.out.print("Enter the number of processes: ");
```

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

        NextFit(blockSize, m, processSize, n);
        scanner.close();
    }
}
```

Output:

Output

```
java -cp /tmp/Tjzvd43TSZ GFG
Name - Pinky Pamecha
 Roll no - C114
 Sap id - 60004220056
Enter the number of memory blocks: 3
Enter the sizes of memory blocks:
5
10
15
Enter the number of processes: 3
Enter the sizes of processes:
9
12
4

Process No. Process Size    Block no.
1               9               2
2               12              3
3               4               Not Allocated
```