Practical-6

Q. Write a program to simulate Memory placement strategies - best fit, first fit, next fit and worst fit

//best fit

```
import java.util.Scanner;

public class BestFit {
      static void bestFit(int blockSize[], int m, int processSize[], int
n, int remblockSize[]) {
            int allocation[] = new int[n];
            for (int i = 0; i < allocation.length; i++) {
                  allocation[i] = -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)
                                    bestIdx = j;
                              else if (blockSize[bestIdx] > blockSize[j])
                                    bestIdx = j;
                        }
                  }
                  if (bestIdx != -1) {
                        allocation[i] = bestIdx;
                        blockSize[bestIdx] -= processSize[i];
                        remblockSize[i] = blockSize[bestIdx];
                  }
            }

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

      public static void main(String[] args) {
            int m, n, num;
            Scanner in = new Scanner(System.in);
            System.out.print("Enter how many number of blocks you want to
enter:");
            m = in.nextInt();
            int remblockSize[] = new int[m];
            int blockSize[] = new int[m];
```

```
            for (int i = 0; i < m; i++) {
                  System.out.print("Enter Data " + (i + 1) + ":");
                  num = in.nextInt();
                  blockSize[i] = num;
            }
            System.out.print("Enter how many number of process you want
to enter:");
            n = in.nextInt();
            int processSize[] = new int[n];
            for (int i = 0; i < n; i++) {
                  System.out.print("Enter Data " + (i + 1) + ":");
                  num = in.nextInt();
                  processSize[i] = num;
            }
            bestFit(blockSize, m, processSize, n, remblockSize);

            in.close();
      }

}
```

//first fit

```
import java.util.Scanner;

public class FirstFit {
      static void firstFit(int blockSize[], int m, int processSize[], int
n, int remblockSize[]) {
            int allocation[] = new int[n];
            for (int i = 0; i < allocation.length; i++) {
                  allocation[i] = -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];
                              remblockSize[i] = blockSize[j];
                              break;
                        }
                  }
            }
            System.out.println("\nProcess No.\tProcess Size\tBlock
no.\tRemaninig Block Size");
            for (int i = 0; i < n; i++) {
                  System.out.print(" " + (i + 1) + "\t\t" + processSize[i]
+ "\t\t");
                  if (allocation[i] != -1) {
                        System.out.print((allocation[i] + 1) + "\t\t" +
remblockSize[i]);
                  } else {
                        System.out.print("Not Allocated" + "\t" +
remblockSize[i]);
                  }
                  System.out.println();
```

```
            }
      }

      public static void main(String[] args) {
            int m, n, num;
            Scanner in = new Scanner(System.in);
            System.out.print("Enter how many number of blocks you want to
enter:");
            m = in.nextInt();
            int blockSize[] = new int[m];
            int remblockSize[] = new int[m];
            for (int i = 0; i < m; i++) {
                  System.out.print("Enter Data " + (i + 1) + ":");
                  num = in.nextInt();
                  blockSize[i] = num;
            }
            System.out.print("Enter how many number of process you want
to enter:");
            n = in.nextInt();
            int processSize[] = new int[n];
            for (int i = 0; i < n; i++) {
                  System.out.print("Enter Data " + (i + 1) + ":");
                  num = in.nextInt();
                  processSize[i] = num;
            }
            firstFit(blockSize, m, processSize, n, remblockSize);
            in.close();
      }

}
```

//next fit

```
import java.util.Arrays;
import java.util.Scanner;

public class NextFit {

      static void NextFit(int blockSize[], int m, int processSize[], int
n, int remblockSize[]) {

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

```
                }
            }

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

      public static void main(String[] args) {
            int m, n, num;
            Scanner in = new Scanner(System.in);
            System.out.print("Enter how many number of blocks you want to
enter:");
            m = in.nextInt();
            int blockSize[] = new int[m];
            int remblockSize[] = new int[m];
            for (int i = 0; i < m; i++) {
                System.out.print("Enter Data " + (i + 1) + ":");
                num = in.nextInt();
                blockSize[i] = num;
            }
            System.out.print("Enter how many number of process you want
to enter:");
            n = in.nextInt();
            int processSize[] = new int[n];
            for (int i = 0; i < n; i++) {
                System.out.print("Enter Data " + (i + 1) + ":");
                num = in.nextInt();
                processSize[i] = num;
            }
            NextFit(blockSize, m, processSize, n, remblockSize);
            in.close();
      }

}
```

//worst fit

```
import java.util.Scanner;

public class WorstFit {

      static void worstFit(int blockSize[], int m, int processSize[], int
n, int remblockSize[]) {
```

```
            int allocation[] = new int[n];
            for (int i = 0; i < allocation.length; i++) {
                  allocation[i] = -1;
            }
            for (int i = 0; i < n; i++) {
                  int wstIdx = -1;
                  for (int j = 0; j < m; j++) {
                        if (blockSize[j] >= processSize[i]) {
                              if (wstIdx == -1)
                                    wstIdx = j;
                              else if (blockSize[wstIdx] < blockSize[j])
                                    wstIdx = j;
                        }
                  }
                  if (wstIdx != -1) {
                        allocation[i] = wstIdx;
                        blockSize[wstIdx] -= processSize[i];
                        remblockSize[i] = blockSize[wstIdx];
                  }
            }

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

      public static void main(String[] args) {
            int m, n, num;
            Scanner in = new Scanner(System.in);
            System.out.print("Enter how many number of blocks you want to
enter:");
            m = in.nextInt();
            int remblockSize[] = new int[m];
            int blockSize[] = new int[m];
            for (int i = 0; i < m; i++) {
                  System.out.print("Enter Data " + (i + 1) + ":");
                  num = in.nextInt();
                  blockSize[i] = num;
            }
            System.out.print("Enter how many number of process you want
to enter:");
            n = in.nextInt();
            int processSize[] = new int[n];
            for (int i = 0; i < n; i++) {
                  System.out.print("Enter Data " + (i + 1) + ":");
                  num = in.nextInt();
                  processSize[i] = num;
            }
```

```
            worstFit(blockSize, m, processSize, n, remblockSize);
            in.close();
        }

}
```