**Lab 10: Page Replacement Algorithms**

|  |
| --- |
| 1. **First In First Out**   import java.util.Scanner;  public class FIFO {  public static void main(String[] args) {  Scanner input = new Scanner(System.in);  // Inputs  System.out.print("Enter Page frame: ");  int page = input.nextInt();  System.out.print("Enter number of entries in queue: ");  int entry = input.nextInt();  int[] frame = new int[page];  int[] entries = new int[entry];  for (int i = 0; i < entry; i++) {  System.out.print("Enter value of entry " + (i + 1) + ": ");  entries[i] = input.nextInt();  }  for (int i = 0; i < page; i++) {  frame[i] = -1;  }  int miss = 0;  int next = 0;  for (int i = 0; i < entry; i++) {  int count = 0;  for (int j = 0; j < page; j++) {  if (frame[j] == entries[i]) {  break;  }  count++;  }  if (count == page) {  miss++;  frame[next] = entries[i];  next = (next + 1) % page;  }  }  System.out.println("Page Faults: " + miss);  System.out.println("Page Hits: " + (entry -miss));  input.close();  }  } |
| **Output:** |
| 1. **Least Recently Used**   import java.util.Scanner;  public class LRU {  public static void main(String[] args) {  Scanner input = new Scanner(System.in);  // Inputs  System.out.print("Enter Page frame: ");  int page = input.nextInt();  System.out.print("Enter number of entries in queue: ");  int entry = input.nextInt();  int[] frame = new int[page];  int[] entries = new int[entry];  for (int i = 0; i < entry; i++) {  System.out.print("Enter value of entry " + (i + 1) + ": ");  entries[i] = input.nextInt();  }  for (int i = 0; i < page; i++) {  frame[i] = -1;  }  int miss = page;  int[] used = new int[page];  for (int i = 0; i < page; i++) {  frame[i] = entries[i];  }  for (int i = page; i < entry; i++) {  int count = 0;  for (int j = 0; j < page; j++) {  if (entries[i] == frame[j]) {  for (int k = 0; k < page; k++) {  used[k]++;  }  used[j] = 0;  break;  }  count++;  }  if (count == page) {  for (int j = 0; j < page; j++) {  used[j]++;  }  int max = Integer.MIN\_VALUE;  int maxIndex = 0;  for (int j = 0; j < page; j++) {  if (used[j] > max) {  max = used[j];  maxIndex = j;  }  }  used[maxIndex] = 0;  frame[maxIndex] = entries[i];  miss++;  }  }  System.out.println("Page Faults: " + miss);  System.out.println("Page Hits: " + (entry -miss));  input.close();  }  } |
| **Output:** |
| 1. **Optimal Algorithm**   import java.util.Scanner;  public class Optimal {      public static void main(String[] args) {          Scanner input = new Scanner(System.in);            // Inputs          System.out.print("Enter Page frame: ");          int page = input.nextInt();          System.out.print("Enter number of entries in queue: ");          int entry = input.nextInt();          int[] frame = new int[page];          int[] entries = new int[entry];          for (int i = 0; i < entry; i++) {              System.out.print("Enter value of entry " + (i + 1) + ": ");              entries[i] = input.nextInt();          }          for (int i = 0; i < page; i++) {              frame[i] = -1;          }          int miss = page;          for (int i = 0; i < page; i++) {              frame[i] = entries[i];          }          for (int i = page; i < entry; i++) {              int count = 0;              for (int j = 0; j < page; j++) {                  if (frame[j] == entries[i]) {                      break;                  }                  count++;              }              if (count == page) {                  int[] use = new int[page];                  for (int j = i; j < entry; j++) {                      for (int k = 0; k < page; k++) {                          if (frame[k] == entries[j]) {                              use[k]++;                          }                      }                  }                  int min = Integer.MAX\_VALUE;                  int minIndex = 0;                  for (int j = 0; j < page; j++) {                      if (use[j] < min) {                          min = use[j];                          minIndex = j;                      }                  }                  frame[minIndex] = entries[i];                  miss++;              }          }          System.out.println("Page Faults: " + miss);          System.out.println("Page Hits: " + (entry -miss));          input.close();      }  } |
| **Output:** |
| 1. **Least Frequently Used**   public class LFU {      public static int pageFaults(int n, int c, int[] pages) {          int count = 0;          List<Integer> v = new ArrayList<>();          Map<Integer, Integer> mp = new HashMap<>();          int i;          for (i = 0; i <= n - 1; i++) {              int index = v.indexOf(pages[i]);              if (index == -1) {                  if (v.size() == c) {                      mp.put(v.get(0), mp.get(v.get(0))-1);                      v.remove(0);                  }                  v.add(pages[i]);                  mp.put(pages[i], mp.getOrDefault(pages[i], 0)+1);                  count++;              } else {                  mp.put(pages[i], mp.get(pages[i])+1);                  v.remove(index);                  v.add(pages[i]);              }              int k = v.size() - 2;              while (k > -1 && mp.get(v.get(k)) > mp.get(v.get(k + 1))) {                  Collections.swap(v, k, k+1);                  k--;              }          }          return count;      }      public static void main(String[] args) {          Scanner input = new Scanner(System.in);          // Inputs          System.out.print("Enter Page frame: ");          int c = input.nextInt();          System.out.print("Enter number of entries in queue: ");          int n = input.nextInt();            int[] pages = new int[n];          for (int i = 0; i < n; i++) {              System.out.print("Enter value of entry " + (i + 1) + ": ");              pages[i] = input.nextInt();          }          System.out.println("Page Faults = " + pageFaults(n, c, pages));          System.out.println("Page Hits = " + (n - pageFaults(n, c, pages)));          input.close();      }  } |
| **Output:** |