1.LRU

INPUT:

```
import java.util.*;
public class Lru {
    static Scanner scanner = new Scanner(System.in);
    public void LruImplementation(int pages[], int capacity) {
        int pageFaults = 0;
        HashMap<Integer, Integer> map = new HashMap();
        HashSet<Integer> currentSet = new HashSet();
        for (int i = 0; i < pages.length; i++) {
            if (currentSet.size() < capacity) {
                if (!currentSet.contains(pages[i])) {
                    currentSet.add(pages[i]);
                    pageFaults++;
                }
                map.put(pages[i], i);
            } else {
                if (!currentSet.contains(pages[i])) {
                    Iterator<Integer> it = currentSet.iterator();
                    int lru = Integer.MAX_VALUE;
                    int val = 0;
                    while (it.hasNext()) {
                        int temp = it.next();
                        if (map.get(temp) < lru) {
                            lru = map.get(temp);
                            val = temp;
                        }
                    }
                    currentSet.remove(val);
                    map.remove(val);
```

```
                        currentSet.add(pages[i]);
                        pageFaults++;
                    }
                    map.put(pages[i], i);
                }
            }
            System.out.println("Page Faults: " + pageFaults);
            int pageHits = pages.length - pageFaults;
            System.out.println("Page Hits: " + pageHits);
            System.out.println("Hit Ratio: " + pageHits + "/" + pages.length + " = " + (double)
pageHits / pages.length);
        }
        public static void main(String[] args) {
            int capacity, n, pages[];
            // int pages[] = {1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6};
            Lru lru = new Lru();
            System.out.print("Enter capacity of page frame: ");
            capacity = scanner.nextInt();
            System.out.print("Enter number of page sequence: ");
            n = scanner.nextInt();
            pages = new int[n];
            System.out.print("Enter values (space separated): ");
            for (int i = 0; i < n; i++) {
                pages[i] = scanner.nextInt();
            }
            lru.LruImplementation(pages, capacity);
        }
}
```

OUTPUT:

```
Enter capacity of page frame: 4
Enter number of page sequence: 20
Enter values (space separated): 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
Page Faults: 10
Page Hits: 10
Hit Ratio: 10/20 = 0.5
```

2. Optimal Page Replacement

INPUT:

```java
import java.util.*;
public class Optimal {
    static Scanner scanner = new Scanner(System.in);
    private int predict(int pages[],HashSet<Integer> currentSet,int index) {
        Iterator<Integer> it = currentSet.iterator();
        int val = -1;
        int farthestIndex = index-1;
        while(it.hasNext()) {
            int temp = it.next();
            int i;
            for(i = index; i < pages.length; i++) {
                if(pages[i] == temp) {
                    if(i > farthestIndex) {
                        farthestIndex = i;
                        val = temp;
                    }
                    break;
                }
            }
            if(i == pages.length)
            return temp;
        }
        return val;
    }

    public void OptimalImplementation(int pages[], int capacity) {
        int pageFaults = 0;
        HashMap<Integer, Integer> map = new HashMap();
        HashSet<Integer> currentSet = new HashSet();

        for(int i = 0 ; i < pages.length; i++) {

            if(currentSet.size() < capacity) {
                if(!currentSet.contains(pages[i])) {
                    currentSet.add(pages[i]);
                    pageFaults++;
                }
            }
            else {
                if(!currentSet.contains(pages[i])) {
                    int predictedElement = predict(pages,currentSet,i+1);
                    currentSet.remove(predictedElement);
                    currentSet.add(pages[i]);
                    pageFaults++;
                }
            }
        }
        System.out.println("Page Faults: "+pageFaults);
        int pageHits = pages.length - pageFaults;
        System.out.println("Page Hits: "+pageHits);
        System.out.println("Hit Ratio: "+pageHits + "/" + pages.length + " = " +
(double)pageHits/pages.length);
    }
    public static void main(String[] args) {
        int capacity, n, pages[];
        // int pages[] = {1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6};
        Optimal optimal = new Optimal();
        System.out.print("Enter capacity of page frame: ");
        capacity = scanner.nextInt();
        System.out.print("Enter number of page sequence: ");
        n = scanner.nextInt();
        pages = new int[n];
        System.out.print("Enter values (space separated): ");
        for(int i = 0 ; i < n ; i++) {
         pages[i] = scanner.nextInt();
        }
        optimal.OptimalImplementation(pages, capacity);
    }
}
```

OTPUT:

```
Enter capacity of page frame: 4
Enter number of page sequence: 20
Enter values (space separated): 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
Page Faults: 8
Page Hits: 12
Hit Ratio: 12/20 = 0.6
```