import java.util.\*;

public class PageReplacement {

    static Scanner sc = new Scanner(System.in);

    // Method to calculate page faults for FIFO

    public static int fifo(int frames[], int n, int reference[]) {

        Set<Integer> s = new HashSet<>();

        Queue<Integer> q = new LinkedList<>();

        int pageFaults = 0;

        for (int i = 0; i < n; i++) {

            if (!s.contains(reference[i])) {

                if (q.size() == frames.length) {

                    int removed = q.poll();

                    s.remove(removed);

                }

                q.add(reference[i]);

                s.add(reference[i]);

                pageFaults++;

            }

        }

        return pageFaults;

    }

    // Method to calculate page faults for LRU

    public static int lru(int frames[], int n, int reference[]) {

        Set<Integer> s = new HashSet<>();

        Map<Integer, Integer> indexMap = new HashMap<>();

        int pageFaults = 0;

        for (int i = 0; i < n; i++) {

            if (!s.contains(reference[i])) {

                if (s.size() == frames.length) {

                    int lru = Integer.MAX\_VALUE, val = -1;

                    for (Map.Entry<Integer, Integer> entry : indexMap.entrySet()) {

                        if (entry.getValue() < lru) {

                            lru = entry.getValue();

                            val = entry.getKey();

                        }

                    }

                    s.remove(val);

                    indexMap.remove(val);

                }

                s.add(reference[i]);

                pageFaults++;

            }

            indexMap.put(reference[i], i);

        }

        return pageFaults;

    }

    // Method to calculate page faults for Optimal

    public static int optimal(int frames[], int n, int reference[]) {

        Set<Integer> s = new HashSet<>();

        int pageFaults = 0;

        for (int i = 0; i < n; i++) {

            if (!s.contains(reference[i])) {

                if (s.size() == frames.length) {

                    int farthest = -1, val = -1;

                    for (int f : s) {

                        int j;

                        for (j = i + 1; j < n; j++) {

                            if (reference[j] == f) {

                                break;

                            }

                        }

                        if (j == n) {

                            val = f;

                            break;

                        }

                        if (j > farthest) {

                            farthest = j;

                            val = f;

                        }

                    }

                    s.remove(val);

                }

                s.add(reference[i]);

                pageFaults++;

            }

        }

        return pageFaults;

    }

    public static void main(String[] args) {

        System.out.println("Enter number of frames:");

        int numFrames = sc.nextInt();

        System.out.println("Enter number of reference pages:");

        int n = sc.nextInt();

        int reference[] = new int[n];

        System.out.println("Enter the reference string:");

        for (int i = 0; i < n; i++) {

            reference[i] = sc.nextInt();

        }

        int frames[] = new int[numFrames];

        Arrays.fill(frames, -1);

        while (true) {

            System.out.println("\nPage Replacement Algorithms:");

            System.out.println("1. FIFO");

            System.out.println("2. LRU");

            System.out.println("3. Optimal");

            System.out.println("4. Exit");

            System.out.print("Enter your choice: ");

            int choice = sc.nextInt();

            switch (choice) {

                case 1:

                    System.out.println("FIFO Page Faults: " + fifo(frames, n, reference));

                    break;

                case 2:

                    System.out.println("LRU Page Faults: " + lru(frames, n, reference));

                    break;

                case 3:

                    System.out.println("Optimal Page Faults: " + optimal(frames, n, reference));

                    break;

                case 4:

                    System.out.println("Exiting...");

                    System.exit(0);

                    break;

                default:

                    System.out.println("Invalid choice! Please try again.");

            }

        }

    }

}