Activity Selection Problem solution

```
import java.io.*;
import java.lang.*;
import java.util.*;
class ActivitySelection {
     public static void printMaxActivities(int s[], int f[],
                                                int n)
     {
          int i, j,cout = 0;
          System.out.println(
               "Following activities are selected : ");
          i = 0;
          System.out.print(i + " ");

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

               if (s[j] >= f[i]) {
                    System.out.print(j + " ");
                    i = j;
                    cout++;
               }
          }
          System.out.println("\n");
          System.out.println( "Total count : " + (cout+1));
     }
     public static void main(String[] args)
     {
          int s[] = { 1, 3, 0, 5, 8, 5 };
          int f[] = { 2, 4, 6, 7, 9, 9 };
          int n = s.length;

          printMaxActivities(s, f, n);
     }
}
```

String Matching Problem solution

```
public class NaiveSearch {
     static void search(String pat, String txt)
     {
          int l1 = pat.length();
          int l2 = txt.length();
          int i = 0, j = l2 - 1;
          for (i = 0, j = l2 - 1; j < l1;) {
               if (txt.equals(pat.substring(i, j + 1))) {
                    System.out.println("Pattern found at index
"+i);
               }
               i++;
               j++;
          }
     }
     public static void main(String args[])
     {
          String pat = "AABAACAADAABAAABAA";
          String txt = "AABA";
          search(pat, txt);
     }
}
```

HalfMan Coding Problem solution

```java
package com.mycompany.main.java;

import java.util.PriorityQueue;
import java.util.Comparator;

class HuffmanNode {
    int item;
    char c;
    HuffmanNode left;
    HuffmanNode right;
}
class ImplementComparator implements Comparator<HuffmanNode> {
    public int compare(HuffmanNode x, HuffmanNode y) {
        return x.item - y.item;
    }
}

public class Main {
    public static void printCode(HuffmanNode root, String s) {
        if (root.left == null && root.right == null &&
Character.isLetter(root.c)) {
            System.out.println(root.c + " | " + s);
            return;
        }
        printCode(root.left, s + "0");
        printCode(root.right, s + "1");
    }

    public static void main(String[] args) {
        int n = 5;
        char[] charArray = { 'A', 'B', 'C', 'D', 'R' };
        int[] charfreq = { 5, 2, 1, 1, 2 };
```

```
        PriorityQueue<HuffmanNode> q = new
PriorityQueue<HuffmanNode>(n, new ImplementComparator());

        for (int i = 0; i < n; i++) {
            HuffmanNode hn = new HuffmanNode();
            hn.c = charArray[i];
            hn.item = charfreq[i];
            hn.left = null;
            hn.right = null;
            q.add(hn);
        }
        HuffmanNode root = null;
        while (q.size() > 1) {
            HuffmanNode x = q.peek();
            q.poll();

            HuffmanNode y = q.peek();
            q.poll();

            HuffmanNode f = new HuffmanNode();
            f.item = x.item + y.item;
            f.c = '-';
            f.left = x;
            f.right = y;

            root = f;

            q.add(f);
        }
        System.out.println("Char | Code");
        System.out.println("-----+------");
        printCode(root, "");
    }
}
```