Program:11a

```
public class LongestPalindromeSubstring {
  public static String longestPalindromeTab(String s) {
     int n = s.length();
     if (n < 2) return s;
     boolean[][] dp = new boolean[n][n];
     int start = 0, maxLen = 1;
         for (int i = 0; i < n; i++) dp[i][i] = true;
         for (int len = 2; len <= n; len++) {
       for (int i = 0; i + len - 1 < n; i++) {
         int j = i + len - 1;
         if (s.charAt(i) == s.charAt(j)) {
           if (len == 2 || dp[i+1][j-1]) {
             dp[i][j] = true;
             if (len > maxLen) {
               start = i;
               maxLen = len;
             }}
         } else {
           dp[i][j] = false;
         }}
     }
     return s.substring(start, start + maxLen);
  }
}
```

Input:

```
s = "babad"
```

Output:

```
Longest Palindromic Substring: bab
```

Program:11b

```
public class MaxSubarray {
    public static int maxSubArrayKadane(int[] nums) {
        if (nums == null || nums.length == 0) return 0;
        int maxEndingHere = nums[0];
        int maxSoFar = nums[0];
        for (int i = 1; i < nums.length; i++) {
            maxEndingHere = Math.max(nums[i], maxEndingHere + nums[i]);
            maxSoFar = Math.max(maxSoFar, maxEndingHere);
        }
        return maxSoFar;
    }
}
```

Input:

```
nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]
```

Output:

```
Maximum Subarray Sum: 6
```

Program:11c

```
import java.util.*;
public class MinCostTicketsTab {
    public int mincostTickets(int[] days, int[] costs) {
        int n = days.length;
        int[] dp = new int[n + 1]; // dp[i] = min cost to cover days[i..n-1], dp[n] = 0
        for (int i = n - 1; i >= 0; i--) {
            int cost1 = costs[0] + dp[i + 1];
            int j = i;
            while (j < n && days[j] <= days[i] + 6) j++;
            int cost7 = costs[1] + dp[j];
            int k = i;
            while (k < n && days[k] <= days[i] + 29) k++;
            int cost30 = costs[2] + dp[k];
            dp[i] = Math.min(cost1, Math.min(cost7, cost30));
        }
        return dp[0];
    }
}
```

Input:

```
days  = [1, 4, 6, 7, 8, 20]
costs = [2, 7, 15]
```

Output:

```
Minimum Travel Cost: 11
```