Q1) Mancher's Algorithm

```
import java.util.*;

class Solution
{
static void findLongestPalindromicString(String text)
{
int N = text.length();
if (N == 0)
return;
N = 2 * N + 1; // Position count
int[] L = new int[N + 1]; // LPS Length Array
L[0] = 0;
L[1] = 1;
int C = 1; // centerPosition
int R = 2; // centerRightPosition
int i = 0; // currentRightPosition
int iMirror; // currentLeftPosition
int maxLPSLength = 0;
int maxLPSCenterPosition = 0;
int start = -1;
int end = -1;
int diff = -1;
for (i = 2; i < N; i++)
{

iMirror = 2 * C - i;
L[i] = 0;
```

```
diff = R - i;

if (diff > 0)
L[i] = Math.min(L[iMirror], diff);

while (((i + L[i]) + 1 < N && (i - L[i]) > 0) &&
(((i + L[i] + 1) % 2 == 0) ||
(text.charAt((i + L[i] + 1) / 2) ==
text.charAt((i - L[i] - 1) / 2))))
{
L[i]++;
}

if (L[i] > maxLPSLength)
{
maxLPSLength = L[i];
maxLPSCenterPosition = i;
}
if (i + L[i] > R)
{
C = i;
R = i + L[i];
}
}

start = (maxLPSCenterPosition - maxLPSLength) / 2;
end = start + maxLPSLength - 1;
System.out.printf("LPS of string is %s : ", text);
for (i = start; i <= end; i++)
```

```
System.out.print(text.charAt(i));
System.out.println();
}
public static void main(String[] args)
{
String text = "babcbabcbaccba";
findLongestPalindromicString(text);
}
}
```

Q2) Count of all sub strings with weight of character atmost k

```
import java.util.*;

class Solution{

// Function to find the count of
// all the subStrings with weight
// of characters atmost K
static int distinctSubString(String P, String Q,int K, int N)
{

// Hashmap to store all subStrings
HashSet<String> S = new HashSet<String>();

// Iterate over all subStrings
```

```
for (int i = 0; i < N; ++i) {

// Maintain the sum of all characters
// encountered so far
int sum = 0;

// Maintain the subString till the
// current position
String s = "";

for (int j = i; j < N; ++j) {

// Get the position of the
// character in String Q
int pos = P.charAt(j) - 'a';

// Add weight to current sum
sum += Q.charAt(pos) - '0';

// Add current character to subString
s += P.charAt(j);

// If sum of characters is <=K
// then insert into the set
if (sum <= K) {
S.add(s);
}

else {
```

```
break;
}
}
}

return S.size();
}


public static void main(String[] args)
{
String P = "abcde";
String Q = "123456789123456789123456789";
int K = 5;
int N = P.length();

System.out.print(distinctSubString(P, Q, K, N));
}
}
```

Q3) Move Hyphen to the Begining

```
class Solution
```

```
{


static void moveHyphenInFront(char str[])
{
// Traverse from end and swap Hyphens
int i = str.length-1;
for (int j = i; j >= 0; j--)
if (str[j] != '-')
{
char c = str[i];
str[i] = str[j];
str[j] = c;
i--;
}
}

// Driver code
public static void main(String[] args)
{
char str[] = "Hello-World--".toCharArray();
moveHyphenInFront(str);
System.out.println(String.valueOf(str));
}
}
```