Q1) lexicographically first palindromic string

Rearrange the characters of the given string to form a lexicographically first palindromic string.

If no such string exists display message “no palindromic string

Input : malayalam

Output : aalmymlaa

Input : apple

Output : no palindromic string

Approach 1)

1. Sort the string characters in alphabetical(ascending) order.

2. One be one find lexicographically next permutation of the given string.

3. The first permutation which is palindrome is the answer.

Approach 2)

1. If length of string is even, then the frequency of each character in the string must be even.

2. If the length is odd then there should be one character whose frequency is odd and all other chars must have even frequency

and at-least one occurrence of the odd character must be present in the middle of the string.

Solution 1)

1. Store frequency of each character in the given string

2. Check whether a palindromic string can be formed or not using the properties of palindromic string mentioned above.

3. If palindromic string cannot be formed, return “No Palindromic String”.

4. Else we create three strings and then return front_str + odd_str + rear_str.

odd_str : It is empty if there is no character with odd frequency. Else it contains all occurrences of odd character.

front_str : Contains half occurrences of all even occurring characters of string in increasing order.

rear_str Contains half occurrences of all even occurring characters of string in reverse order of front_str.

```
class Solution {

static char MAX_CHAR = 26;

static void countFreq(String str, int freq[], int len)
{
for (int i = 0; i < len; i++)
{
freq[str.charAt(i) - 'a']++;
}
```

```
}

static boolean canMakePalindrome(int freq[], int len)
{
int count_odd = 0;
for (int i = 0; i < MAX_CHAR; i++)
{
if (freq[i] % 2 != 0)
{
count_odd++;
}
}

if (len % 2 == 0)
{
if (count_odd > 0)
{
return false;
}
else
{
return true;
}
}

if (count_odd != 1)
{
return false;
}
```

```
return true;
}

static String findOddAndRemoveItsFreq(int freq[])
{
String odd_str = "";
for (int i = 0; i < MAX_CHAR; i++)
{
if (freq[i] % 2 != 0)
{
freq[i]--;
odd_str = odd_str + (char) (i + 'a');
return odd_str;
}
}
return odd_str;
}

static String findPalindromicString(String str)
{
int len = str.length();
int freq[] = new int[MAX_CHAR];
countFreq(str, freq, len);

if (!canMakePalindrome(freq, len))
{
return "No Palindromic String";
}
```

```
String odd_str = findOddAndRemoveItsFreq(freq);

String front_str = "", rear_str = " ";

for (int i = 0; i < MAX_CHAR; i++)
{
String temp = "";
if (freq[i] != 0)
{
char ch = (char) (i + 'a');

for (int j = 1; j <= freq[i] / 2; j++)
{
temp = temp + ch;
}

front_str = front_str + temp;

rear_str = temp + rear_str;
}
}

return (front_str + odd_str + rear_str);
}

public static void main(String[] args)
{
String str = "malayalam";
```

```
System.out.println(findPalindromicString(str));
}
}
```

My solution

1) Sort the characters from string in accending order

2) Check frequency of each character

3) if it is even

store one in start (storing all in other string)

store other in last

4) if it is odd

if only one character is of odd frequency store in center

else display palindromic string not possible

Q 2) Quick sort

Example

low								high	
Index	0	1	2	3	4	5	6	7	8	9
value	10	16	8	12	15	6	3	9	5	infinity

Low 10

high infinity or max element

pivot index 0 element 10

we have to find the sorted position of pivot element

varible i <- starting from pivot search for element greater than pivot

j <- end of array or last element search for element lower than pivot

i will incremented till we find the element greater than pivot

j will decrement till we find the element lower than pivot

as soon as i an j will find respective element swap elements pointed by i and j

continue the above steps till i < j

Result 1)	10	5	8	12	15	6	3	9	16	infinity
Result 2)	10	5	8	9	15	6	3	12	16	infinity
Result 3)	10	5	8	9	3	6	15	12	16	infinity

now i < j condition failed

so swap index pointed by j is position of pivot now pivot is sorted

Result 4)	6	5	8	9	3	10	15	12	16	infinity

after sorting pivot perform same algorithm on left side and right side of pivot value

partition (low , high)

{

```
pivot     =           arr[ low ];
i = low;
j = high;

while ( i < j )
{
do
{
i++;
}while( arr[ i ] <= pivot );

do
{
j--;
}while( arr[ i ] > pivot );

if ( i < j )
swap ( arr[ i ] , arr[ j ] );
}
swap( arr[ low ], arr [ j ] );
return j;
}

Quick_sort( low , high )
{
if ( low < high )
{
j = partition ( low , high );
Quick_sort( low, j );
```

```
Quick_sort( j+1 , h );
}
}
```

Q4) Selection sort

Example

Index	0	1	2	3	4	5
value	7	4	10	8	3	1

n = 6

1) Start the travelsal of the array from the first i.e. index 0

2) find the lowest element in array from index 1 to n-1 check that lowest number is lower than element at 0 index as well

3) if yes swap that number with o index element

4) increment the count of varialbe pass

Result 1) 1 4 10 8 3 7

Sorted array = 1

Non sorted array = 4 10 8 3 7

Result 2) 1 3 10 8 4 7

Sorted array = 1 3

Non sorted array = 10 8 4 7

Result 3) 1 3 4 8 10 7

Sorted array = 1 3 4

Non sorted array = 8 10 7

Result 4) 1 3 4 7 10 8

Sorted array = 1 3 4 7

Non sorted array = 10 8

Result 5) 1 3 4 7 8 10

Sorted array = 1 3 4 7 8

Non sorted array = 10

We will stop process now as all elements are sorted

if only one element is present in non sorted that means sorting of all elements is done

```
// Java program for implementation of Selection Sort
class SelectionSort
{
void sort(int arr[])
{
int n = arr.length;

// One by one move boundary of unsorted subarray
```

```
for (int i = 0; i < n-1; i++)
{
// Find the minimum element in unsorted array
int min_idx = i;
for (int j = i+1; j < n; j++)
if (arr[j] < arr[min_idx])
min_idx = j;

// Swap the found minimum element with the first
// element
int temp = arr[min_idx];
arr[min_idx] = arr[i];
arr[i] = temp;
}
}

// Prints the array
void printArray(int arr[])
{
int n = arr.length;
for (int i=0; i<n; ++i)
System.out.print(arr[i]+" ");
System.out.println();
}

// Driver code to test above
public static void main(String args[])
{
SelectionSort ob = new SelectionSort();
```

```
int arr[] = { 7,     4,          10,        8,          3,          1, };
ob.sort(arr);
System.out.println("Sorted array");
ob.printArray(arr);
}
}
```