the strings in non-decreasing order of their lengths. If two strings have the same length, then the lexicographically smaller string should appear first.

Input Format

You just need to complete the function `string\_sort` and implement the four string comparison functions.

Constraints

- $1 \leq$ No. of Strings ≤ 50
- $1 \leq$ Total Length of all the strings ≤ 2500
- You have to write your own sorting function and you cannot use the inbuilt ***qsort*** function
- The strings consists of lower-case English Alphabets only.

Output Format

The locked code-stub will check the logic of your code. The output consists of the strings sorted according to the four comparsion functions in the order mentioned in the problem statement.

Sample Input 0

```
4
wkue
qoi
```


Change Theme Language: C

```
59
60      char** arr;
61      arr = (char**)malloc(n * sizeof(char*));
62
63      for(int i = 0; i < n; i++){
64          *(arr + i) = malloc(1024 * sizeof(char));
65          scanf("%s", *(arr + i));
66          *(arr + i) = realloc(*(arr + i), strlen(*(
67      }
68
69      string_sort(arr, n, lexicographic_sort);
70      for(int i = 0; i < n; i++)
71          printf("%s\n", arr[i]);
72      printf("\n");
73
74      string_sort(arr, n, lexicographic_sort_reverse
75      for(int i = 0; i < n; i++)
76          printf("%s\n", arr[i]);
77      printf("\n");
78
79      string_sort(arr, n, sort_by_length);
80      for(int i = 0; i < n; i++)
81          printf("%s\n", arr[i]);
82      printf("\n");
83
84      string_sort(arr, n, sort_by_number_of_distinct
85      for(int i = 0; i < n; i++)
86          printf("%s\n", arr[i]);
87      printf("\n");
88  }
```