Code, Compile & Run

Ide

Contest Code/Name (e.g. JULY15/PRACTICE)

Problem Code/Name (e.g. TEST)

Select

C (gcc 6.3)

Code gets autosaved every second

```
#include <stdio.h>
 #include <stdlib.h>
 #include <string.h>
 void sort_string(char*);
 int main()
{ char string[100];
 printf("Enter some text\n");
 gets(string);
sort_string(string);
 printf("%s\n", string);
 return 0;
 }
 void sort_string(char *s)
 {
 int c, d = 0, length;
 char *pointer, *result, ch;
 length = strlen(s);
 result = (char*)malloc(length+1);
 pointer = s;
 for ( ch = 'a' ; ch <= 'z' ; ch++ )
 {
for ( c = 0 ; c < length ; c++ )
{
if ( *pointer == ch )
{
 *(result+d) = *pointer; d++;
 }
```

0:0

Open File

Custom Input

Run

Custom Input

pink

Status Successfully executed **Date** 2020-06-21 09:28:46 **Time** 0 sec **Mem** 9.424 kB

Input

pink

Output

Enter some text
iknp

Code, Compile & Run

Ide ✖ +

Contest Code/Name (e.g. JULY15/PRACTICE) | Problem Code/Name (e.g. TEST) | Select

C (gcc 6.3) | Code gets autosaved every second

```
14   printf("%s\n", string);
15   return 0;
16   }
17   void sort_string(char *s)
18   {
19   int c, d = 0, length;
20   char *pointer, *result, ch;
21   length = strlen(s);
22   result = (char*)malloc(length+1);
23   pointer = s;
24   for ( ch = 'a' ; ch <= 'z' ; ch++ )
25   {
26  for ( c = 0 ; c < length ; c++ )
27  {
28  if ( *pointer == ch )
29  {
30   *(result+d) = *pointer; d++;
31   }
32  pointer++;
33   }
34   pointer = s;
35  }
36   *(result+d) = '\0';
37   strcpy(s, result);
38   free(result);
39  }
40
41
```

0:0

Open File | ✓Custom Input | Run

Custom Input

```
pink
```

Status Successfully executed **Date** 2020-06-21 09:28:46 **Time** 0 sec **Mem** 9.424 kB ✖

Input

```
pink
```

Output

```
Enter some text
iknp
```

Priyanka shet
4AL19CS070

(i) Algorithm: (sort strings in alphabetic order)

step1 : start

step2 : create two 2D character array of some fixed capacity

step 3 : Take the size of the array as input from the user, & fill the array with names taking them as input.

step 4 : Now to sort this array, first make a nested loop with i and j as iterators respectively.

step 5 : The outer will run from 0 to size - 1, extracting each name of position i, one by one

step 6 : The inner loop will run from i+1 to size-1, comparing the name extracted by outer loop to all the names below it.

step 7 : At each comparision, if the name above is alphabetically greater than the name below, then these two names are interchanged.

step 8 ; After executing the nested loop code section we will obtain an array of names in proper alphabetical order

step 9 : stop.

(ii) Flowchart :