Stage-3-Report-Merge-Sort-Recursive

Vatsal Jingar - 2020CS50449

February 3, 2022

1 Recursive Merge Sort - Algorithm

Used Legacy for this stage
Considered A < a and for insensitive considering A = a
I have implement Merge Sort using two stacks one is temprorary stack and one is main stack. One stack handles the stack and the temporary stack generate output.
The main stack stores the length, initial index and then final index.
    length,initial index, final index

Now if the length is greater than 1 we will recurse.
I will try to explain this using an example. Lets take the example of sorting [a,d,b,c]. To sort this I will store [4,0,4] the length, initial index and final index. Now after this, check whether the length is greater than 1 if yes then recurse. For recursing store length/2, initial index which will be equal to previous initial index, and final index will be initial index + new size. So, the main stack will have [4,0,4,2,0,2], Now again check this whether the length is greater than 1. Again recurse, So Store len by 2, initial index which will be equal to previous initial index and for final index store initial index + new size. So we store [4,0,4,2,0,2,1,0,1]. Now again check whether the size is greater than 1 or not. If not then push the list between 0 and 1 index. which [a]. So the temp Stack is [[a]]. And after storing in temp stack shift the indexes. The shift is that initial <= final index and final index <= previous final index.
So after shifting the indexes, we get the main stack as [4,0,4,2,0,2,1,1,2], Also we will update the size accor to final - index.
Now again check whether the new size is greater than 1 or equal to one. So as it is equal to one we will push the list between 1 and 2 that is [d] to the temp stack. Now again shift the indexes, as now the size after being update will be zero so we will pop 3 upper elements current size, current initial index, and current final index. So the List will be [4,0,4,2,0,2]. Now whenever we something from the main List, we will also pop two elements from temp stack, merge them and push it back, So the temp stack would be [[a,d]]. Now then shift the indexes, So the new indexes and size are accor to : [4,0,4,2,2,4]. Check whether the size is greater than one or not. If greater than recurse otherwise push the list in temp stack. Following this, we will have a termination step where if previous final is equal to null then it will have to pop the list from temp stack and that will be the final value. Also in each recursion we will store the linkers to go on the previous function call.
This is the algorithm I have used for implementing recursive merge sort.

2 Files

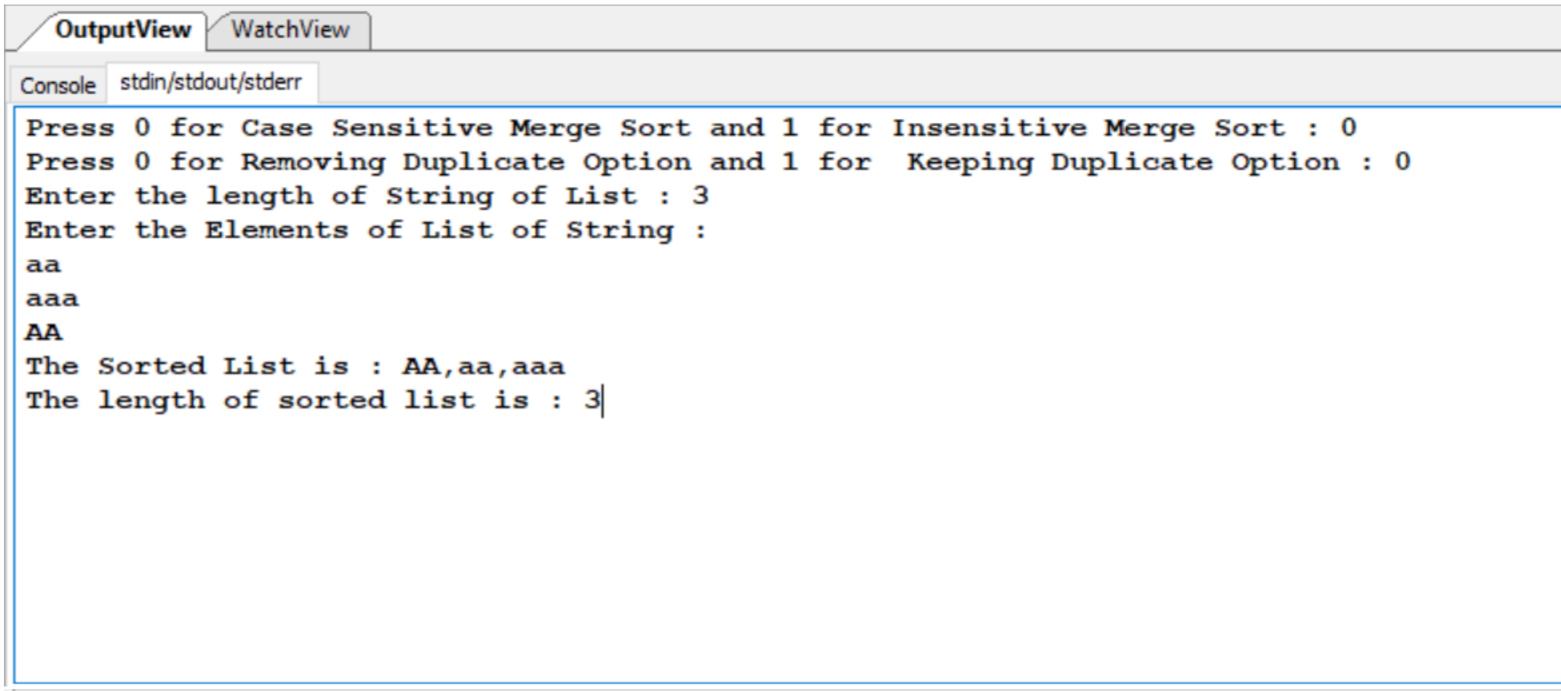
There are five files:
1. input.s
2. mergesort.s
3. merge.s
4. compare.s
5. output.s

Input.s : This file will take the input from user and store the addresses in the list.
mergesort.s : This file will take use the sorting algorithm and use the helper files merge and indeed compare.s which combinely return the sorted list.
merge.s and compare.s: Are similar to Stage-2 and Stage-1 files with some changes. They are helper files.
output.s : Prints the output on stdout.

The input file stores the input in r10 in the form of a list :
    Case,Duplicate,length,List-Addresses

The input stored in r10 is passed to mergesort file which uses s13 for main stack and r9 for temp stack. It passes the result in r2. This output is send to output.s for printing.

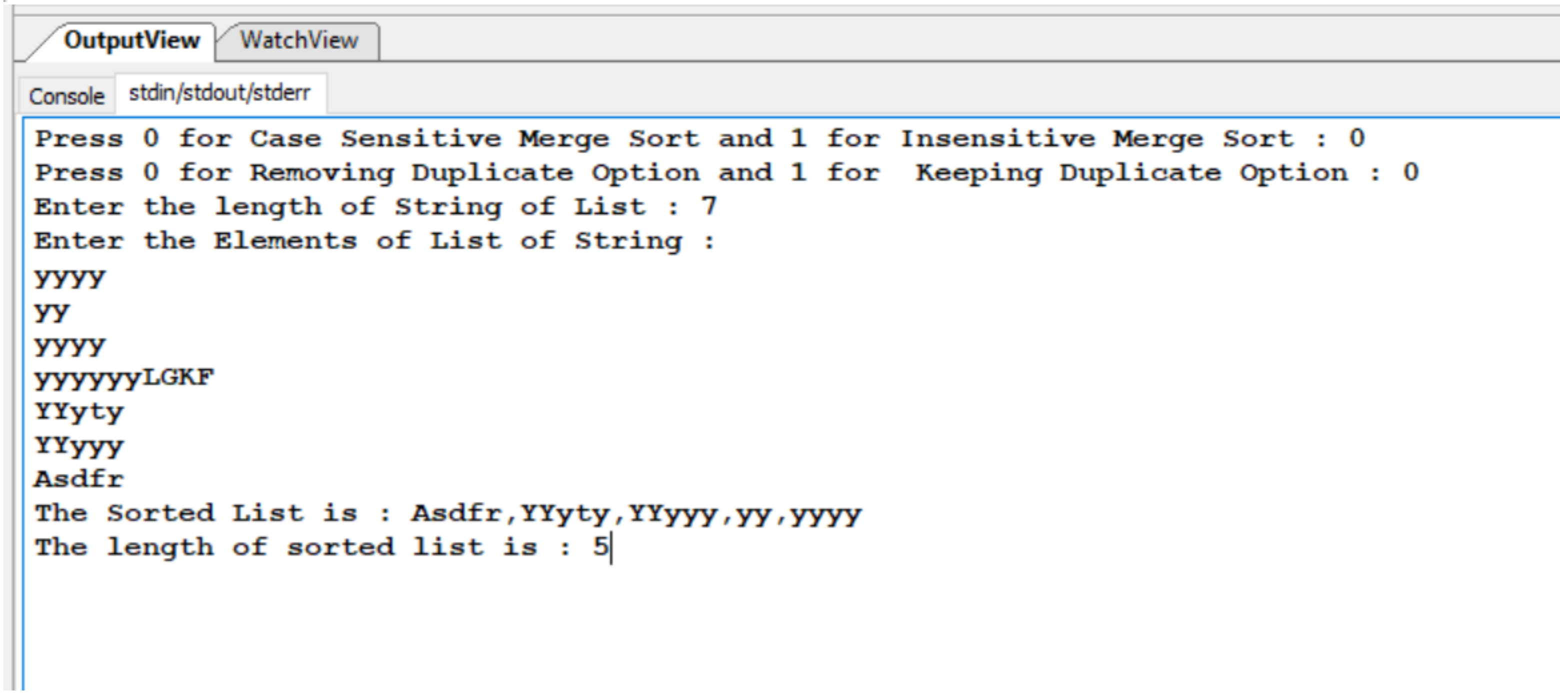
3 Test-Cases



OutputView WatchView

Console stdin/stdout/stderr

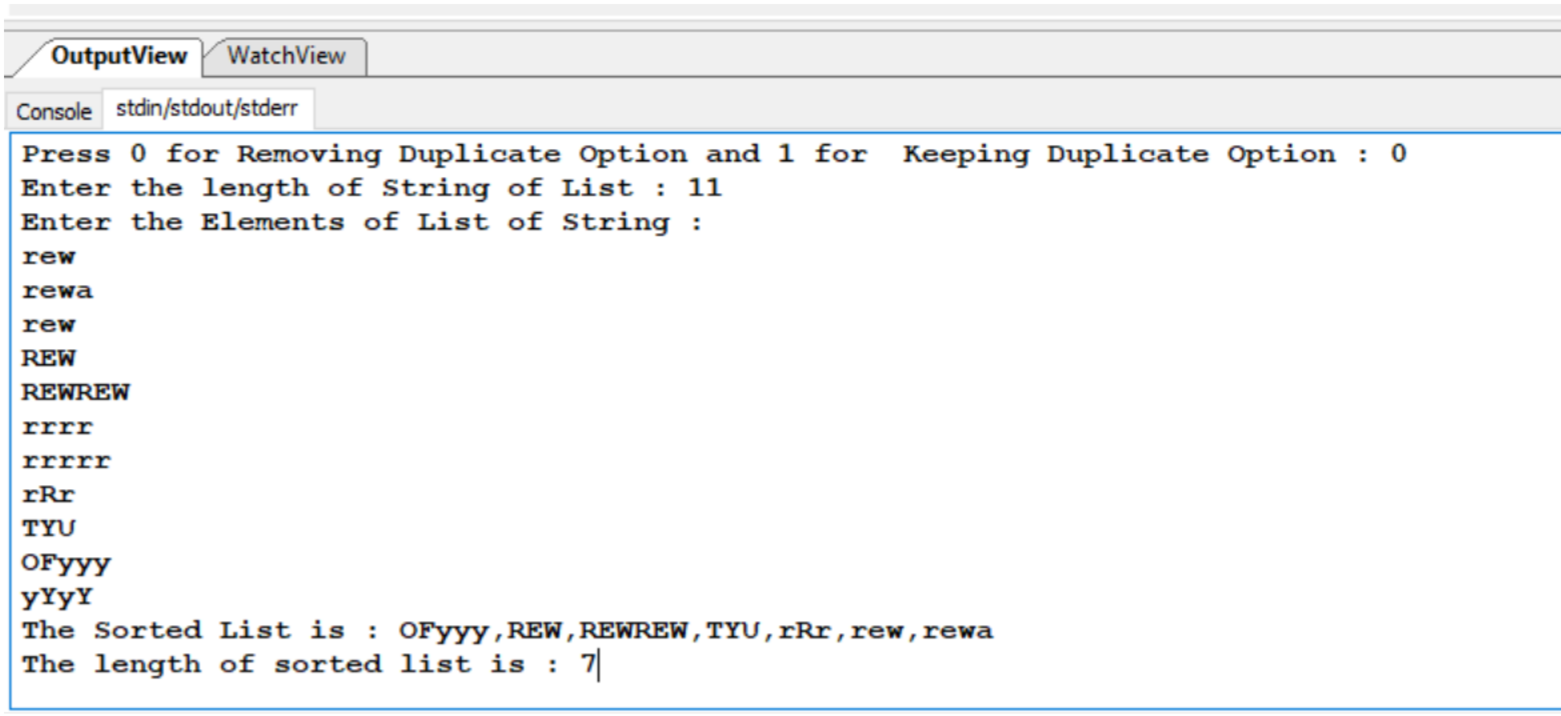
```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 0
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 0
Enter the length of String of List : 3
Enter the Elements of List of String :
aa
aaa
AA
The Sorted List is : AA,aa,aaa
The length of sorted list is : 3
```



OutputView WatchView

Console stdin/stdout/stderr

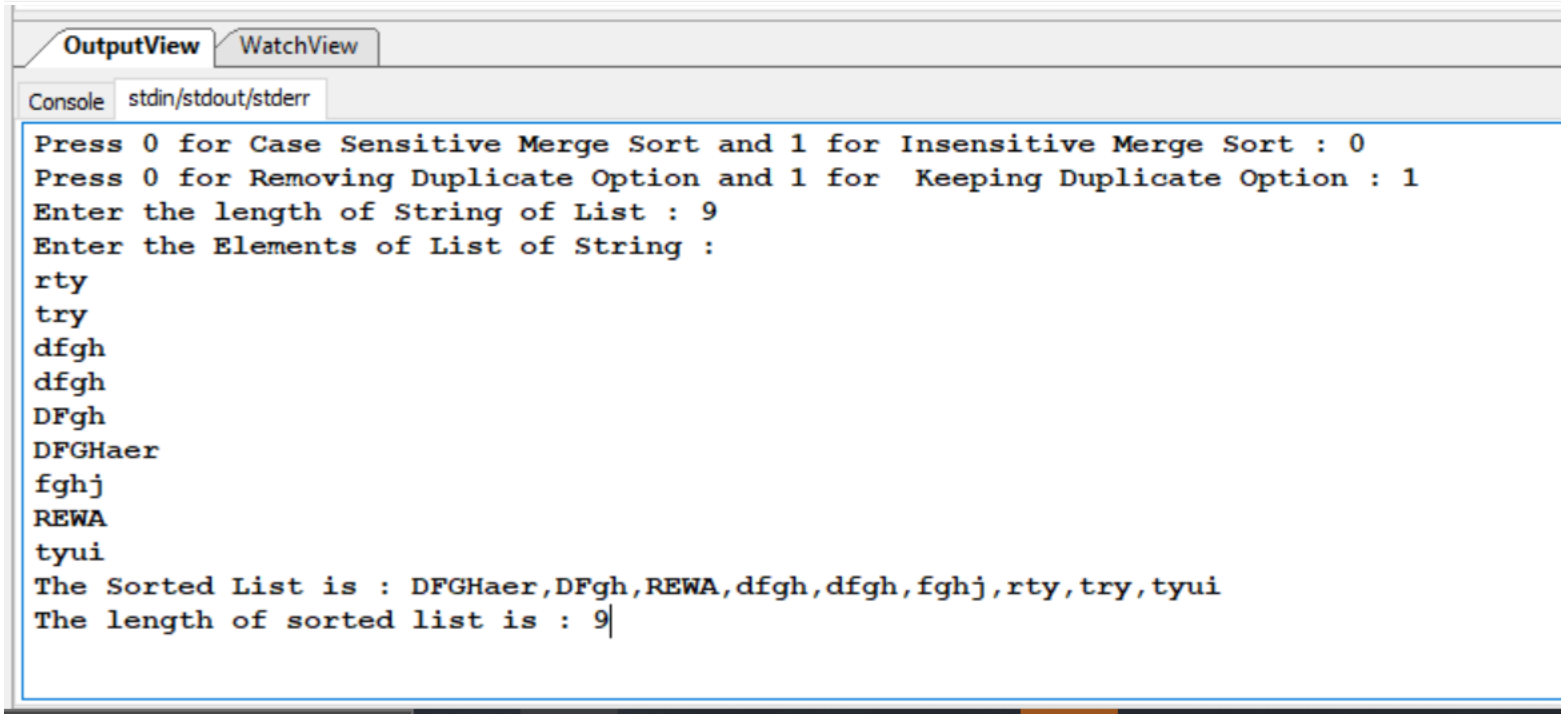
```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 0
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 0
Enter the length of String of List : 7
Enter the Elements of List of String :
yyyy
yy
yyyy
yyyyyyLGKF
YYyty
YYyyy
Asdfr
The Sorted List is : Asdfr,YYyty,YYyyy,yy,yyyy
The length of sorted list is : 5
```



OutputView WatchView

Console stdin/stdout/stderr

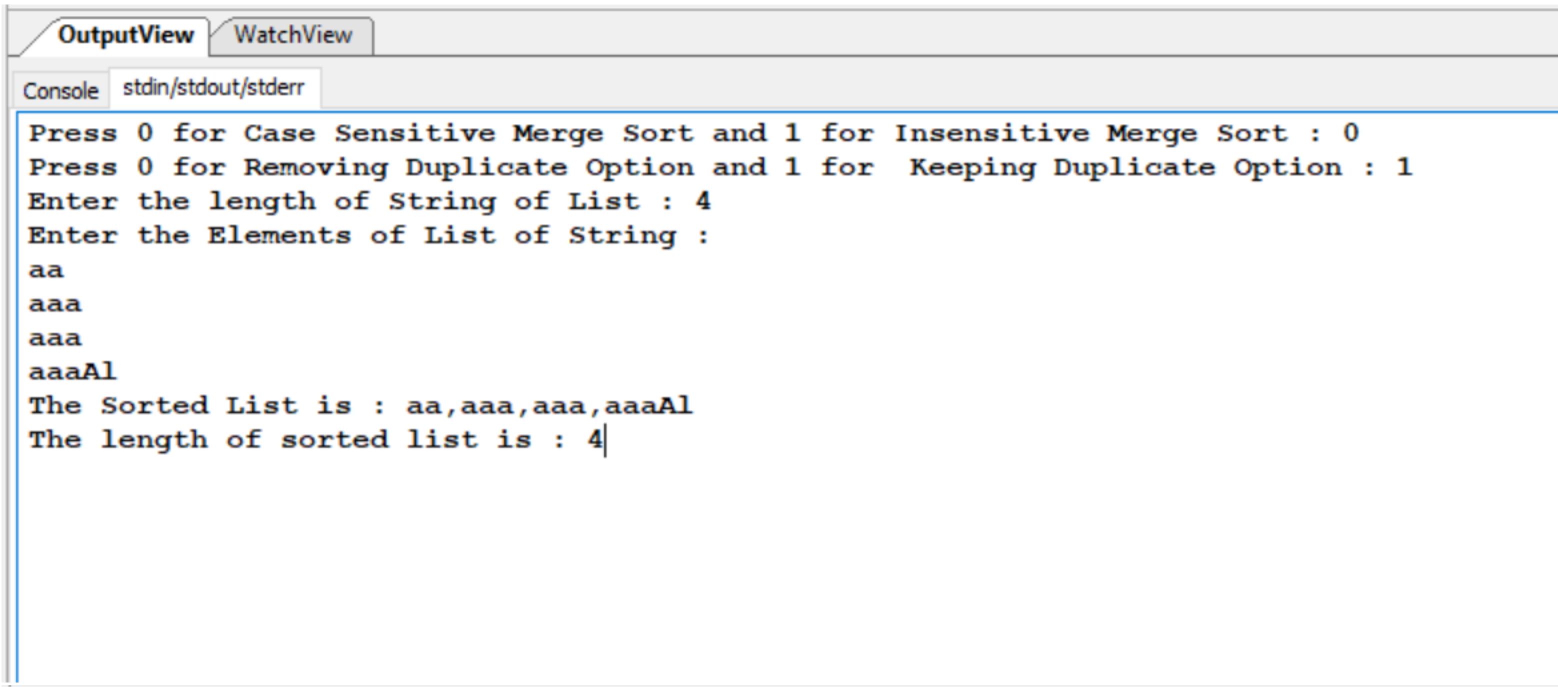
```
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 0
Enter the length of String of List : 11
Enter the Elements of List of String :
rew
rewa
rew
REW
REWREW
rrrr
rrrrr
rRr
TYU
OFyyy
yYyY
The Sorted List is : OFyyy,REW,REWREW,TYU,rRr,rew,rewa
The length of sorted list is : 7
```



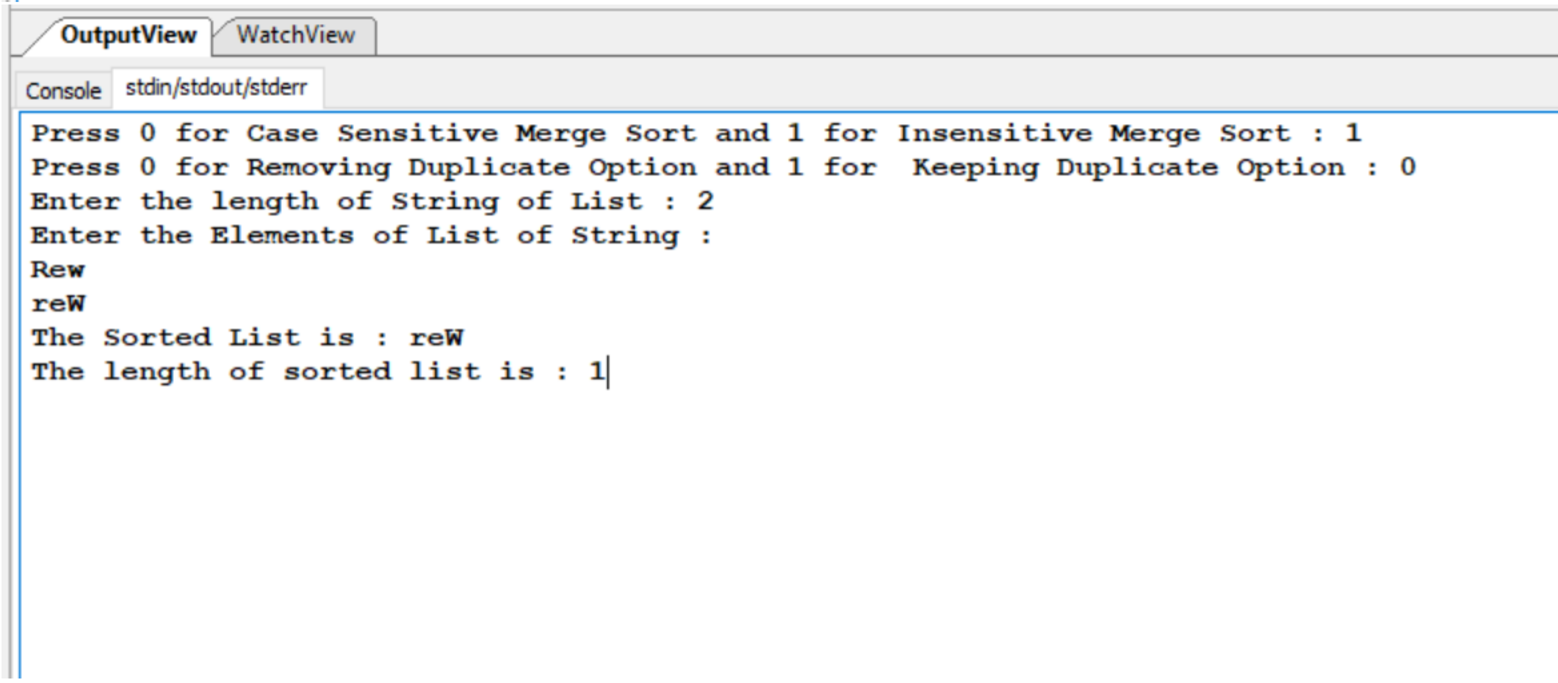
OutputView WatchView

Console stdin/stdout/stderr

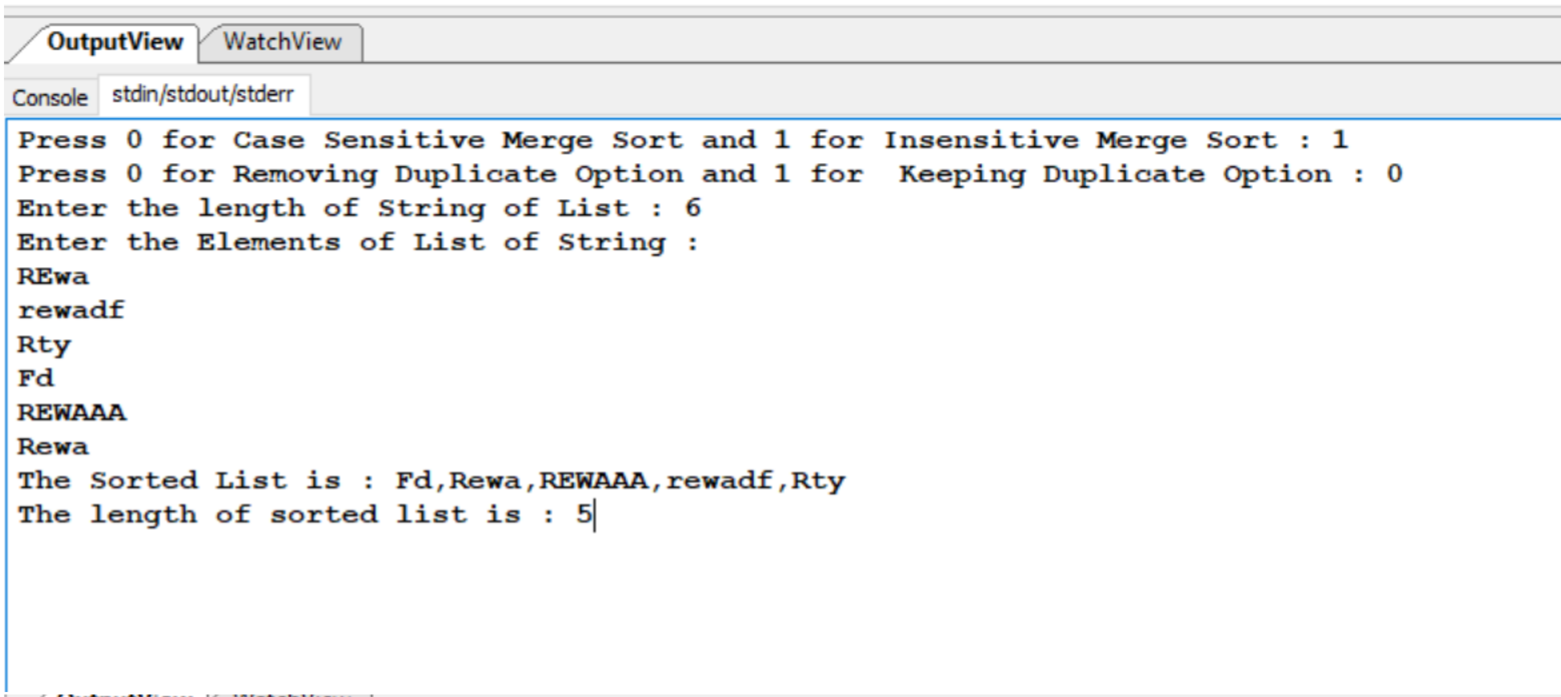
```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 0
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 1
Enter the length of String of List : 9
Enter the Elements of List of String :
rty
try
dfgh
dfgh
DFgh
DFGHaer
fghj
REWA
tyui
The Sorted List is : DFGHaer,DFgh,REWA,dfgh,dfgh,fghj,rty,try,tyui
The length of sorted list is : 9
```



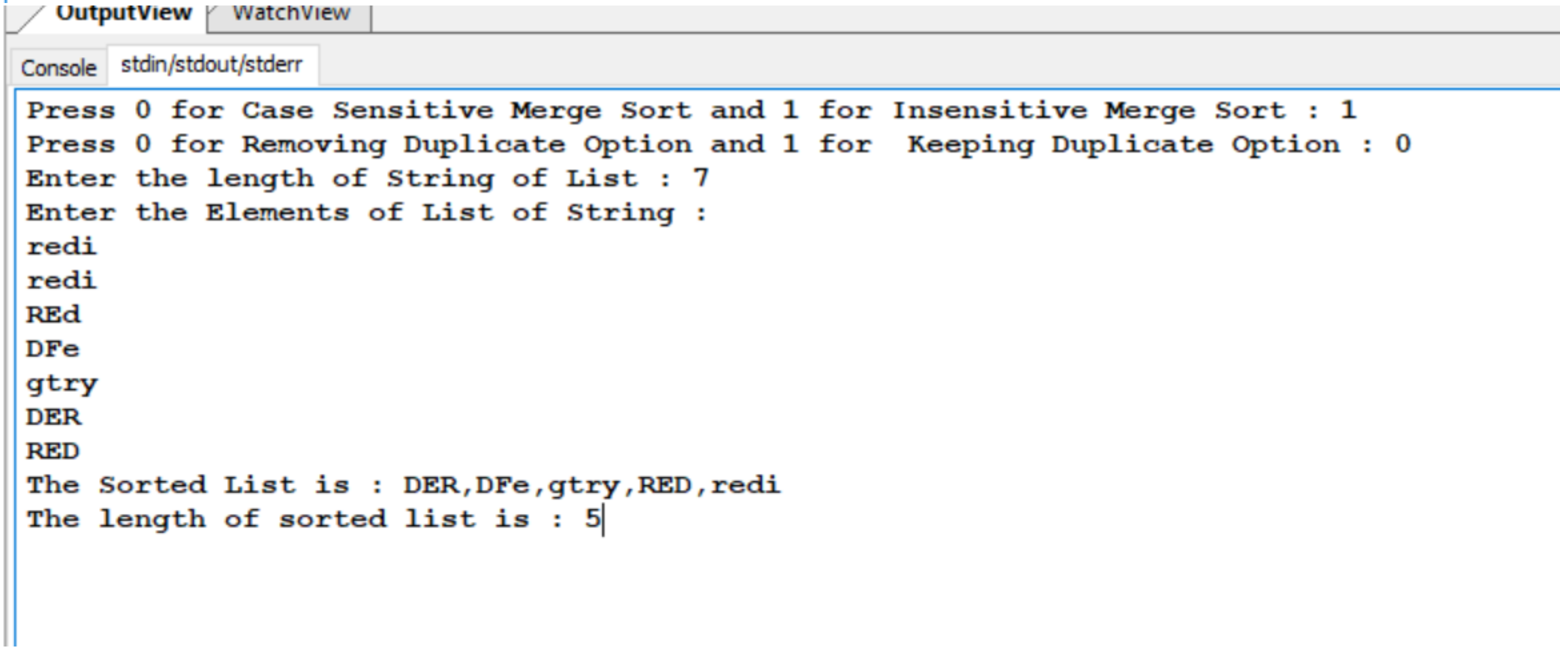
```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 0
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 1
Enter the length of String of List : 4
Enter the Elements of List of String :
aa
aaa
aaa
aaaAl
The Sorted List is : aa,aaa,aaa,aaaAl
The length of sorted list is : 4
```



```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 1
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 0
Enter the length of String of List : 2
Enter the Elements of List of String :
Rew
reW
The Sorted List is : reW
The length of sorted list is : 1
```

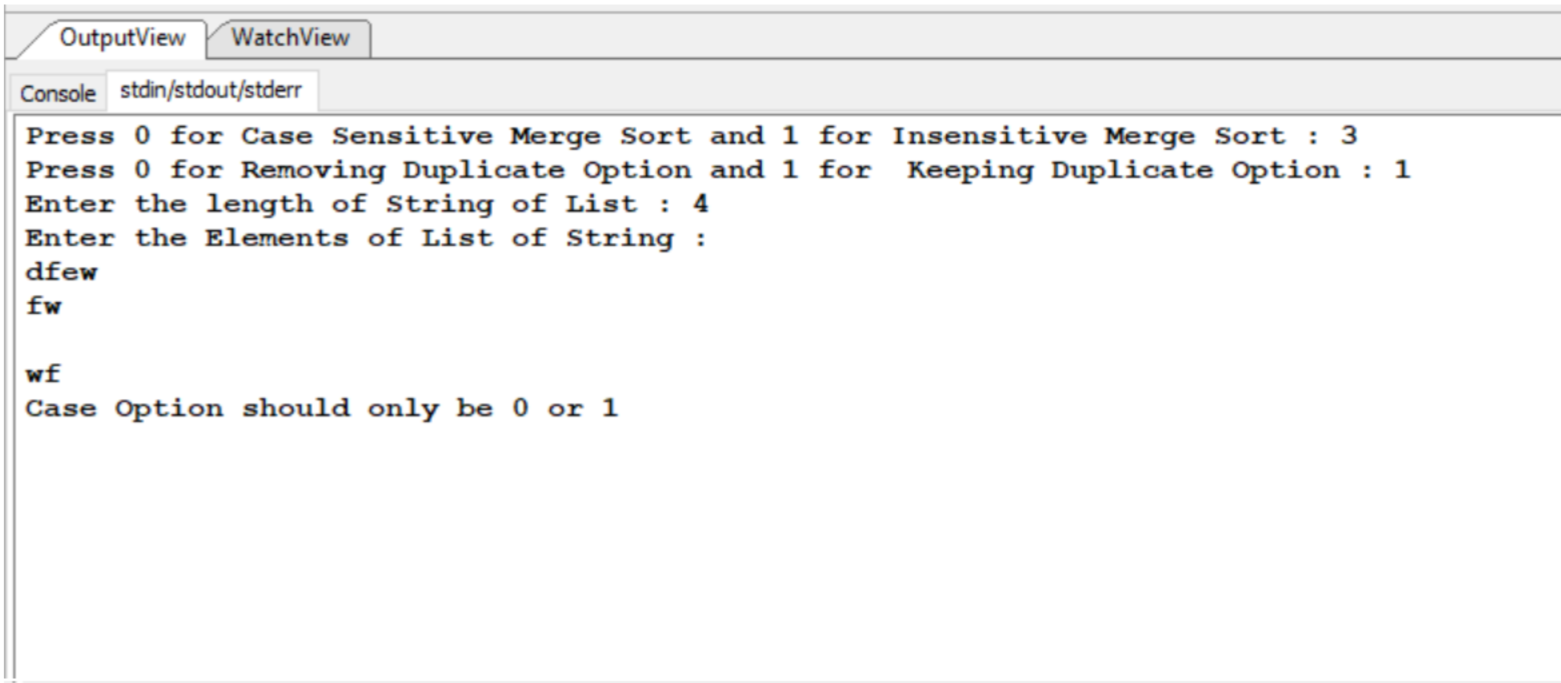


```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 1
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 0
Enter the length of String of List : 6
Enter the Elements of List of String :
REwa
rewadf
Rty
Fd
REWAAA
Rewa
The Sorted List is : Fd,Rewa,REWAAA,rewadf,Rty
The length of sorted list is : 5
```

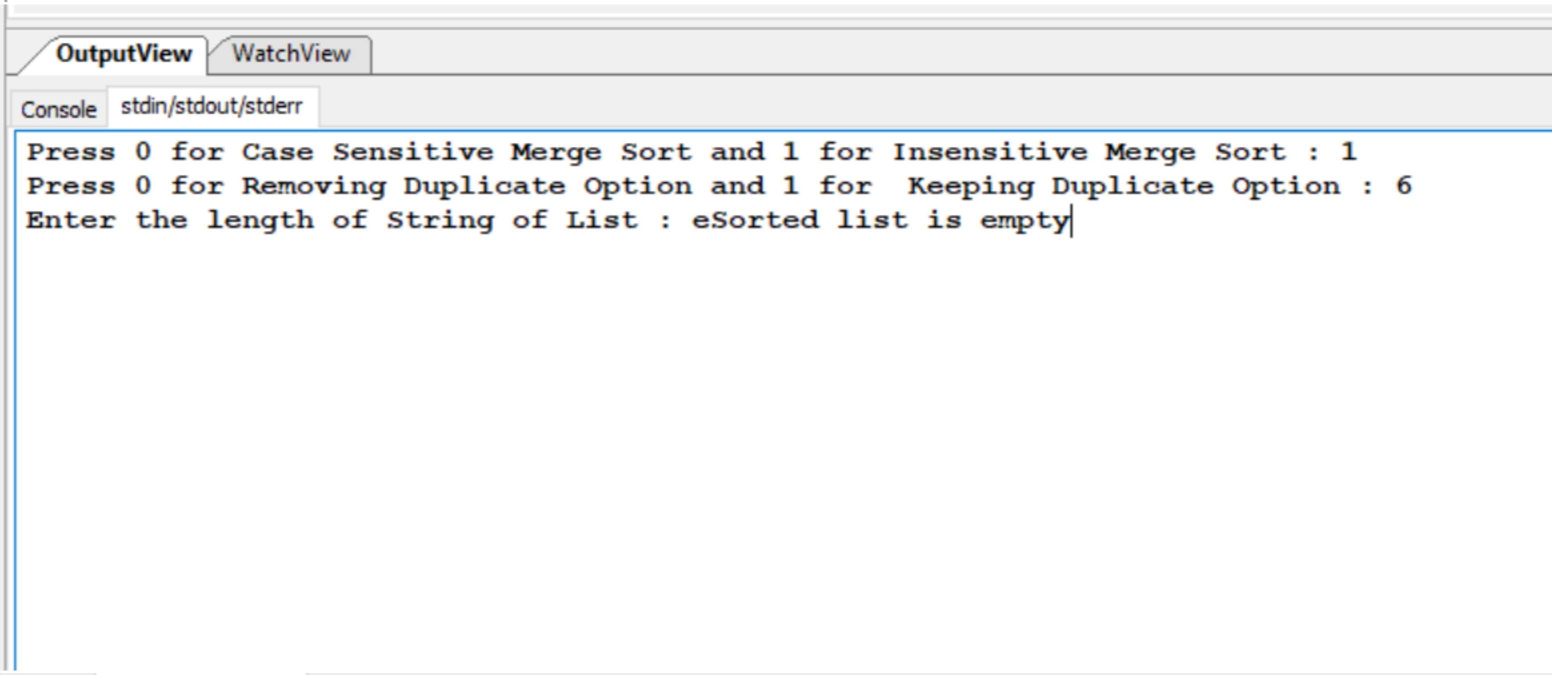


```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 1
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 0
Enter the length of String of List : 7
Enter the Elements of List of String :
redi
redi
REd
DFe
gtry
DER
RED
The Sorted List is : DER,DFe,gtry,RED,redi
The length of sorted list is : 5
```

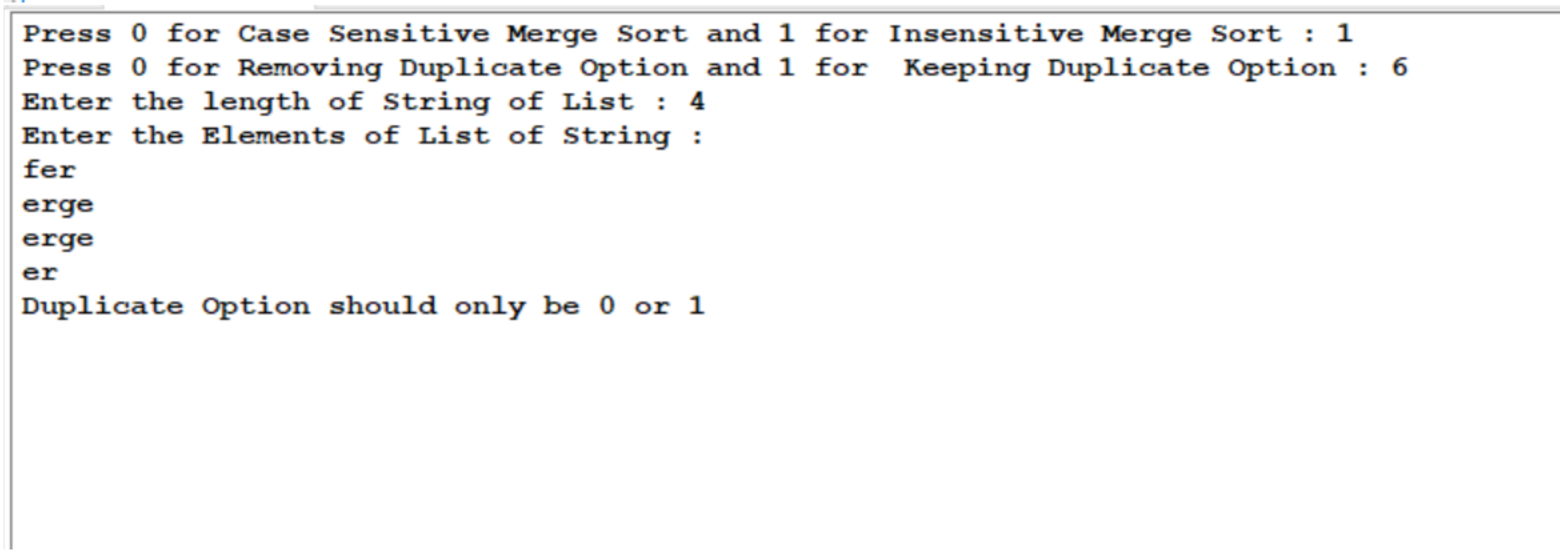
4 Error Handling



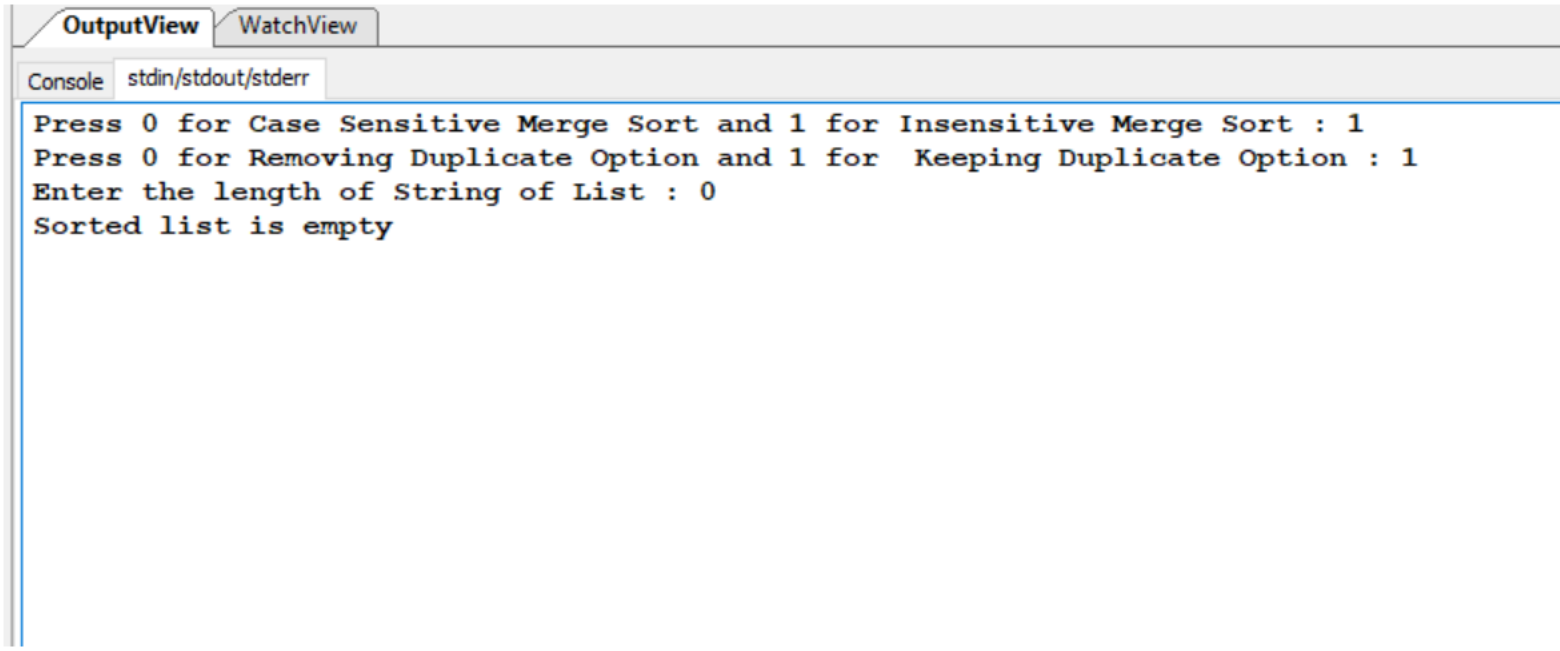
```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 3
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 1
Enter the length of String of List : 4
Enter the Elements of List of String :
dfew
fw

wf
Case Option should only be 0 or 1
```



```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 1
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 6
Enter the length of String of List : eSorted list is empty
```



```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 1
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 6
Enter the length of String of List : 4
Enter the Elements of List of String :
fer
erge
erge
er
Duplicate Option should only be 0 or 1
```



```
Press 0 for Case Sensitive Merge Sort and 1 for Insensitive Merge Sort : 1
Press 0 for Removing Duplicate Option and 1 for  Keeping Duplicate Option : 1
Enter the length of String of List : 0
Sorted list is empty
```