COL216 Lab Assignment-1 (S3)

- Mayank Mangla (2020CS50430)

1 Introduction

This project's aim is to develop a program in ARM Assembly Language for arranging a sequence of character strings in dictionary order (lexicographic order).

Stage 3: This is the final stage of our project which is capable to fulfil our objective. This program takes length of list, duplicate mode, comp mode and input strings list and outputs the length of sorted list and the sorted list's elements.

The report for the stage 3 program is given here:

2 Program Information

The program starts from input.s

1. **Assumptions:** We assume that the input strings are not more than 99 for the list and the user enters only integers for the lengths and mode inputs (the three integers).
2. **Software:** This program has been run on **ARMSim (v2.0.1)** in **Windows OS (v windows 10) machine** and the test cases were chosen at random. Also the **Angel Instructions** were used.

2.1 Input and Output

The program prompts for the input by stdin wherever required
User is expected to input three integers, length of list, comparison mode and 0 or 1 for duplicates. Then the user is asked to enter the strings for the list till the user has entered as many strings as mentioned by the length of list by the user.
The output is in the form of a message followed by the length of the sorted list. Then, a message prompts user for a new line character for each printing string (this is made for beautifying the interface, don't think this as weakness of the program). When all the strings in sorted list are printed then a message is printed stating the end of program.

2.2 Algorithm

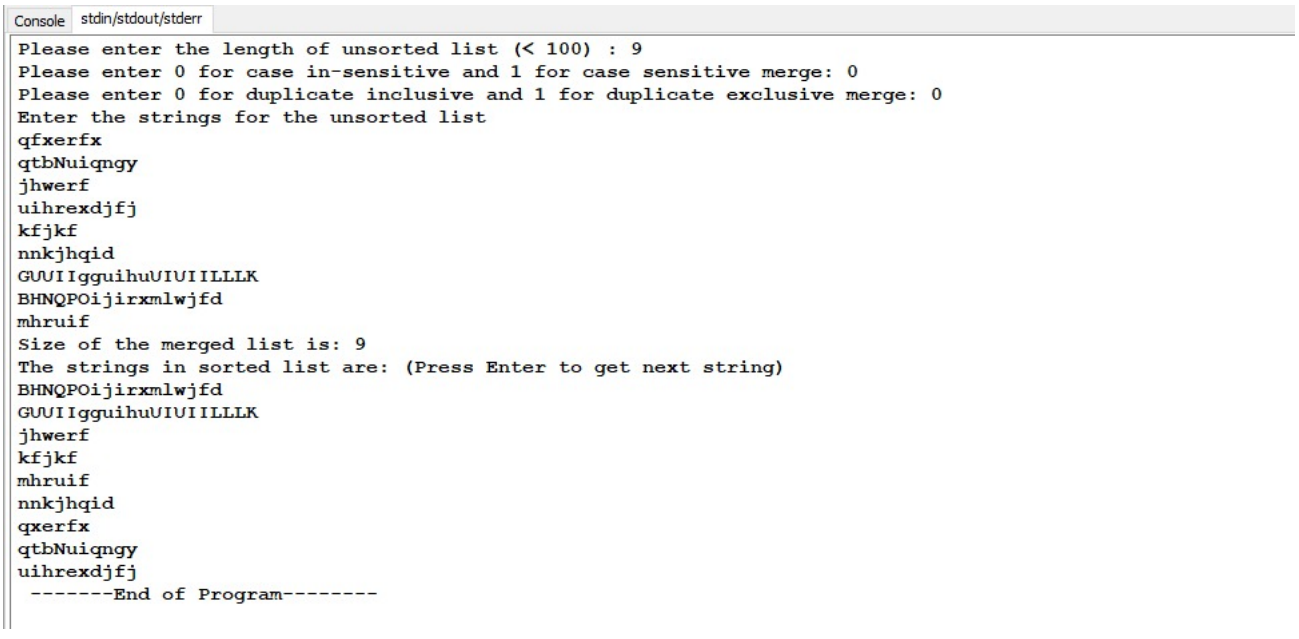
- Input is read character by character. First we read the characters for the length of the lists. A function converts this string into the corresponding integer. Similarly the other integer inputs are taken. These 4 integers are stored in memory.
- If the user enters the length as 0 for both the lists then program shows an error.
- Then we read the input strings character by character. When we get a next line character, to get a null terminated string, we store a null character after the next line character. This signifies the end of a string. (backspace is properly adjusted)
- Then we can generate the list of pointers for which we have already allocated the space in data part of the file.
- In this way we have got 4 integers and 2 lists of addresses from input file and then ***we move to manager.s*** for further computation.
- Manager.s maintains the stack. We push -1 for signifying the mergesort instruction and -1 for merge instruction. Here we initialize the stack by pushing the initial indices and moving to mergesort.
- For mergesort instruction we pop next two elements as right and left and perform the mergesort algorithm. According to which we push a merge instruction and then two mergesort instruction to the stack.
- If the right and left index are same then we need not do anything.

- After this iteration, we go with poping another element from the stack which will tell us the next instruction.
- If the next element is -2, then we go to merge.s.
- Next three elements in the stack are the right, mid and left values required for merging the list.
- In merge we create a new list of pointers with length same as the given list that is to be merged in place. In case of duplicates and nulls, the merged list is appended by null pointers that will be ignored at last.
- This way by traversing down the stack, our recursive algorithm is implemented.
- After we have emptied the stack, we find the length of the sorted list ignoring the null pointers in the last.
- Now we move to ***output.s***.
- Starting of the file prints the message and the length of the sorted list stored in a register.
- This file has a function that converts the integer to a string that could be printed in the console.
- The output file prompts user for a new line character with a message that would tell program that the user wants to see the next string in the sorted list.
- Then by each new line character the program outputs the next string in the sorted list.
- A message stating the end of program is printed.

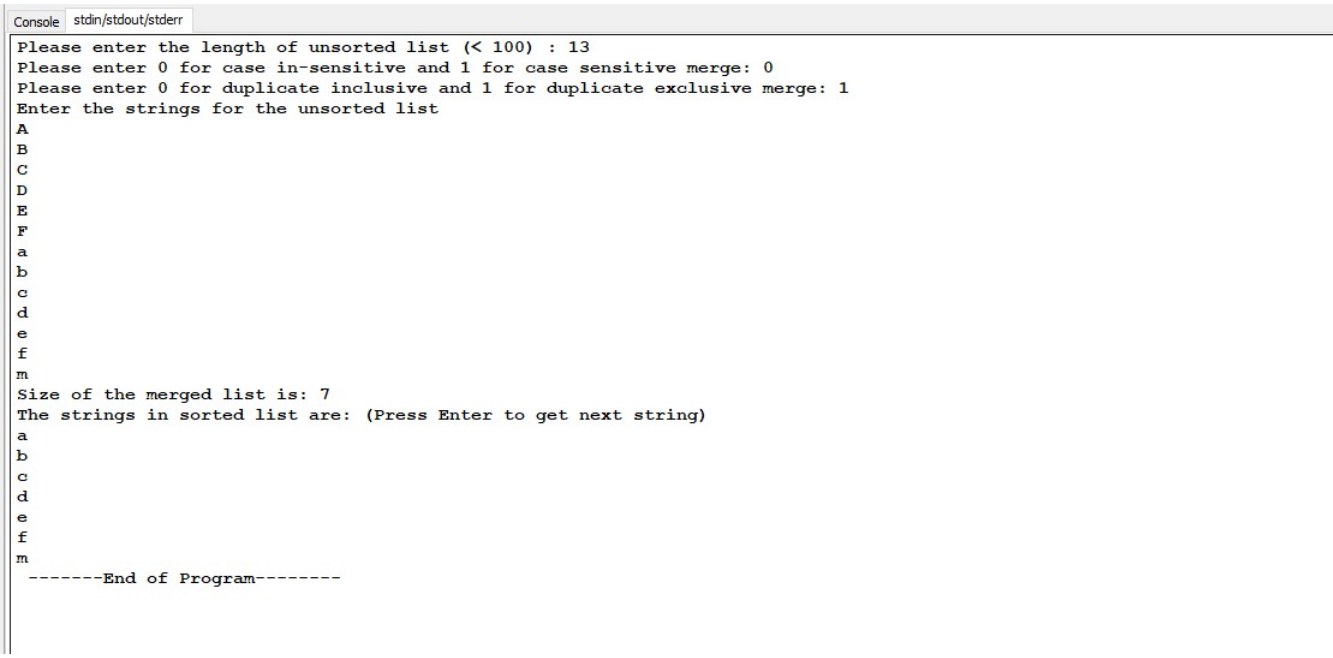
2.3 Test Cases and Results

Some random test cases were tested on the program. The results of which are shown below:

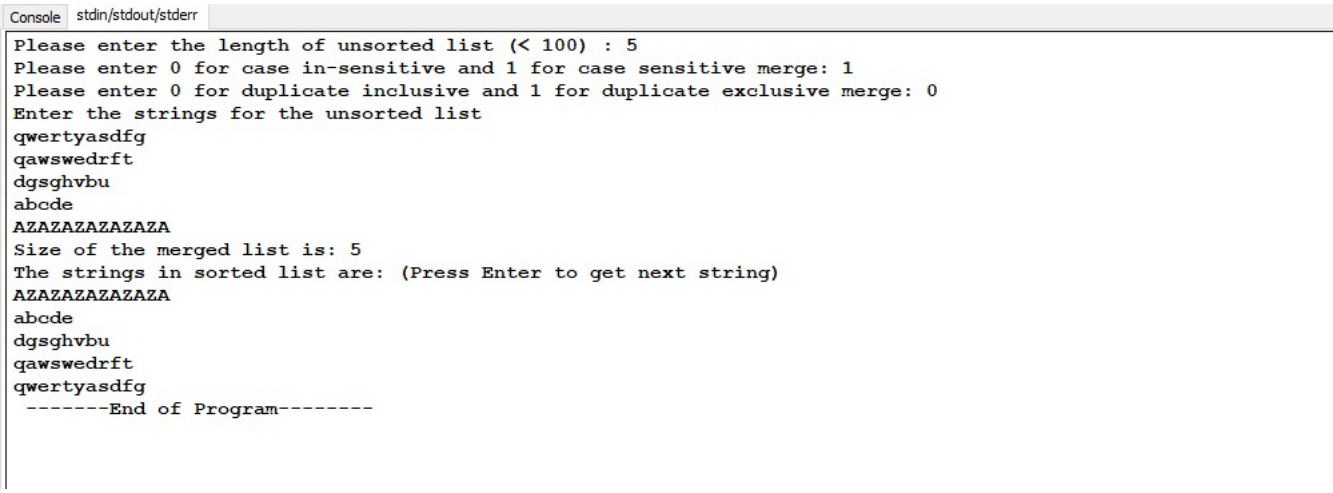
- Case in-sensitive comparison and duplicate inclusive



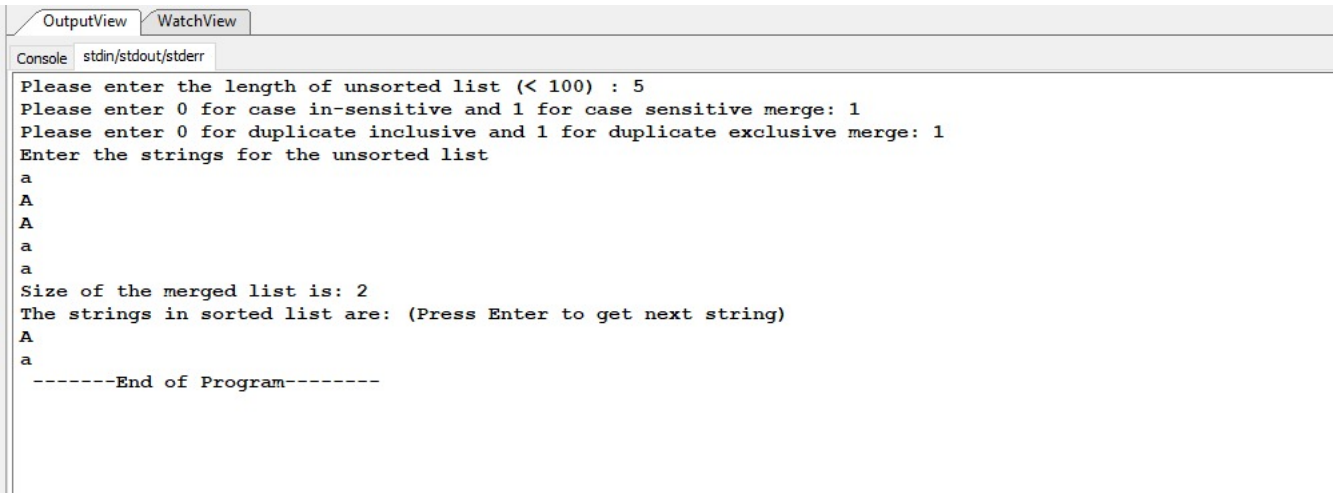
- Case in-sensitive comparison and duplicate exclusive



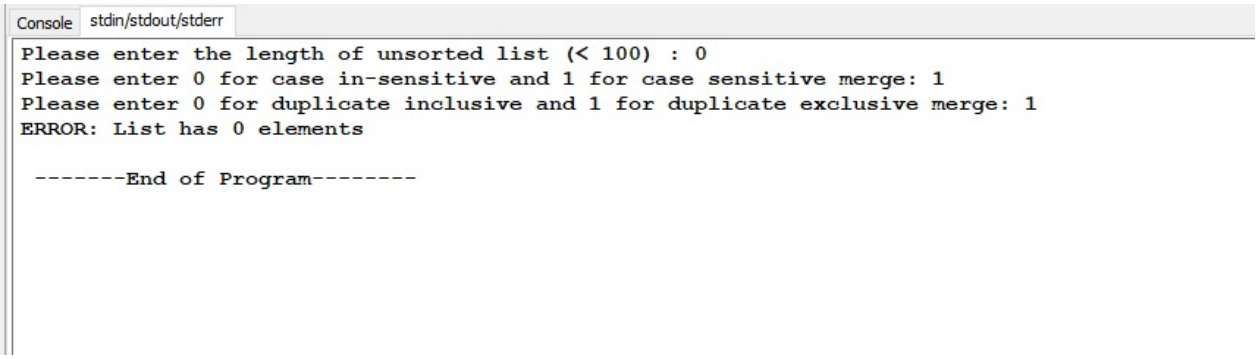
- Case sensitive comparison and duplicate inclusive



- Case sensitive comparison and duplicate exclusive



- Error for zero length list



2.4 Conclusion

After running these random Test cases and getting the desired output we have verified the program's success. We have successfully develop the program in assembly language that implements the recursive mergesort algorithm for sorting a list of character strings according to lexicographic order (dictionary order).