ASSIGNMENT 1 STAGE 3 REPORT

Sreemanti Dey

January 2022

1 Objective

An assembly program that sorts a given list of strings. It supports the two comparison modes - case-insensitive and case-sensitive and gives an option for removing the duplicates.

2 Assumptions

ARMSIM VERSION - 2.0.1
ANGEL SWI

3 Design Decisions

1. I have stored the merged list of pointers on the stack inside the merge function, thus I have done dynamic memory allocation and when I am returning the merged list, I dispose off the allocated space on the stack and I store the merged list in the space allocated to the original input lists, thereby reusing space and saving memory.

2. My code is able to handle corner cases and gives an appropriate error message before exiting the program, details of which are under error handling section.

4 Implementation details

NOTE: main.s is the program file that calls other functions, so it needs to be placed at the top while loading multiple files in ARMSIM.

4.1 Details of mergesort.s file

4.1.1 Algorithm

Base case: If size = 1, then I return the string list pointer immediately. Induction step: If size is greater than 1, then I split the string list into 2 halves and recursively call mergesort on those 2 halves, while ensuring I store the link register on the stack. Then finally I merge these 2 string lists I have got back from the mergesort calls.

4.2 Details of merge.s file

4.2.1 Algorithm

I keep two pointers, one at the starting of the first list and the other one at the starting of second list. Then I check which string is less than the other, accordingly I push the lesser one onto the stack. If duplicate removal option is 0, and I have equal strings, then I push the string from the first list onto the stack and carry on with the algorithm. Else if the duplicate removal option is 1, then I push the string from the first list onto the stack and increment the pointer of the second list so that duplicates are not included in my final list. Also I ensure that the strings themselves do not have duplicates beforehand so that my code does not give wrong outputs. Thus overall my algorithm is time efficient, since the time taken is of the order of sum of sizes of the 2 lists and also space efficient, since I store my merged list of pointers on the stack and dispose off the space.

4.3 Details of UsefulFunctions.s file

1. I have used UsefulFunctions.s for I/O handling, after making few corrections as suggested on piazza.
2. I used the atoi, itoa, strlen, fgets and prints routines from the UsefulFunctions.s file.
3. I have modified the atoi routine in UsefulFunctions.s file, such that now if the string is not an integer, then it prints Invalid Input and exits my program.

4.4 Details of main.s file

1. My main.s file takes in any list of strings and displays the sorted list of strings.
2. I have preserved register values in my main.s file before calling any function, mainly those which are used by callee and lie within r0-r3.

3. Since we have to take in so many inputs, I have assigned a memory space called auxiliaryinfo that stores the size of the unsorted list, comparison mode and duplicate removal option.

4. To conserve as much space as possible, I have assigned a space for my string list and another space for my list of string pointer, each pointer can be assumed to be fit inside 4 bytes, since pointer stores address of my strings. Also, I am storing all the strings contiguously in one space, each taking space equivalent to just the length of the string and I have also given 2 extra bytes for each string to ensure smooth input/output handling.

4.5 Details of compare.s file

1. My function compare takes in 3 arguments - r0 has pointer to s1, r1 has pointer to s2, r2 has comparison mode. Finally it returns a value in [0,1,2] in r0, r0 = #0 means Both strings are equal, r0 = #1 means First string is less than second, r0 = #2 means First string is greater than second

2. The Body label checks if a character of one string is less or more than the other at the same position and if it is then returns the appropriate value in r0 back to main.

3. It has mainly 2 loops, Loop0 and Loop1, where Loop0 does the comparison in case-insensitive mode while Loop1 does the comparison in case-sensitive mode. In case-insensitive mode, I have converted all upper case characters to lower case characters.

4.6 Error Handling

1. If the comparison mode entered is anything other than 0 or 1, I print Invalid Input and exit my program.

2. If the duplicate removal option entered is anything other than 0 or 1, I print Invalid Input and exit my program.

3. If user inputs sizes of a string as 0 or less than 0, I give an error, saying Invalid Input and exit my program.

4. If user inputs any symbol/character other than an integer value in case of size, I print Invalid Input and exit my program.

5 Results

Test Inputs:

1. The user gets a prompt for every input he/she has to make. The input prompt gives a "Enter the size of list of strings:" prompt for entering the size of the unsorted list of strings, "Enter the list of strings:" prompt for entering the list of strings, "Enter comparison mode (0 for case-insensitive and 1 for case-sensitive):" prompt for entering comparison mode, "Enter the duplicate removal option (0 if duplicates not to be removed and 1 if duplicates are to be removed):" prompt for entering duplicate removal option.

2. Comparison mode has 0 as case-insensitive mode and 1 as case-sensitive mode.

3. Duplicate removal option takes in 0 if duplicates are not to be removed and 1 if duplicates are to be removed.

4. Input size of string list should not be 0.

Test Outputs:

1. First the size of the sorted list of strings is printed.

2. The sorted list of strings are printed one by one,each in a new line.

6 TestCases

1. Enter the size of list of strings:
   5
   Enter the list of strings:
   sree
   vidu
   sree
   mum
   vidu
   Enter the comparison mode(0 if case-insensitive and 1 if case-sensitive):
   0
   Enter the duplicate removal option (0 if duplicates not to be removed and 1 if duplicates are to be removed):
   1
   The size of final sorted list of strings:
   3
   The final sorted list of strings:
   mum
   sree
   vidu

2. Enter the size of list of strings: 0 Invalid input

3. Enter the size of list of strings:
   4
   Enter the list of strings:
   hi
   winner
   bye
   enjoy
   Enter the comparison mode(0 if case-insensitive and 1 if case-sensitive):
   1
   Enter the duplicate removal option (0 if duplicates not to be removed and 1 if duplicates are to be removed):
   0
   The size of final sorted list of strings:
   4
   The final sorted list of strings:
   bye
   enjoy
   hi
   winner

4. Enter the size of list of strings:
   9
   Enter the list of strings:
   ABC
   abc
   hi
   iitd
   iitk
   DEF
   GHI
   #%
   iitb
   Enter the comparison mode(0 if case-insensitive and 1 if case-sensitive):
   1
   Enter the duplicate removal option (0 if duplicates not to be removed and 1 if duplicates are to be removed):
   1
   The size of final sorted list of strings:
   9
   The final sorted list of strings:
   #%
   ABC
   DEF
   GHI
   abc

hi
iitb
iitd
iitk

5. Enter the size of list of strings:
   6
   Enter the list of strings:
   ABC
   abc
   hi
   DEF
   GHI
   #%
   Enter the comparison mode(0 if case-insensitive and 1 if case-sensitive):
   0
   Enter the duplicate removal option (0 if duplicates not to be removed and 1 if duplicates are to be removed):
   0
   The size of final sorted list of strings:
   6
   The final sorted list of strings:
   #%
   ABC
   abc
   DEF
   GHI
   hi

6. Enter the size of list of strings:
   k
   Invalid Input