COL226 Assignment 1

Stage 3

Rishabh Dhiman
2020CS10837

3 February 2022

1 Objective

Construct a function to sort a list of strings using merge sort.

2 Technical Details

- ARMSim version 2.0.1, Angel SWI Instructions
- Credits for `io.s` to Ramanuj Goel, 2020CS510437 – Github Link
- Refer to stage 1 report for the definition of a case-insensitive comparison.
- In case duplicates are being removed, the string to be deleted is arbitrarily chosen.
- It's assumed that the total space taken by the input strings won't exceed 4096 bytes and that there will be atmost 1024 strings.

3 Documentation

All the files defining functions from Stage 1 and 2 of the assignment are used. Two new files, `sort.s` and `test_sort.s` have been added.

`sort.s`

This files defines two functions `sort` and `merge_sort` function with the C++ signature,

```
int merge_sort(char** a, int len, int mode, char** res);
pair<char**, int> sort(char** a, int len, int mode);
```

`merge_sort`

It sorts the list of strings `a`, and stores it in `res`.

- `a` is an array of strings.
- `len` is the length of `a`.
- `mode` is an integer parameters to control the way in which the lists are merged
  - if the lowest bit of `mode` is set, then case-insensitive comparison takes place,
  - if the second bit of `mode` is set, then equal strings are deleted while merging.
- `res` is a word-aliged memory location where the merged is stored.
- The function finally returns a single integer representing the length of the sorted list.

In ARM, the arguments `a`, `len`, `mode`, `res` correspond to the registers `r0`, `r1`, `r2`, `r3`, respectively. The output, length, is stored in `r0`.

`sort`

This functions is very similar to the previous `merge_sort` function, however, rather than taking in the output array as `res` an argument it returns a pointer to the sorted list along with its length.

In ARM, the arguments `a`, `len`, `mode` correspond to the registers `r0`, `r1`, `r2`, respectively. The output, sorted list and its length, correspond to `r0` and `r1`, respectively.

`test_sort.s`

This file is used to test the `sort` function defined in `sort.s`. The file interacts with the user via console,

- It first asks the user, if the comparison done is case-insensitive or not.
- It then asks the user, if duplicate strings are to be removed or not.
- It then asks the user the for the number of strings, and then inputs the ASCII strings, each string in a new line.

It finally outputs the merged list of strings to the console, each string on a new line.

4 Tests and Results

To reproduce these, load `io.s`, `compare.s`, `merge.s`, `sort.s` and `test_sort.s` in ARMSim. Run it and follow the instructions printed on the console.

1. ```
   Case-insensitive compare (1 for case-insensitive comparison, 0 for
   ↪ case-sensitive): 0
   Remove duplicates from list (1 to remove duplicates, 0 to not remove):
   ↪ 0
   Number of strings in the list: 10
   Input the strings, each string on a new line:
   one
   one
   One
   two
   Three
   four
   five
   six
   seven
   Eight

   The sorted list is:
   Eight
   One
   Three
   five
   four
   one
   one
   seven
   six
   two
   ```

2. ```
   Case-insensitive compare (1 for case-insensitive comparison, 0 for
   ↪ case-sensitive): 0
   Remove duplicates from list (1 to remove duplicates, 0 to not remove):
   ↪ 1
   Number of strings in the list: 10
   Input the strings, each string on a new line:
   one
   two
   One
   three
   four
   one
   five
   six
   ```

```
seven
Eight

The sorted list is:
Eight
One
five
four
one
seven
six
three
two
```

3. ```
   Case-insensitive compare (1 for case-insensitive comparison, 0 for
   ↪ case-sensitive): 1
   Remove duplicates from list (1 to remove duplicates, 0 to not remove):
   ↪ 0
   Number of strings in the list: 10
   Input the strings, each string on a new line:
   one
   two
   One
   three
   four
   one
   five
   six
   seven
   Eight

   The sorted list is:
   Eight
   five
   four
   one
   One
   one
   seven
   six
   three
   two
   ```
   ```

4. ```
   Case-insensitive compare (1 for case-insensitive comparison, 0 for
   ↪ case-sensitive): 1
   Remove duplicates from list (1 to remove duplicates, 0 to not remove):
   ↪ 1
   Number of strings in the list: 10
   Input the strings, each string on a new line:
   one
   two
   One
   three
   four
   one
   Five
   six
   seven
   eight

   The sorted list is:
   eight
   Five
   four
   one
   seven
   six
   three
   two
   ```
   ```