Assignment -1

Name: Leela Sumanth Manduva

Round Robin VS Priority FCFS Analysis:

- Data is shown in ticks.
- The input file used for analysis has 1500 lines.

sort <input_filename>:

Bubble Sort :

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	59	51	55	58	53	**55.2**
Priority + FCFS	49	51	56	54	51	**52.2**

Insertion Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	55	51	53	50	52	52.2
Priority + FCFS	50	50	51	48	51	50

Selection Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	56	56	55	55	54	55.2
Priority + FCFS	55	53	55	53	55	54.2

sort -b <input_filename>:

Bubble Sort :

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	60	66	67	69	71	**66.6**
Priority + FCFS	73	72	73	74	75	**73.4**

Insertion Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	49	50	48	49	50	49.2
Priority + FCFS	52	54	51	53	55	53

Selection Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	71	72	71	71	72	71.4
Priority + FCFS	81	81	79	76	74	78.2

sort -r <input_filename>:

Bubble Sort :

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	53	53	52	53	55	**53.2**
Priority + FCFS	52	53	53	53	53	**52.8**

Insertion Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	48	49	49	50	53	49.5
Priority + FCFS	50	51	52	55	51	51.8

Selection Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	54	57	53	55	53	54.4
Priority + FCFS	56	55	55	57	56	55.8

sort -o <output_file> <input_filename>:

Bubble Sort :

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	394	401	393	389	396	**394.6**
Priority + FCFS	410	409	403	397	391	**411.18**

Insertion Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	393	394	392	389	384	390.4
Priority + FCFS	391	407	402	419	411	406

Selection Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	386	400	395	389	388	391.6
Priority + FCFS	400	408	407	407	415	407.4

sort -n <input_filename>:

Bubble Sort :

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	51	56	54	52	55	**53.6**
Priority + FCFS	49	55	52	53	55	**52.8**

Insertion Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	51	53	51	51	50	51.2
Priority + FCFS	52	49	49	50	51	50.2

Selection Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	54	58	54	54	59	55.8
Priority + FCFS	54	52	53	54	54	53.4

sort -M <input_filename>:

Bubble Sort :

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	65	68	63	69	64	**65.8**
Priority + FCFS	59	58	60	65	62	**60.8**

Insertion Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	48	50	50	51	51	50
Priority + FCFS	48	50	49	52	51	50

Selection Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	53	56	53	52	53	53.4
Priority + FCFS	52	51	54	55	55	53.4

sort -u <input_filename>:

Bubble Sort :

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	54	57	55	54	58	**55.6**
Priority + FCFS	56	53	57	57	57	**56**

Insertion Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	53	53	52	54	54	53.2
Priority + FCFS	54	52	52	53	56	53.4

Selection Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	48	52	55	54	56	53
Priority + FCFS	59	55	56	59	55	56.8

sort -User input:

- The ticks value may be subject to user input delay for this command as the user gives the input.

Bubble Sort :

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	29	30	29	27	27	**28.4**
Priority + FCFS	25	26	30	29	28	**27.6**

Insertion Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	31	26	25	25	26	26.4
Priority + FCFS	30	29	28	30	32	29.8

Selection Sort:

Schedulers	Test 1	Test 2	Test 3	Test 4	Test 5	Average Ticks
Round Robin	26	27	27	27	26	26.6
Priority + FCFS	26	25	26	30	24	26.2

References:

1. To trim the string - https://www.geeksforgeeks.org/c-program-to-trim-leading-white-spaces-from-string/
2. Bubble sort logic - https://www.geeksforgeeks.org/bubble-sort/
3. Insertion sort logic - https://www.geeksforgeeks.org/insertion-sort/
4. Selection sort logic - https://www.geeksforgeeks.org/selection-sort/
5. For input handling, I have used the demo code posted on canvas as a reference for getting input from the file.

Bubble Sort:

bubblesort Default:

```
$ bubblesort
zebra
sumanth
vinay
1
kamal
157
fayaz
100
vijay
samantha
0
0
1
100
157
fayaz
kamal
samantha
sumanth
vijay
vinay
zebra
$
```

bubblesort -b <filename>:

- Sorting logic is the same
- Before comparing two strings trimLeadingSpaces(Reference 1) function is called.
- This function deletes the leading spacing in a string and returns it.

```
$ bubblesort -b datafile.txt
0
     1
1
100
157
fayaz
kamal
samantha
samantha
      sumanth
sumanth
vijay
vinay
zebra
zebra
$
```

bubblesort -r <filename>:

```
$ bubblesort -r datafile.txt
zebra
zebra
vinay
vijay
sumanth
samantha
samantha
kamal
fayaz
157
100
1
0
     1
      sumanth
$
```

bubblesort -o <output_filename> <filename>:

- I used the fprintf function to print each line into a file.

```
$ bubblesort -o output_file.txt datafile.txt
$ cat output_file.txt
      sumanth
     1
0
1
100
157
fayaz
kamal
samantha
samantha
sumanth
vijay
vinay
zebra
zebra
$
```

bubblesort -n <filename>:

- Uses the same logic but while comparing atoi(converts string to integer) function is used.

```
$ bubblesort -n datafile2.txt
0
1
2
3
50
100
101
123
$
```

bubblesort -M <filename>:

- While taking input from the file if the first 3 letters of a string are equal to months, then that string is replaced by a dummy value.
- And the matching string is moved to a different array.
- Both these arrays are sorted differently.
- While printing first the non-matching array is printed followed by the matching array.

```
$ bubblesort -M datafile2.txt
jan
feb
mar
apr
apr
aug
sep
nov
dec
$
```

bubblesort -u <filename>:

- The array is sorted.
- Then If a string is repeated in the data it is replaced with a dummy string.
- While printing the dummy string is skipped.

```
$ bubblesort -u datafile.txt
      sumanth
     1
0
1
100
157
fayaz
kamal
samantha
sumanth
vijay
vinay
zebra
$
```

<u>Insertion Sort:</u>

insertionsort Default:

```
$ insertionsort
zebra
sumanth
vinay
1
kamal
157
fayaz
100
vijay
samantha
0
0
1
100
157
fayaz
kamal
samantha
sumanth
vijay
vinay
zebra
$
```

insertionsort -b <filename>:

- Sorting logic is the same
- Before comparing two strings trimLeadingSpaces(Reference 1) function is called.
- This function deletes the leading spacing in a string and returns it.

```
$ insertionsort -b datafile.txt
0
     1
1
100
157
fayaz
kamal
samantha
samantha
      sumanth
sumanth
vijay
vinay
zebra
zebra
$
```

insertionsort -r <filename>:

```
$ insertionsort -r datafile.txt
zebra
zebra
vinay
vijay
sumanth
samantha
samantha
kamal
fayaz
157
100
1
0
     1
      sumanth
$
```

insertionsort -o <output_filename> <filename>:

- I used the fprintf function to print each line into a file.

```
$ insertionsort -o output_file.txt datafile.txt
$ cat output_file.txt
      sumanth
     1
0
1
100
157
fayaz
kamal
samantha
samantha
sumanth
vijay
vinay
zebra
zebra
$
```

insertionsort -n <filename>:

- Uses the same logic but while comparing atoi(converts string to integer) function is used.

```
$ insertionsort -n datafile2.txt
0
1
2
3
50
100
101
123
$
```

insertionsort -M <filename>:

- While taking input from the file if the first 3 letters of a string are equal to months then that string is replaced by a dummy value.
- And the matching string is moved to a different array.

- Both these arrays are sorted differently.
- While printing first the non-matching array is printed followed by the matching array.

```
$ insertionsort -M datafile2.txt
jan
feb
mar
apr
apr
aug
sep
nov
dec
```

insertionsort -u <filename>:

- The array is sorted.
- Then If a string is repeated in the data it is replaced with a dummy string.
- While printing the dummy string is skipped.

```
$ insertionsort -u datafile.txt
      sumanth
     1
0
1
100
157
fayaz
kamal
samantha
sumanth
vijay
vinay
zebra
$
```

<u>Selection Sort:</u>

selectionsort Default:

```
$ selectionsort
zebra
sumanth
vinay
1
kamal
157
fayaz
100
vijay
samantha
0
0
1
100
157
fayaz
kamal
samantha
sumanth
vijay
vinay
zebra
$
```

selectionsort -b <filename>:

- Sorting logic is the same
- Before comparing two strings trimLeadingSpaces(Reference 1) function is called.
- This function deletes the leading spacing in a string and returns it.

```
$ selectionsort -b datafile.txt
0
     1
1
100
157
fayaz
kamal
samantha
samantha
      sumanth
sumanth
vijay
vinay
zebra
zebra
$
```

selectionsort -r <filename>:

```
$ selectionsort -r datafile.txt
zebra
zebra
vinay
vijay
sumanth
samantha
samantha
kamal
fayaz
157
100
1
0
     1
      sumanth
$
```

selectionsort -o <output_filename> <filename>:

- I used the fprintf function to print each line into a file.

```
$ selectionsort -o output_file.txt datafile.txt
$ cat output_file.txt
      sumanth
     1
0
1
100
157
fayaz
kamal
samantha
samantha
sumanth
vijay
vinay
zebra
zebra
$
```

selectionsort -n <filename>:

- Uses the same logic but while comparing atoi(converts string to integer) function is used.

```
$ selectionsort -n datafile2.txt
0
1
2
3
50
100
101
123
$
```

selectionsort -M <filename>:

- While taking input from the file if the first 3 letters of a string are equal to months, then that string is replaced by a dummy value.
- And the matching string is moved to a different array.
- Both these arrays are sorted differently.
- While printing first the non-matching array is printed followed by the matching array.

```
$ selectionsort -M datafile2.txt
jan
feb
mar
apr
apr
aug
sep
nov
dec
$
```

selectionsort -u <filename>:

- The array is sorted.
- Then if a string is repeated in the data it is replaced with a dummy string.
- While printing the dummy string is skipped.

```
$ selectionsort -u datafile.txt
      sumanth
     1
0
1
100
157
fayaz
kamal
samantha
sumanth
vijay
vinay
zebra
$
```