Homework #7

01286120 Elementary Systems Programming

Software Engineering Program

Faculty of Engineering, KMITL

By

66011149 Phatthadon Sornplang

1. Write a program that sort values in a list of numbers

1.1) Write a program that takes values from the command-line as a list of numbers and use Vec::sort_by to sort numbers in ascending and descending order. Add an integration test to verify the correctness of the program.

```
    Finished dev [unoptimized + debuginfo] target(s) in 0.50s
     Running `target\debug\Q1_1.exe 7 8 6 9`
[([9, 8, 7, 6], [6, 7, 8, 9])]
```

```
    Finished test [unoptimized + debuginfo] target(s) in 0.33s
     Running unittests src\main.rs (target\debug\deps\Q1_1-838188370cdeb37c.exe)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running tests\cli.rs (target\debug\deps\cli-2ef26b39916627c0.exe)

running 1 test
test main ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.01s
```

1.2) Modify the program in 1.1) to use loop and bubble sort instead of Vec::sort_by in order to achieve the same result. Add an integration test to verify the correctness of the program.

```
    Finished dev [unoptimized + debuginfo] target(s) in 0.07s
     Running `target\debug\Q1_2.exe 7 8 6 9`
[([9, 8, 7, 6], [6, 7, 8, 9])]
```

```
    Finished test [unoptimized + debuginfo] target(s) in 0.03s
     Running unittests src\main.rs (target\debug\deps\Q1_2-8f11ea150963c92c.exe)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running tests\cli.rs (target\debug\deps\cli-10cf5f4c5937dda2.exe)

running 1 test
test main1 ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.01s
```

2. Write a program that sort points in a list of points

2.1) Write a program that takes values from the command-line as a list of the pair (x, y) for the point coordinate (for example, cargo run — 1 5 2 7 3 will make the point list [(1., 5.), (2., 7.)] (discarding 3)) and use Vec::sort_by to sort points by x then y in ascending and descending order. Add an integration test to verify the correctness of the program.

```
    Finished dev [unoptimized + debuginfo] target(s) in 0.07s
     Running `target\debug\Q2_1.exe 8 9 0 7 6 9 5`
Ascending X: [(0.0, 7.0), (6.0, 9.0), (8.0, 9.0)]
Descending X: [(8.0, 9.0), (6.0, 9.0), (0.0, 7.0)]
Ascending Y: [(0.0, 7.0), (8.0, 9.0), (6.0, 9.0)]
Descending Y: [(8.0, 9.0), (6.0, 9.0), (0.0, 7.0)]
```

```
    Finished test [unoptimized + debuginfo] target(s) in 0.29s
     Running unittests src\main.rs (target\debug\deps\Q2_1-98467a8205a0cc53.exe)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running tests\cli.rs (target\debug\deps\cli-6cf707591189419e.exe)

running 2 tests
test main ... ok
test main1 ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.01s
```

2.2) Modify the program in 2.1) to use loop and bubble sort instead of Vec::sort_by in order to achieve the same result. Add an integration test to verify the correctness of the program. 16 01286120 Elementary Systems Programming

```
    Finished dev [unoptimized + debuginfo] target(s) in 0.06s
     Running `target\debug\Q2_2.exe 8 9 0 7 6 9 5`
Descending X: [(8.0, 9.0), (6.0, 9.0), (0.0, 7.0)]
Ascending X: [(0.0, 7.0), (6.0, 9.0), (8.0, 9.0)]
Descending Y: [(8.0, 9.0), (6.0, 9.0), (0.0, 7.0)]
Ascending Y: [(0.0, 7.0), (8.0, 9.0), (6.0, 9.0)]
```

```
    Finished test [unoptimized + debuginfo] target(s) in 0.31s
     Running unittests src\main.rs (target\debug\deps\Q2_2-a0fb08e1053d3c3f.exe)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running tests\cli.rs (target\debug\deps\cli-326a5b24b96e2ae3.exe)

running 2 tests
test main ... ok
test main1 ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.01s
```

3. Write a program that generates a simple HTML table

The following code is an example of a simple HTML table that can be opened in browser: HTML code Visual representation Col 1 Col 2 Col 3 Cell 1 Cell 2 Cell 3 Cell 4 Cell 5 Cell 6

3.1) From the Fahrenheit to Celsius formula °C = (5/9)(°F - 32) , write a program that prints conversion tables similar to the homework exercise 2.1) from Lab #3 in HTML format instead of plain text: The following table shows the example output from the homework exercise 2.1) from Lab #3:

```
Finished dev [unoptimized + debuginfo] target(s) in 0.01s
 Running `C:\Users\phatt\OneDrive\Desktop\Code Files\Rust\Lab_7\HW\Q3_1\target\debug\Q3_1.exe 0 300 20`
```

Fahrenheit	Celcius
0	-17.8
20	-6.7
40	4.4
60	15.6
80	26.7
100	37.8
120	48.9
140	60.0
160	71.1
180	82.2
200	93.3
220	104.4
240	115.6
260	126.7
280	137.8
300	148.9

3.2) write a program that prints tables in HTML format showing the number x , x , and x in separate columns. The program should take similar command-line format as in the exercise 3.1)

```
Finished dev [unoptimized + debuginfo] target(s) in 0.02s
 Running `target\debug\Q3_2.exe 20 200 10`
```

x	x^2	x^3
20.00	400.00	8000.00
30.00	900.00	27000.00
40.00	1600.00	64000.00
50.00	2500.00	125000.00
60.00	3600.00	216000.00
70.00	4900.00	343000.00
80.00	6400.00	512000.00
90.00	8100.00	729000.00
100.00	10000.00	1000000.00
110.00	12100.00	1331000.00
120.00	14400.00	1728000.00
130.00	16900.00	2197000.00
140.00	19600.00	2744000.00
150.00	22500.00	3375000.00
160.00	25600.00	4096000.00
170.00	28900.00	4913000.00
180.00	32400.00	5832000.00
190.00	36100.00	6859000.00
200.00	40000.00	8000000.00