Assignment #2

Course: SEG3103

Authors: Fatimah Vakily (300125671) and Jasmine Kokkat (300115249)

Professor: Andrew Forward

TAs: Zahra Kakavand, Nazanin Bayati Chaleshtari, Henry Chen

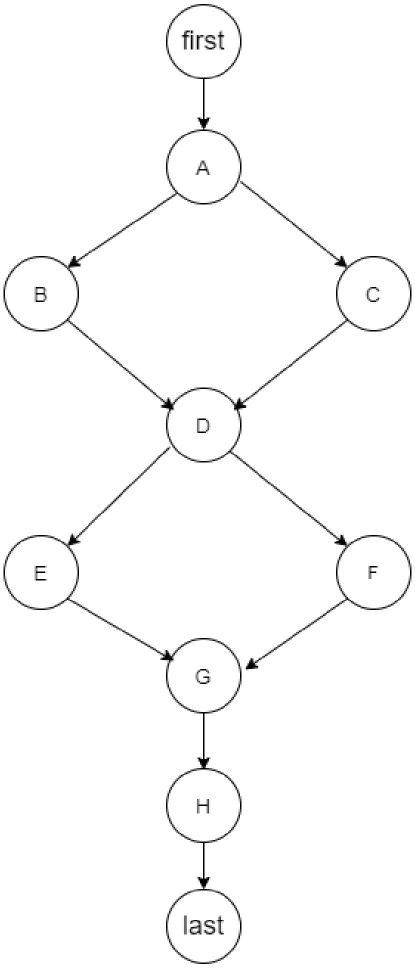
June 10, 2021

Question 1.1 (10%)

Draw the simplified control flow graph corresponding to each of the methods percentage_grade, letter_grade, and numeric_grade.

percentage_grade

```
def percentage_grade(%{homework: homework, labs: labs, midterm: midterm, final: final}) do
  avg_homework =
   if Enum.count(homework) == 0 do (A)
    0 (B)
   else
    Enum.sum(homework) / Enum.count(homework) (C)
   end

  avg_labs =
   if Enum.count(labs) == 0 do (D)
    0 (E)
   else
    Enum.sum(labs) / Enum.count(labs) (F)
   end

  mark = 0.2 * avg_labs + 0.3 * avg_homework + 0.2 * midterm + 0.3 * final (G)
  round(mark * 100) (H)
 end
```

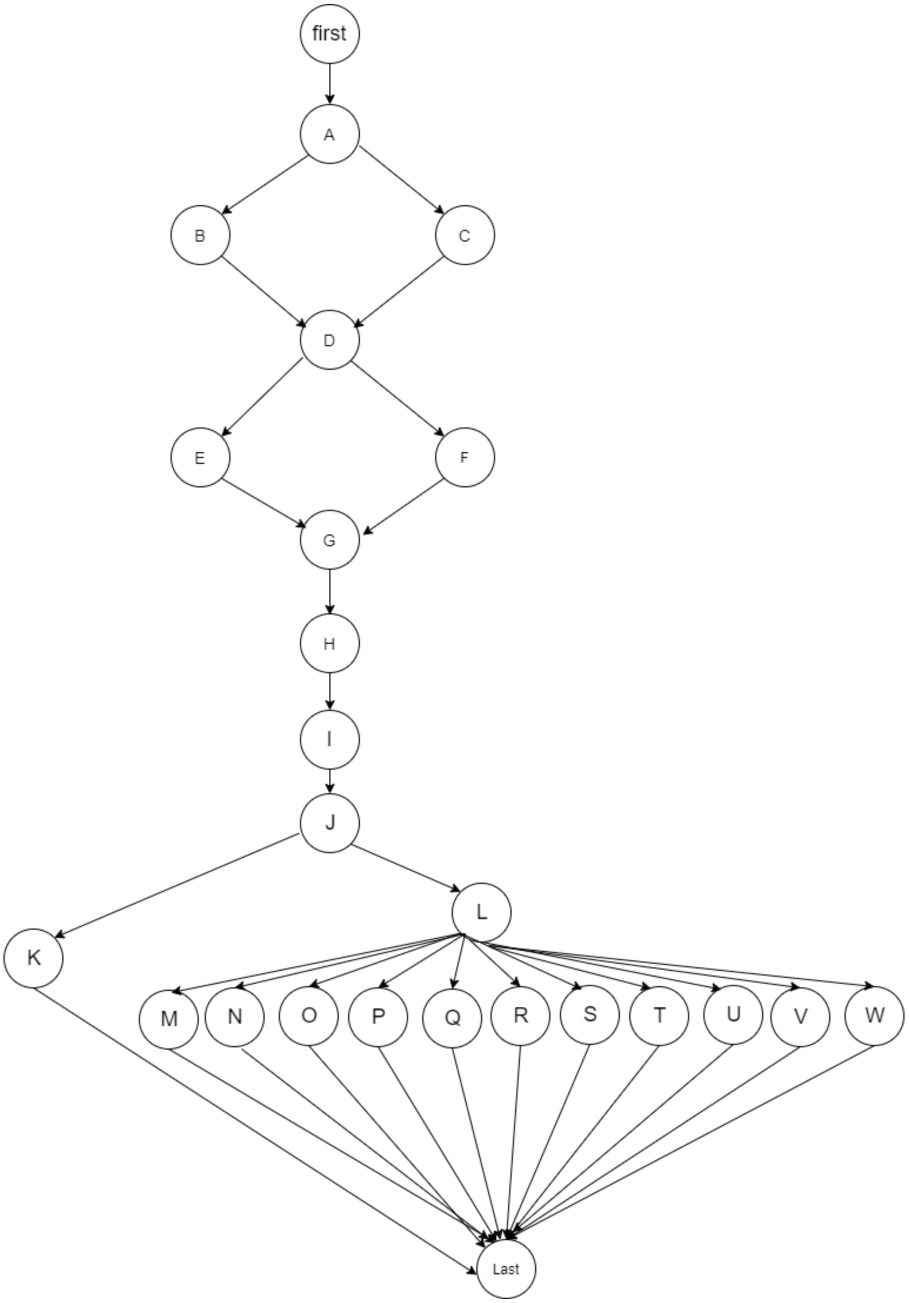


letter_grade

```
def letter_grade(%{homework: homework, labs: labs, midterm: midterm, final: final}) do
  avg_homework =
   if Enum.count(homework) == 0 do (A)
    0 (B)
   else
    Enum.sum(homework) / Enum.count(homework) (C)
   end

  avg_labs =
   if Enum.count(labs) == 0 do (D)
```

```
    0 (E)
  else
    Enum.sum(labs) / Enum.count(labs) (F)
  end

avg_exams = (midterm + final) / 2 (G)

num_labs =
  labs
  |> Enum.reject(fn mark -> mark < 0.25 end) (H)
  |> Enum.count() (I)

if avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 do (J)
  "EIN" (K)
else
  mark = 0.2 * avg_labs + 0.3 * avg_homework + 0.2 * midterm + 0.3 * final (L)

  cond do
    mark > 0.895 -> "A+" (M)
    mark > 0.845 -> "A" (N)
    mark > 0.795 -> "A-" (O)
    mark > 0.745 -> "B+" (P)
    mark > 0.695 -> "B" (Q)
    mark > 0.645 -> "C+" (R)
    mark > 0.595 -> "C" (S)
    mark > 0.545 -> "D+" (T)
    mark > 0.495 -> "D" (U)
```

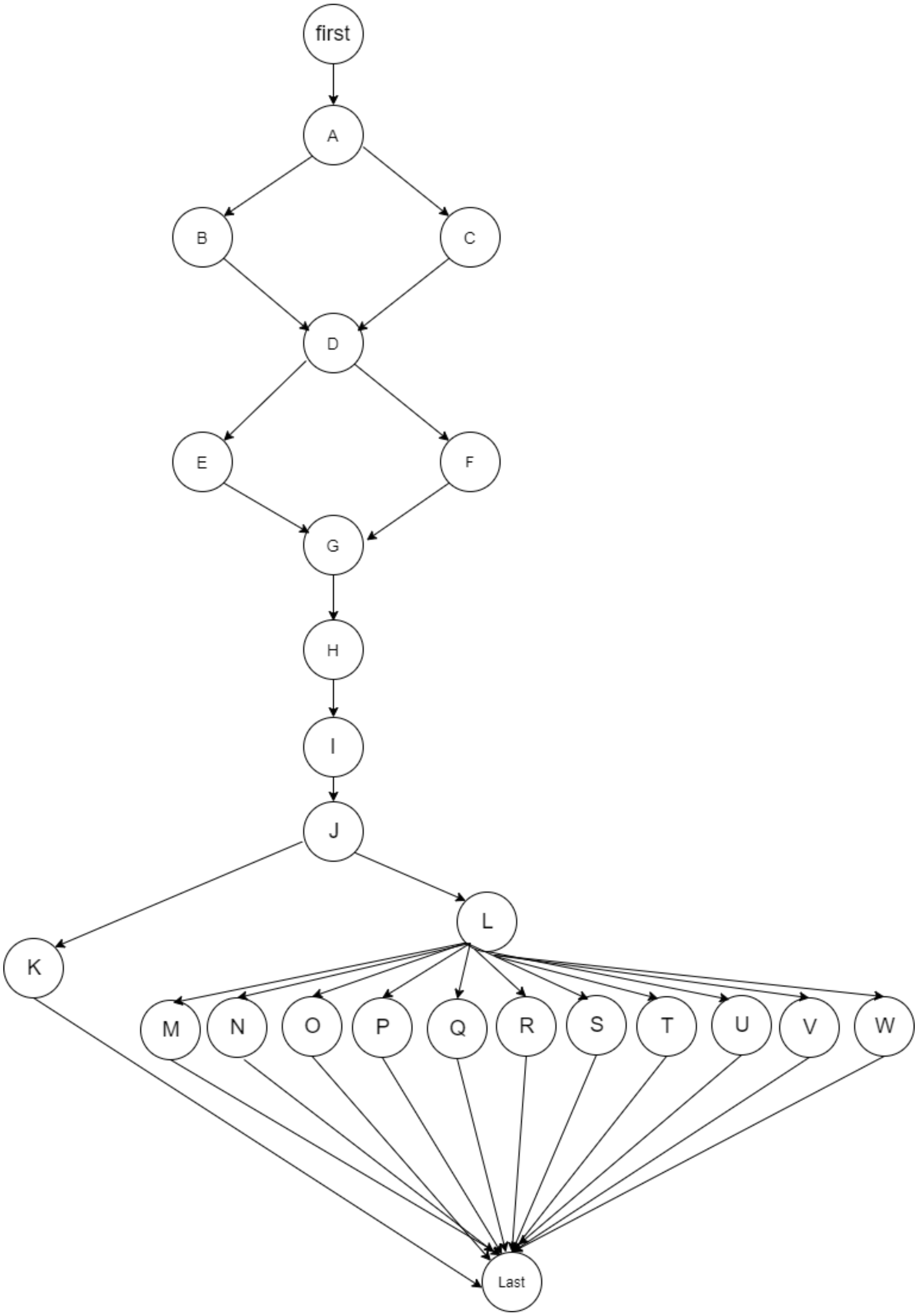
```
      mark > 0.395 -> "E" (V)
      :else -> "F" (W)
    end
  end
end
```



numeric_grade

```
def numeric_grade(%{homework: homework, labs: labs, midterm: midterm, final: final}) do
  avg_homework =
   if Enum.count(homework) == 0 do (A)
    0 (B)
   else
    Enum.sum(homework) / Enum.count(homework) (C)
   end

  avg_labs =
   if Enum.count(labs) == 0 do (D)
    0 (E)
   else
    Enum.sum(labs) / Enum.count(labs) (F)
   end

  avg_exams = (midterm + final) / 2 (G)

  num_labs =
   labs
   |> Enum.reject(fn mark -> mark < 0.25 end) (H)
   |> Enum.count() (I)

  if avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 do (J)
   0 (K)
  else
   mark = 0.2 * avg_labs + 0.3 * avg_homework + 0.2 * midterm + 0.3 * final (L)
```

```
    cond do
      mark > 0.895 -> 10 (M)
      mark > 0.845 -> 9 (N)
      mark > 0.795 -> 8 (O)
      mark > 0.745 -> 7 (P)
      mark > 0.695 -> 6 (Q)
      mark > 0.645 -> 5 (R)
      mark > 0.595 -> 4 (S)
      mark > 0.545 -> 3 (T)
      mark > 0.495 -> 2 (U)
      mark > 0.395 -> 1 (V)
      :else -> 0 (W)
    end
  end
end
```


first
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
Last

Question 1.2 (20%)

Provide a white box test design for 100% branch coverage of the methods percentage_grade, letter_grade, and numeric_grade. Your test suite will be evaluated on the number of its test cases (try to have the smallest possible number of test cases that will allow 100% branch coverage). Use the following template for your test case design:

Test Case Number	Test Data	Expected Results	Conditions Covered	Branches Covered
Sample	homework: [0.8] labs: [1, 1, 1] midterm: 0.70 final: 0.9	85	Enum.count(homework) != 0 Enum.count(labs) != 0	First (Entry) – A - C –D –F– G –H –Last (Exit)
1	homework: [0.7] labs: [0.75] midterm: 0.7 final: 0.9	77	Enum.count(homework) != 0 Enum.count(labs) != 0	First (Entry) – A - C –D –F– G –H –Last (Exit)
2	homework: [] labs: [] midterm: 0.0 final: 0.0	0	Enum.count(homework) == 0 Enum.count(labs) == 0	First (Entry) – A - B –D –E– G –H –Last (Exit)
3	homework: [] labs: [] midterm: 0.3 final: 0.3	EIN	Enum.count(homework) == 0 Enum.count(labs) == 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3	First (Entry) – A - B –D –E– G –H –I -J -K - Last (Exit)
4	homework: 0.9 labs: 0.9,0.93,0.9 midterm:1. 0 final: 1.0	A+	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.895	First (Entry) –A - C -D– F– G –H –I -J –L –M -Last (Exit)

5	homework: 0.87 labs: [0.89,084,084] midterm: 0.88 final: 0.89	A	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.845	First (Entry) –A -C -D– F– G –H –I -J –L –N -Last (Exit)
6	homework: 0.82 labs: [0.83,0.82, 0.81] midterm: 0.83 final: 0.8	A-	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.795	First (Entry) –A -C -D– F– G –H –I -J –L –O -Last (Exit)
7	homework: 0.78 labs: [0.76,0.75,0.77] midterm: 0.78 final: 0.76	B+	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.745	First (Entry) –A -C -D– F– G –H –I -J –L –P -Last (Exit)
8	homework: 0.71 labs: [0.73,0.72,0.7] midterm: 0.74 final: 0.7	B	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.695	First (Entry) –A -C -D– F– G –H –I -J –L –Q-Last (Exit)
9	homework: 0.66 labs: [0.65,0.62,0.6] midterm: 0.65 final: 0.66	C+	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3	First (Entry) –A -C -D– F– G –H –I -J –L –R -Last (Exit)

<table>
<tr><td></td><td></td><td></td><td>mark > 0.645</td><td></td></tr>
<tr><td>10</td><td>homework: 0.6
labs: 0.61, 0.62, 0.6]
midterm: 0.61
final: 0</td><td>C</td><td>Enum.count(homework) != 0
Enum.count(labs) != 0
avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3
mark > 0.595</td><td>First (Entry) –A -C -D– F– G –H –I -J –L –S -Last (Exit)</td></tr>
<tr><td>11</td><td>homework: 0.56
labs: [0.55,0.56,0.57]
midterm: 0.56
final: 0.55</td><td>D+</td><td>Enum.count(homework) != 0
Enum.count(labs) != 0
avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3
mark > 0.545</td><td>First (Entry) –A -C -D– F– G –H –I -J –L –T -Last (Exit)</td></tr>
<tr><td>12</td><td>homework: 0.51
labs: [05,0.51, 0.52]
midterm: 0.51
final: 0.5</td><td>D</td><td>Enum.count(homework) != 0
Enum.count(labs) != 0
avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3
mark > 0.495</td><td>First (Entry) –A -C -D– F– G –H –I -J –L –U -Last (Exit)</td></tr>
<tr><td>13</td><td>homework: 0.4
labs: 0.45,0.45,0.45
midterm: 0.4
final: 0.4</td><td>E</td><td>Enum.count(homework) != 0
Enum.count(labs) != 0
avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3
mark > 0.395</td><td>First (Entry) –A -C -D– F– G –H –I -J –L –V -Last (Exit)</td></tr>
<tr><td>14</td><td>homework: 0.4
labs: 0.34,0.35,0.33
midterm: 0.4
final: 0.4</td><td>F</td><td>Enum.count(homework) != 0
Enum.count(labs) != 0</td><td>First (Entry) –A -C -D– F– G –H –I -J –L –W -Last (Exit)</td></tr>
</table>

			avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 (ADJW)	
15	homework: [] labs: [] midterm: 0.15 final: 0.2	0	Enum.count(homework) == 0 Enum.count(labs) == 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3	First (Entry) –A - B -D– E– G –H –I -J –K –Last (Exit)
16	homework: [0.95] labs: [0.9,0.92,0.9] midterm: 1.0 final: 1.0	10	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.895	First (Entry) – A - C–D- F– G – H – I – J – L – M – Last (Exit)
17	homework: [0.87] labs: [0.87,0.88,0.87] midterm: 0.89 final: 0.86	9	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.845	First (Entry) – A - C–D- F– G – H – I – J – L – N – Last (Exit)
18	homework: [0.81] labs: [0.8,0.81,0.8] midterm: 0.82 final: 0.83	8	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.795	First (Entry) – A - C–D- F– G – H – I – J – L – O – Last (Exit)
19	homework: [0.76]	7	Enum.count(homework) != 0	First (Entry) – A - C–D- F– G – H – I

	labs: [0.75,0.77,0.78] midterm: 0.76 final: 0.75		Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.745	– J – L – P – Last (Exit)
20	homework: [0.72] labs: [0.73,0.71,0.7] midterm: 0.72 final: 0.7	6	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.695	First (Entry) – A - C–D- F– G – H – I – J – L – Q – Last (Exit)
21	homework: [0.66] labs: [0.66, 0.65, 0.65] midterm: 0.65 final: 0.65	5	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.645	First (Entry) – A - C–D- F– G – H – I – J – L – R – Last (Exit)
22	homework: [0.61], labs: [0.61, 0.62, 0.6], midterm: 0.63, final: 0.6	4	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.595	First (Entry) – A - C–D- F– G – H – I – J – L – S – Last (Exit)
23	homework: [0.56] labs: [0.56, 0.55, 0.57], midterm: 0.56, final: 0.55	3	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.545	First (Entry) – A - C–D- F– G – H – I – J – L – T – Last (Exit)

24	homework: [0.51] labs: [0.51, 0.52, 0.5] midterm: 0.5 final: 0.5	2	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.495	First (Entry) – A - C–D- F– G – H – I – J – L – U – Last (Exit)
25	homework: [0.4] labs: [0.45, 0.46, 0.45] midterm: 0.4, final: 0.4	1	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 mark > 0.395	First (Entry) – A - C–D- F– G – H – I – J – L – V– Last (Exit)
26	homework: [0.4] labs: [0.36, 0.36, 0.35] midterm: 0.4 final: 0.4	0	Enum.count(homework) != 0 Enum.count(labs) != 0 avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 (ADJW)	First (Entry) – C–D-F– G – H – I – J – L – W – Last (Exit)

Question 1.3 (15%)

Provide an implementation of your test suite using ExUnit.

(In zip folder)

Question 1.4 (5%)

What is the degree of statement coverage obtained? If you weren’t able to achieve

100% coverage explain why.

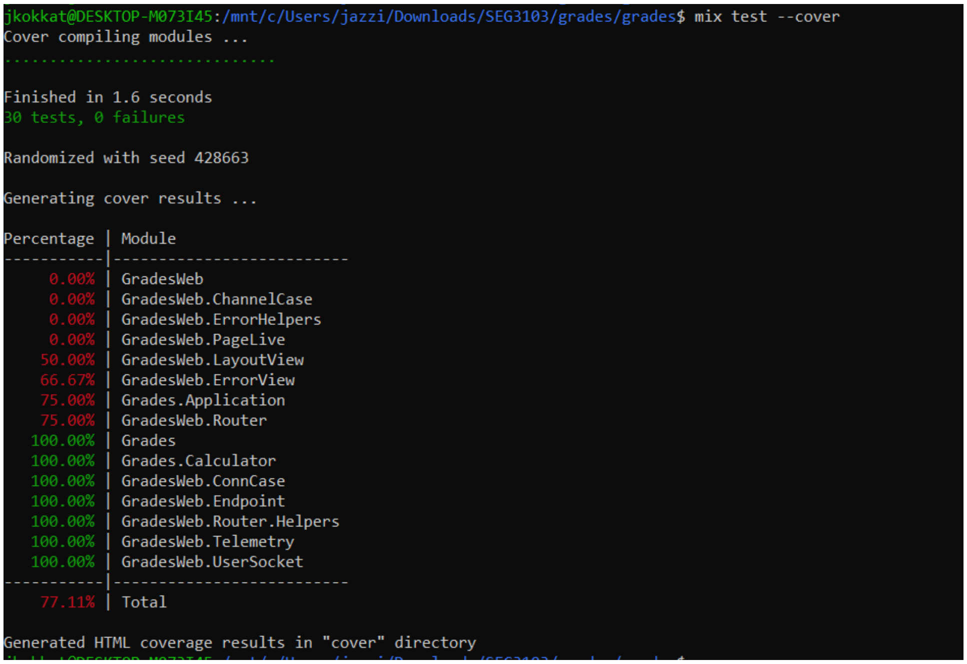
Please be sure to attach screenshots of your coverage results.

Elixir’s coverage tool is primitive, as it only provides statement level accuracy.

mix test --cover

How might you address the limitations of a testing tool that only provides statement level coverage?

100% coverage was achieved for Grades.Calculator through the ExUnit test suite. The limitations of statement level coverage in this case are that we can only test conditions that we expect to be valid input, without testing false conditions. For example, if the value of A was read instead of a 3, this would cause an error that cannot be covered by statement coverage. I would address this limitation by restricting what the user can enter as input so that this error would not occur.



```
jkokkat@DESKTOP-M073I45:/mnt/c/Users/jazzi/Downloads/SEG3103/grades/grades$ mix test --cover
Cover compiling modules ...
..............................

Finished in 1.6 seconds
30 tests, 0 failures

Randomized with seed 428663

Generating cover results ...

Percentage | Module
-----------|--------------------------
     0.00% | GradesWeb
     0.00% | GradesWeb.ChannelCase
     0.00% | GradesWeb.ErrorHelpers
     0.00% | GradesWeb.PageLive
    50.00% | GradesWeb.LayoutView
    66.67% | GradesWeb.ErrorView
    75.00% | Grades.Application
    75.00% | GradesWeb.Router
   100.00% | Grades
   100.00% | Grades.Calculator
   100.00% | GradesWeb.ConnCase
   100.00% | GradesWeb.Endpoint
   100.00% | GradesWeb.Router.Helpers
   100.00% | GradesWeb.Telemetry
   100.00% | GradesWeb.UserSocket
-----------|--------------------------
    77.11% | Total

Generated HTML coverage results in "cover" directory
```

Question 2:

Github Repo:
https://github.com/Fatimbit/seg3101_playground/tree/master/asg02