# SEG3103 - Assignment 2

By: Youcef Ben Ali
Student Number: 300110797

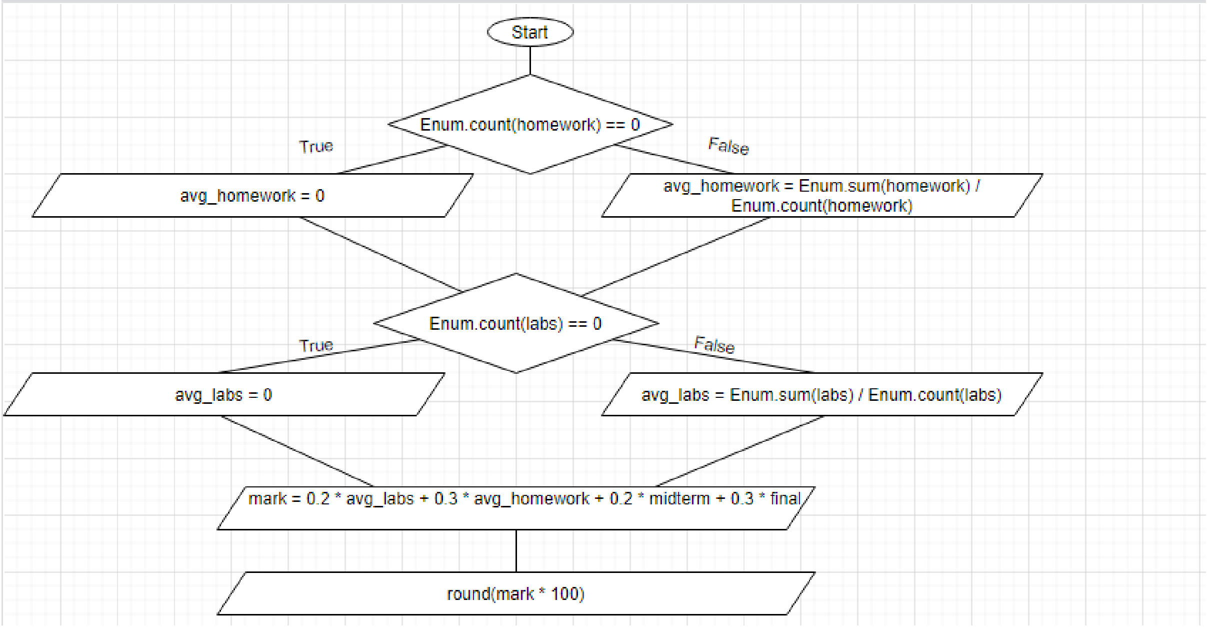
Dr. Andrew Forward

June 10, 2021

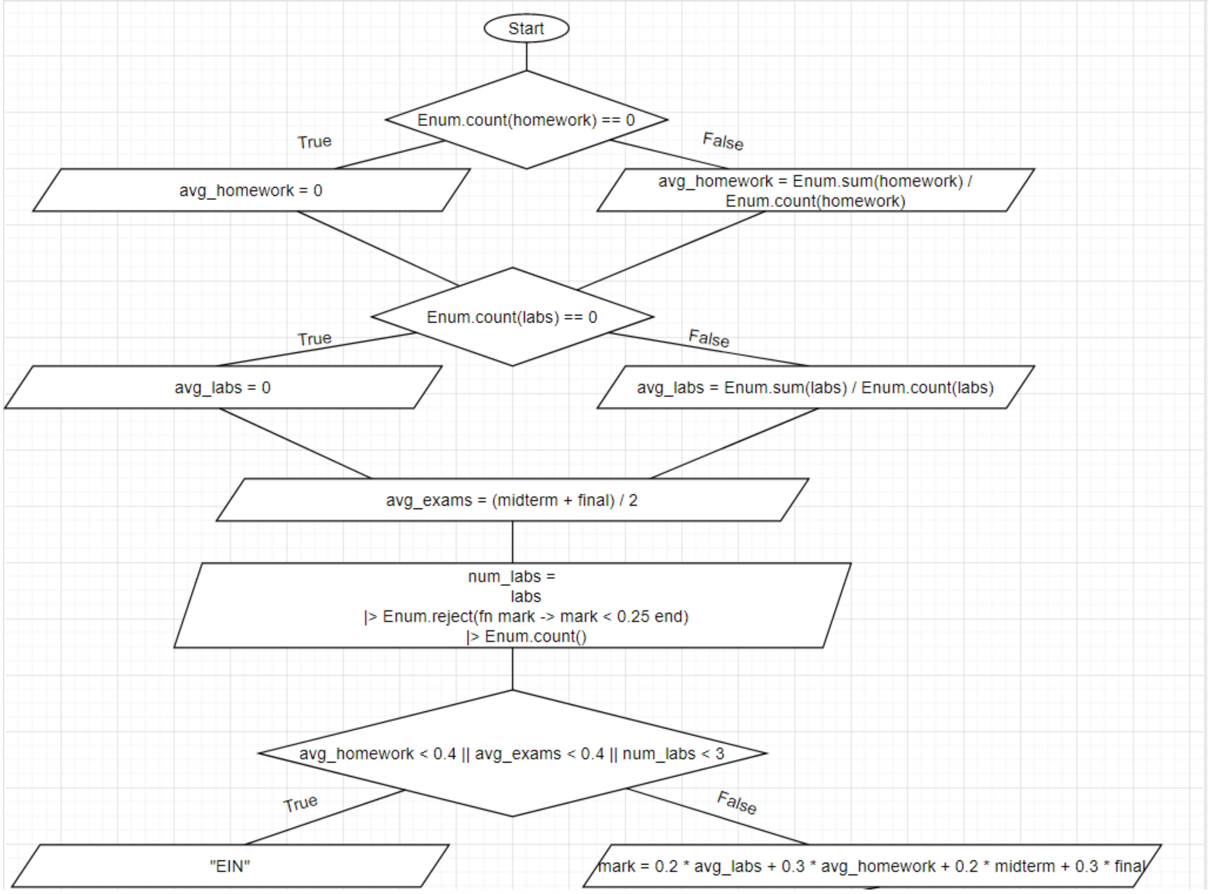
University of Ottawa
Seg3103

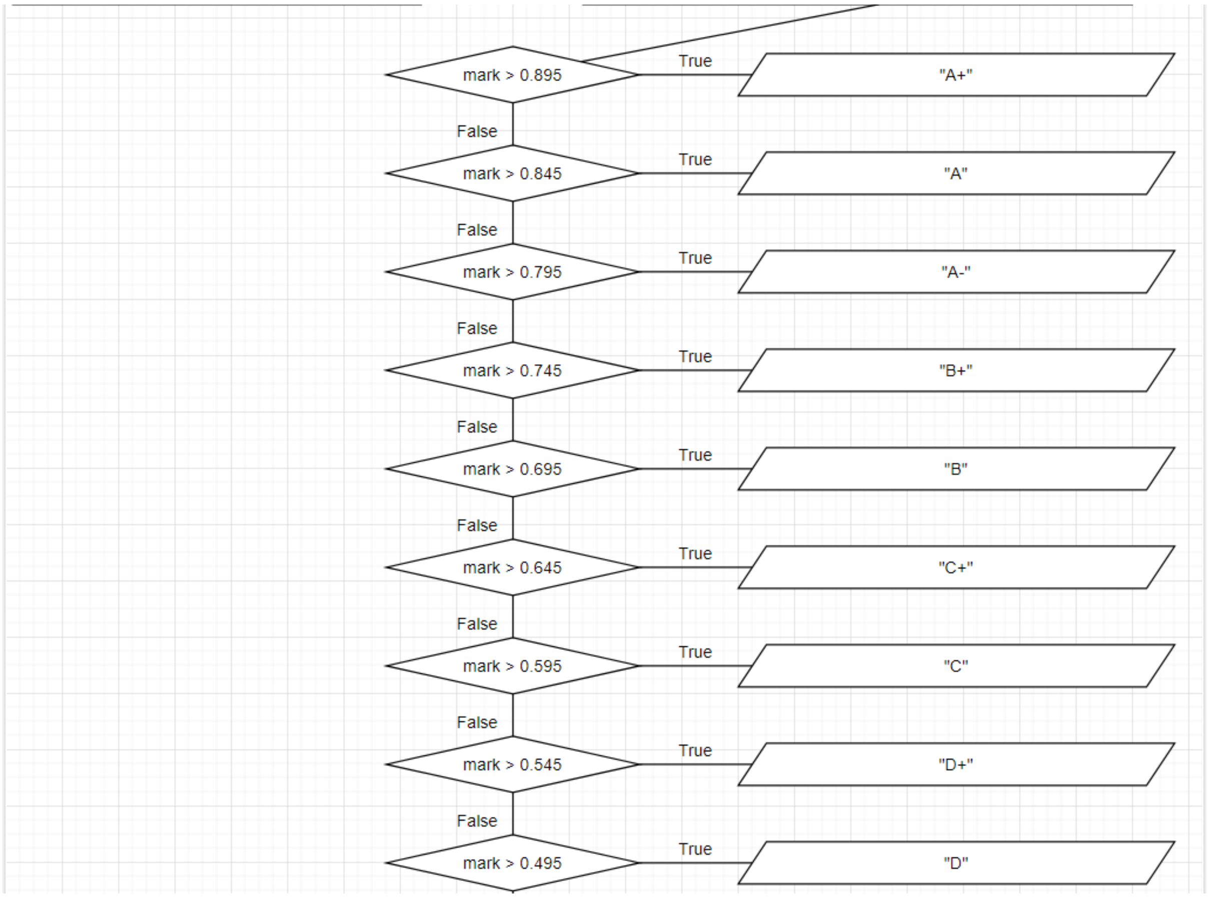
Question 1 - **Draw the simplified control flow graph corresponding to each of the methods percentage_grade, letter_grade, and numeric_grade.**

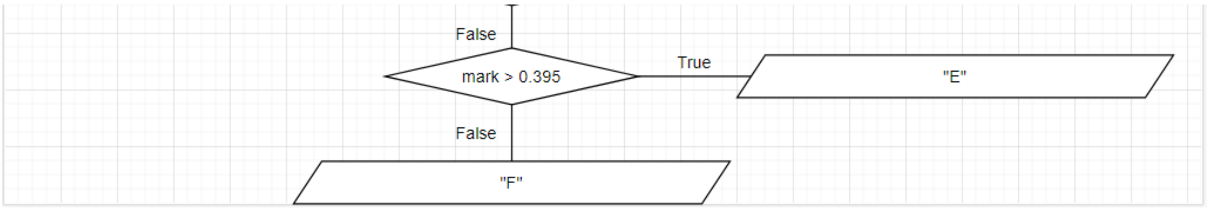
**Percentage Grade control flow graph:**


Start
Enum.count(homework) == 0
True
False
avg_homework = 0
avg_homework = Enum.sum(homework) / Enum.count(homework)
Enum.count(labs) == 0
True
False
avg_labs = 0
avg_labs = Enum.sum(labs) / Enum.count(labs)
mark = 0.2 * avg_labs + 0.3 * avg_homework + 0.2 * midterm + 0.3 * final
round(mark * 100)

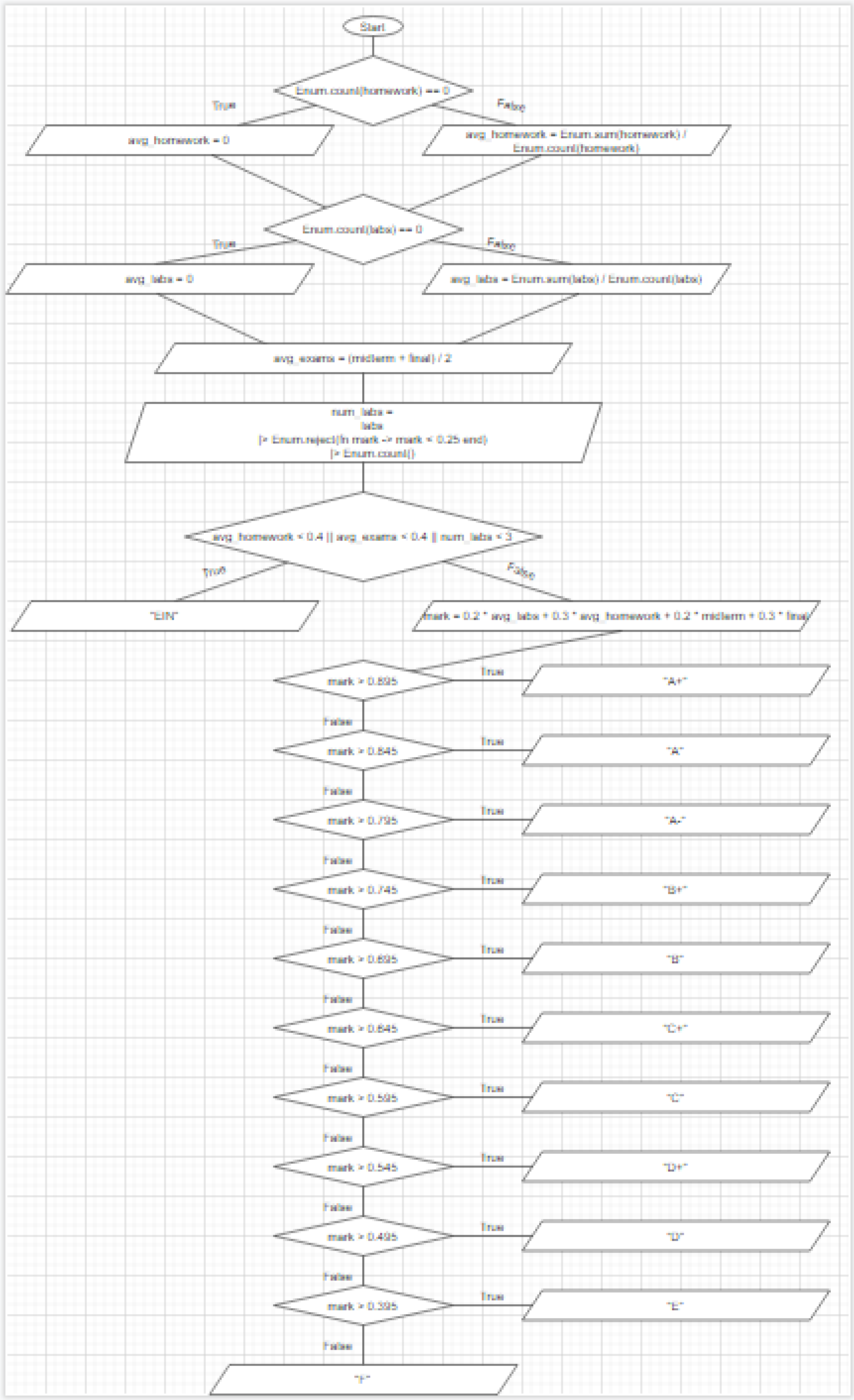
**Letter Grade control flow graph:**


Start
Enum.count(homework) == 0
True
False
avg_homework = 0
avg_homework = Enum.sum(homework) / Enum.count(homework)
Enum.count(labs) == 0
True
False
avg_labs = 0
avg_labs = Enum.sum(labs) / Enum.count(labs)
avg_exams = (midterm + final) / 2
num_labs = labs |> Enum.reject(fn mark -> mark < 0.25 end) |> Enum.count()
avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3
True
False
"EIN"
mark = 0.2 * avg_labs + 0.3 * avg_homework + 0.2 * midterm + 0.3 * final

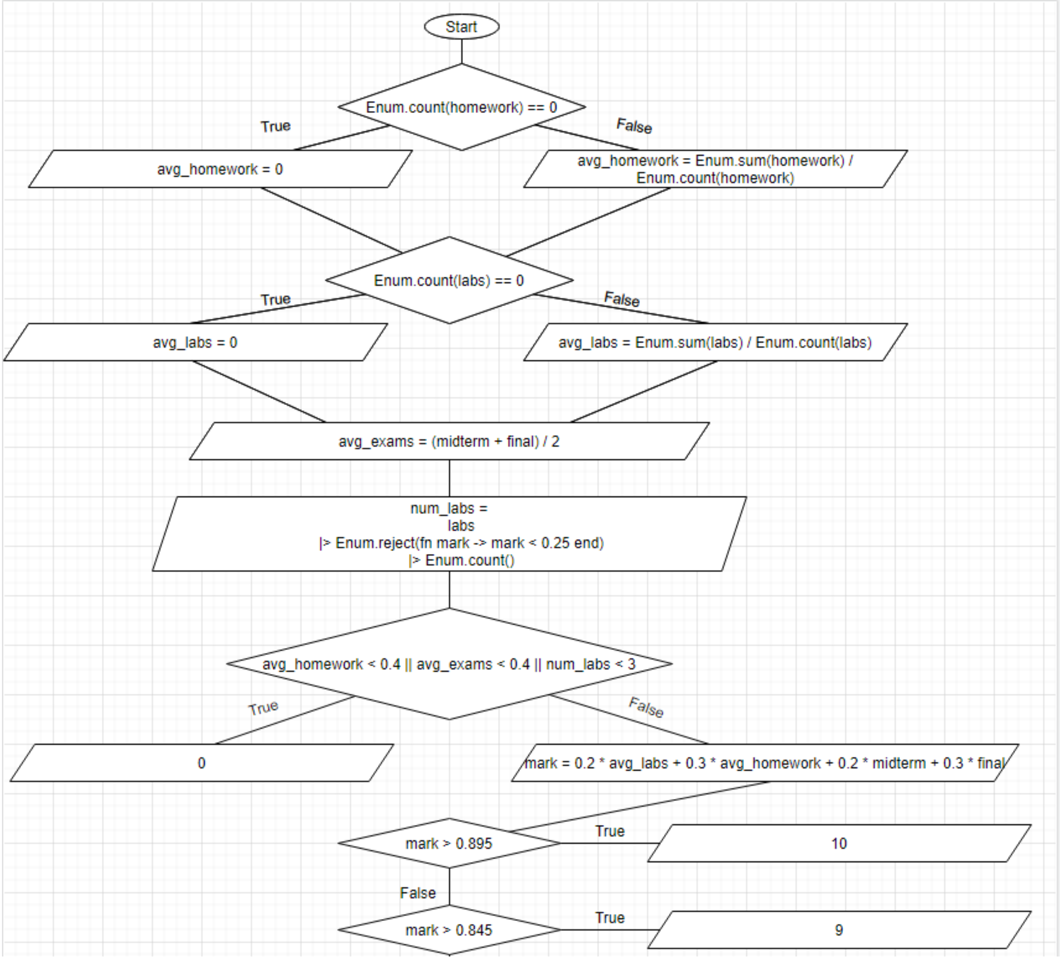


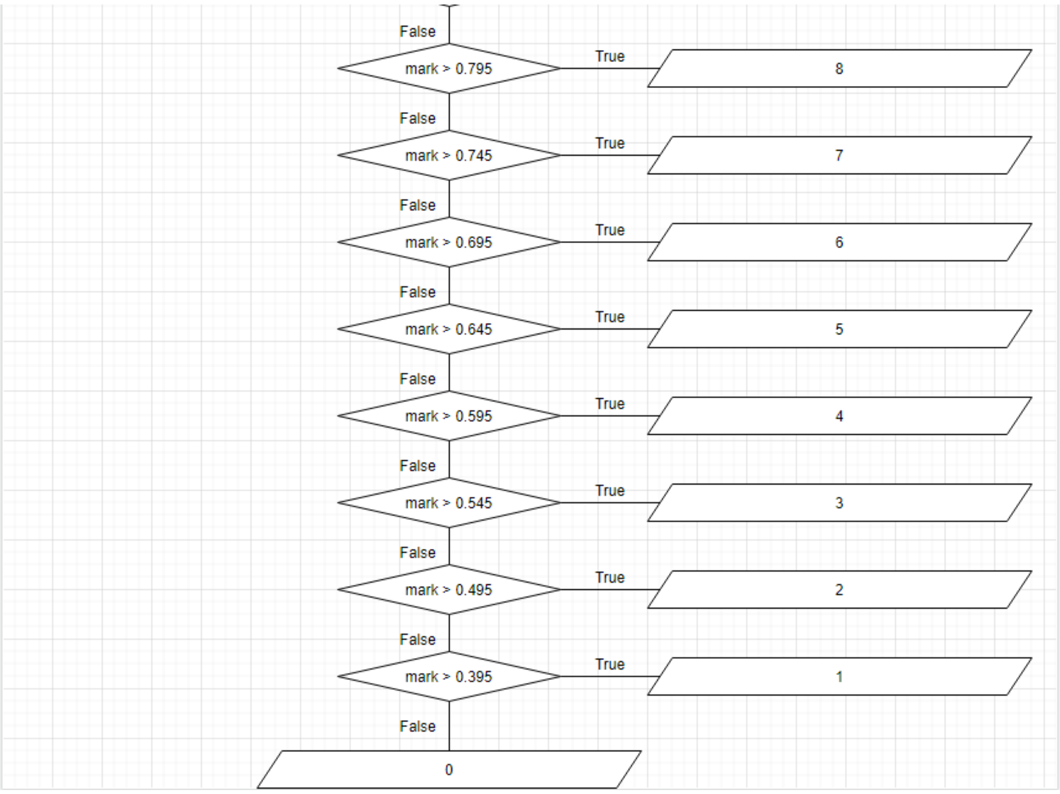


**Note: I separated the graph into 3 pictures and tried to align them since putting them in one picture would make it hard to read the details. But I also included the uncut picture after this text just in case.**

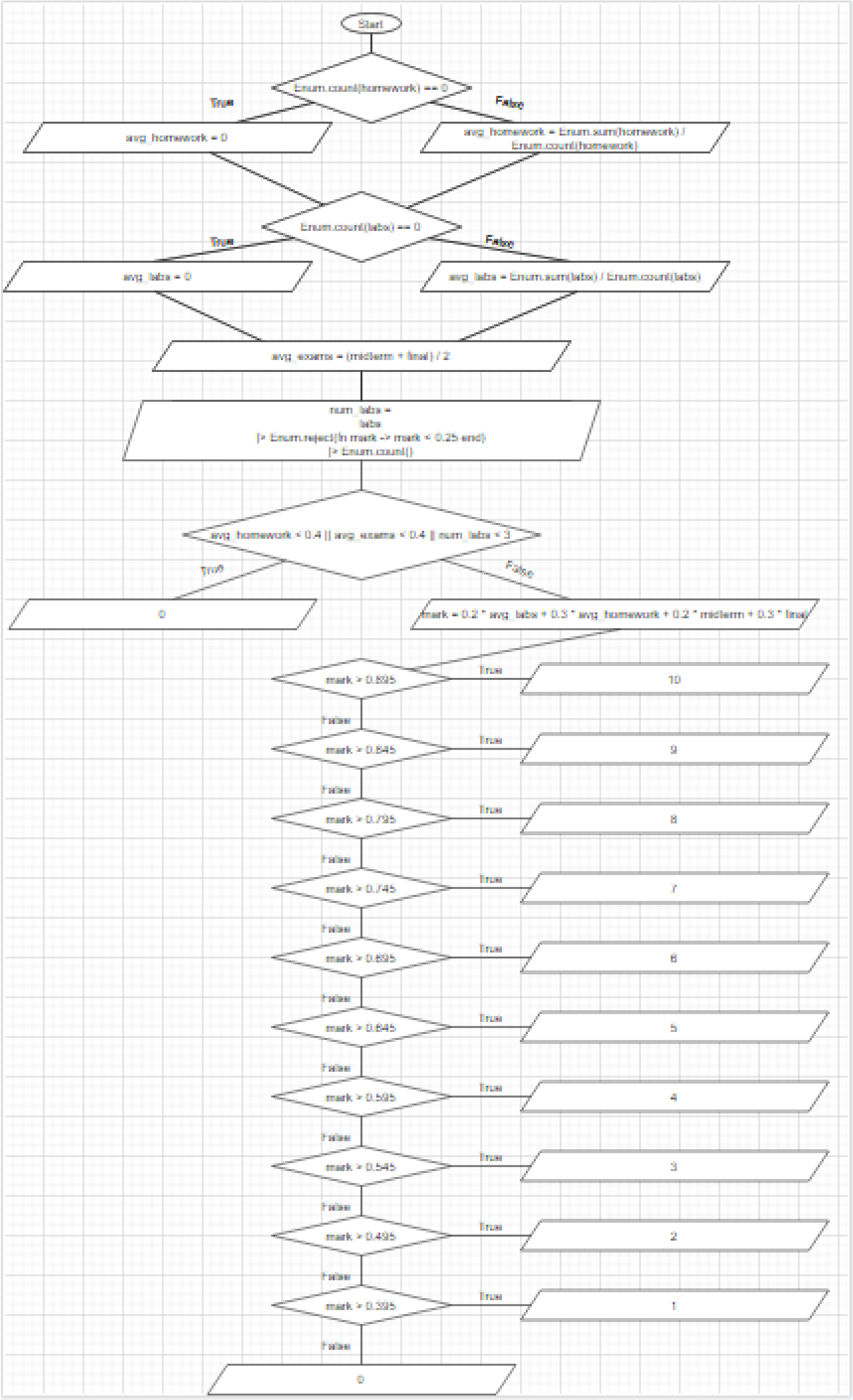


**Numeric Grade control flow graph:**


Start
Enum.count(homework) == 0
True
False
avg_homework = 0
avg_homework = Enum.sum(homework) / Enum.count(homework)
Enum.count(labs) == 0
True
False
avg_labs = 0
avg_labs = Enum.sum(labs) / Enum.count(labs)
avg_exams = (midterm + final) / 2
num_labs =
labs
|> Enum.reject(fn mark -> mark < 0.25 end)
|> Enum.count()
avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3
True
False
0
mark = 0.2 * avg_labs + 0.3 * avg_homework + 0.2 * midterm + 0.3 * final
mark > 0.895
True
10
False
mark > 0.845
True
9


False
mark > 0.795
True
8
False
mark > 0.745
True
7
False
mark > 0.695
True
6
False
mark > 0.645
True
5
False
mark > 0.595
True
4
False
mark > 0.545
True
3
False
mark > 0.495
True
2
False
mark > 0.395
True
1
False
0

**Note: I separated the graph into 2 pictures and tried to align them since putting them in one picture would make it hard to read the details. But I also included the uncut picture after this text just in case.**



Question 1.2 - **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:**

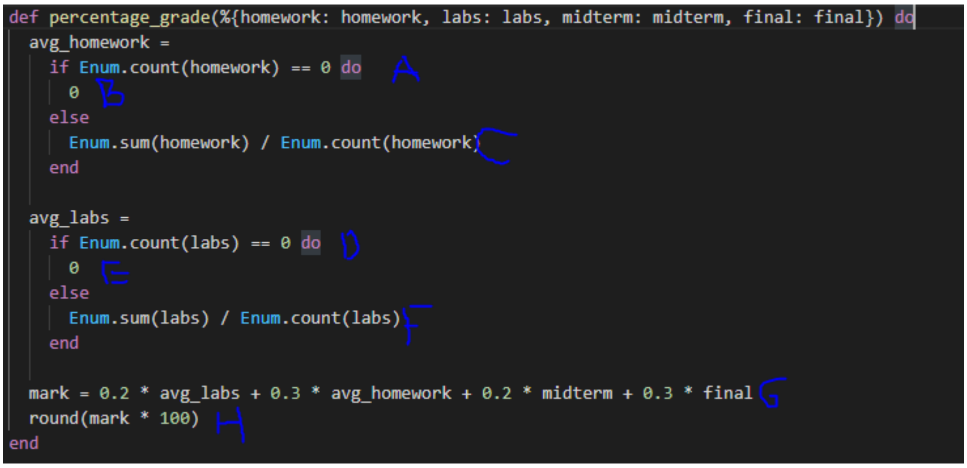
| Test Case | Test Data | Expected Results | Conditions Covered | Branches Covered |
|---|---|---|---|---|
| 1(percentage_grade) | homework: []<br>labs: []<br>midterm: 0<br>final: 0 | 0 | AD | ABDEGH |
| 2(percentage_grade) | homework: [0.5]<br>labs: [0.5]<br>midterm: 0.5<br>final: 0.5 | 50 | !A !D | ACDFGH |
| 1 - (letter_grade) | homework: []<br>labs: []<br>midterm: 0<br>final: 0 | EIN | ADI | ABDEGHIJ |
| 2 - (letter_grade) | homework: [1,1,1]<br>labs: [1,1,1]<br>midterm: 1<br>final: 1 | A+ | !A !D !I<br>1 | ACDFGHIK<br>1 |
| 3 - (letter_grade) | homework: [0.85,0.85,0.85]<br>labs: [0.85,0.85,0.85]<br>midterm: 0.85<br>final: 0.85 | A | !A !D !I<br>!1<br>2 | ACDFGHIK<br>1 2 |
| 4 - (letter_grade) | homework: [0.8,0.8,0.8]<br>labs: [0.8,0.8,0.8]<br>midterm: 0.8<br>final: 0.8 | A- | !A !D !I<br>!1 !2<br>3 | ACDFGHIK<br>1 2 3 |
| 5 - (letter_grade) | homework: [0.75,0.75,0.75]<br>labs: [0.75,0.75,0.75]<br>midterm: 0.75<br>final: 0.75 | B+ | !A !D !I<br>!1 !2 !3<br>4 | ACDFGHIK<br>1 2 3 4 |
| 6 - (letter_grade) | homework: [0.7,0.7,0.7]<br>labs: [0.7,0.7,0.7]<br>midterm: 0.7<br>final: 0.7 | B | !A !D !I<br>!1 !2 !3 !4<br>5 | ACDFGHIK<br>1 2 3 4 5 |
| 7 - (letter_grade) | homework: [0.65,0.65,0.65]<br>labs: [0.65,0.65,0.65]<br>midterm: 0.65<br>final: 0.65 | C+ | !A !D !I<br>!1 !2 !3 !4 !5<br>6 | ACDFGHIK<br>1 2 3 4 5 6 |

| | | | | |
|---|---|---|---|---|
| 8 - (letter_grade) | homework: [0.6,0.6,0.6]<br>labs: [0.6,0.6,0.6]<br>midterm: 0.6<br>final: 0.6 | C | !A !D !I<br>!1 !2 !3 !4 !5 !6<br>7 | ACDFGHIK<br>1 2 3 4 5 6 7 |
| 9 - (letter_grade) | homework: [0.55,0.55,0.55]<br>labs: [0.55,0.55,0.55]<br>midterm: 0.55<br>final: 0.55 | D+ | !A !D !I<br>!1 !2 !3 !4 !5 !6 !7<br>8 | ACDFGHIK<br>1 2 3 4 5 6 7 8 |
| 10 - (letter_grade) | homework: [0.5,0.5,0.5]<br>labs: [0.5,0.5,0.5]<br>midterm: 0.5<br>final: 0.5 | D | !A !D !I<br>!1 !2 !3 !4 !5 !6 !7 !8<br>9 | ACDFGHIK<br>1 2 3 4 5 6 7 8 9 |
| 11 - (letter_grade) | homework: [0.45,0.45,0.45]<br>labs: [0.45,0.45,0.45]<br>midterm: 0.45<br>final: 0.45 | E | !A !D !I<br>!1 !2 !3 !4 !5 !6 !7 !8 !9<br>10 | ACDFGHIK<br>1 2 3 4 5 6 7 8 9 10 |
| 12 - (letter_grade) | homework: [0.41]<br>labs: [0.25,0.25,0.25]<br>midterm: 0.45<br>final: 0.45 | F | !A !D !I<br>!1 !2 !3 !4 !5 !6 !7 !8 !9 !10<br>11 | ACDFGHIK<br>1 2 3 4 5 6 7 8 9 10<br>11 |
| 1 - (numeric grade) | homework: []<br>labs: []<br>midterm: 0<br>final: 0 | 0 | ADI | ABDEGHIJ |
| 2 - (numeric grade) | homework: [1,1,1]<br>labs: [1,1,1]<br>midterm: 1<br>final: 1 | 10 | !A !D !I<br>1 | ACDFGHIK<br>1 |
| 3 - (numeric grade) | homework: [0.85,0.85,0.85]<br>labs: [0.85,0.85,0.85]<br>midterm: 0.85<br>final: 0.85 | 9 | !A !D !I<br>!1<br>2 | ACDFGHIK<br>1 2 |
| 4 - (numeric grade) | homework: [0.8,0.8,0.8]<br>labs: [0.8,0.8,0.8]<br>midterm: 0.8<br>final: 0.8 | 8 | !A !D !I<br>!1 !2<br>3 | ACDFGHIK<br>1 2 3 |
| 5 - (numeric grade) | homework: [0.75,0.75,0.75]<br>labs: [0.75,0.75,0.75]<br>midterm: 0.75<br>final: 0.75 | 7 | !A !D !I<br>!1 !2 !3<br>4 | ACDFGHIK<br>1 2 3 4 |
| 6 - (numeric grade) | homework: [0.7,0.7,0.7]<br>labs: [0.7,0.7,0.7]<br>midterm: 0.7<br>final: 0.7 | 6 | !A !D !I<br>!1 !2 !3 !4<br>5 | ACDFGHIK<br>1 2 3 4 5 |

| | | | | |
|---|---|---|---|---|
| 7 - (numeric grade) | homework: [0.65,0.65,0.65]<br>labs: [0.65,0.65,0.65]<br>midterm: 0.65<br>final: 0.65 | 5 | !A !D !I<br>!1 !2 !3 !4 !5 6 | ACDFGHIK<br>1 2 3 4 5 6 |
| 8 - (numeric grade) | homework: [0.6,0.6,0.6]<br>labs: [0.6,0.6,0.6]<br>midterm: 0.6<br>final: 0.6 | 4 | !A !D !I<br>!1 !2 !3 !4 !5 !6 7 | ACDFGHIK<br>1 2 3 4 5 6 7 |
| 9 - (numeric grade) | homework: [0.55,0.55,0.55]<br>labs: [0.55,0.55,0.55]<br>midterm: 0.55<br>final: 0.55 | 3 | !A !D !I<br>!1 !2 !3 !4 !5 !6 !7 8 | ACDFGHIK<br>1 2 3 4 5 6 7 8 |
| 10 - (numeric grade) | homework: [0.5,0.5,0.5]<br>labs: [0.5,0.5,0.5]<br>midterm: 0.5<br>final: 0.5 | 2 | !A !D !I<br>!1 !2 !3 !4 !5 !6 !7 !8 9 | ACDFGHIK<br>1 2 3 4 5 6 7 8 9 |
| 11 - (numeric grade) | homework: [0.45,0.45,0.45]<br>labs: [0.45,0.45,0.45]<br>midterm: 0.45<br>final: 0.45 | 1 | !A !D !I<br>!1 !2 !3 !4 !5 !6 !7 !8 !9 10 | ACDFGHIK<br>1 2 3 4 5 6 7 8 9 10 |
| 12 - (numeric grade) | homework: [0.4]<br>labs: [0.25,0.25,0.25]<br>midterm: 0.45<br>final: 0.45 | 0 | !A !D !I<br>!1 !2 !3 !4 !5 !6 !7 !8 !9 !10 11 | ACDFGHIK<br>1 2 3 4 5 6 7 8 9 10 11 |

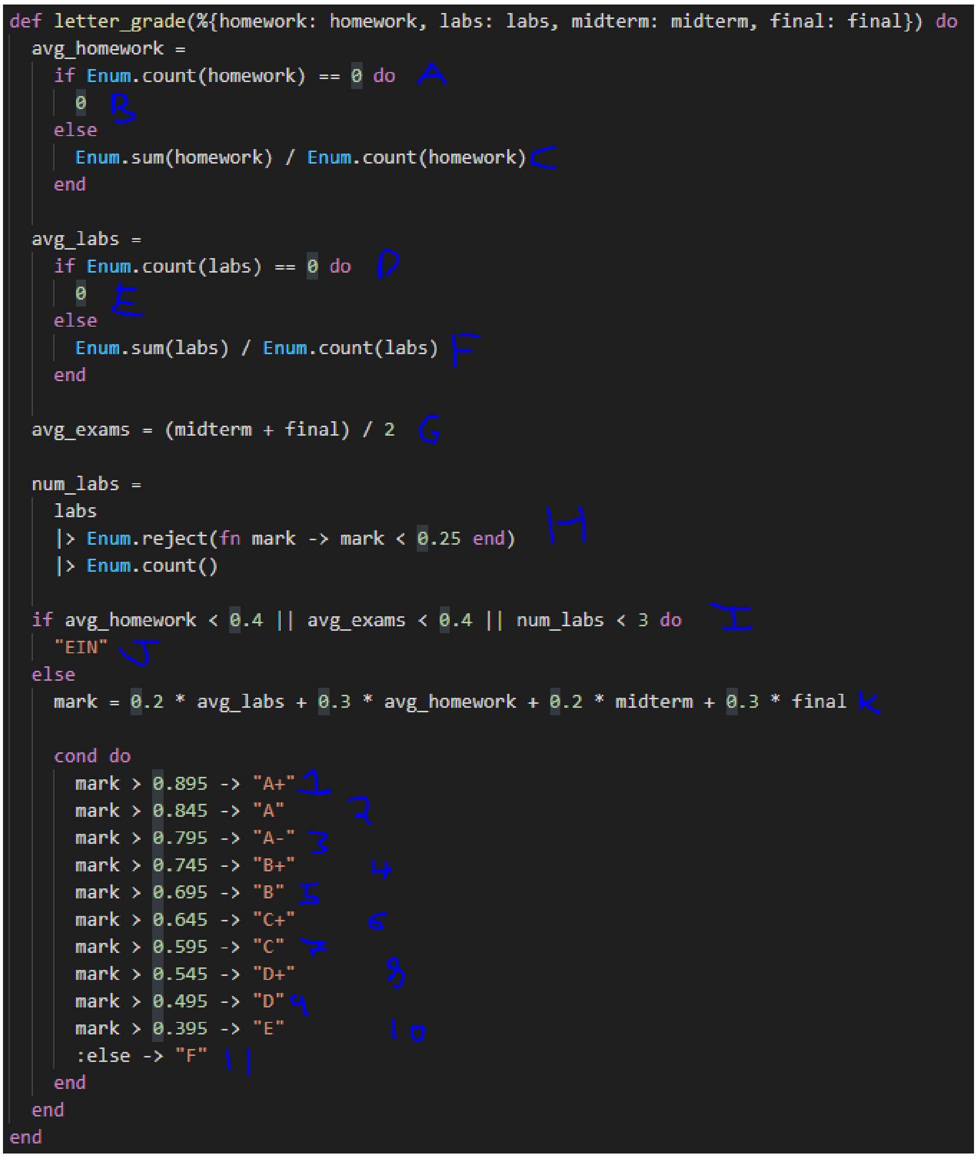
**Note: The annotations listed in the table are as follows:**

## Percentage Grade Test Case Annotations:



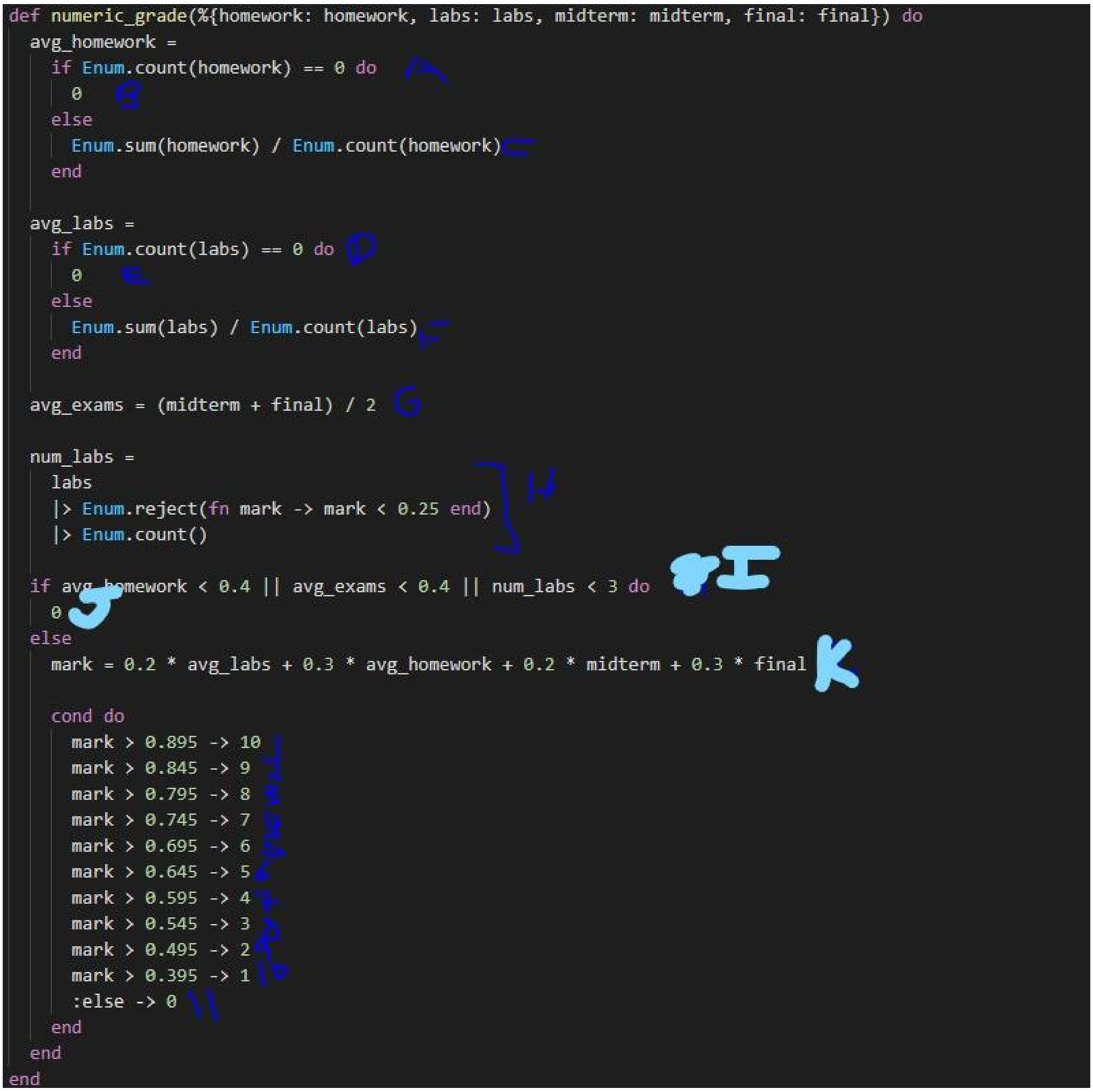
```
def percentage_grade(%{homework: homework, labs: labs, midterm: midterm, final: final}) do
  avg_homework =
    if Enum.count(homework) == 0 do
      0
    else
      Enum.sum(homework) / Enum.count(homework)
    end

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

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

## Letter Grade Test Case Annotations:



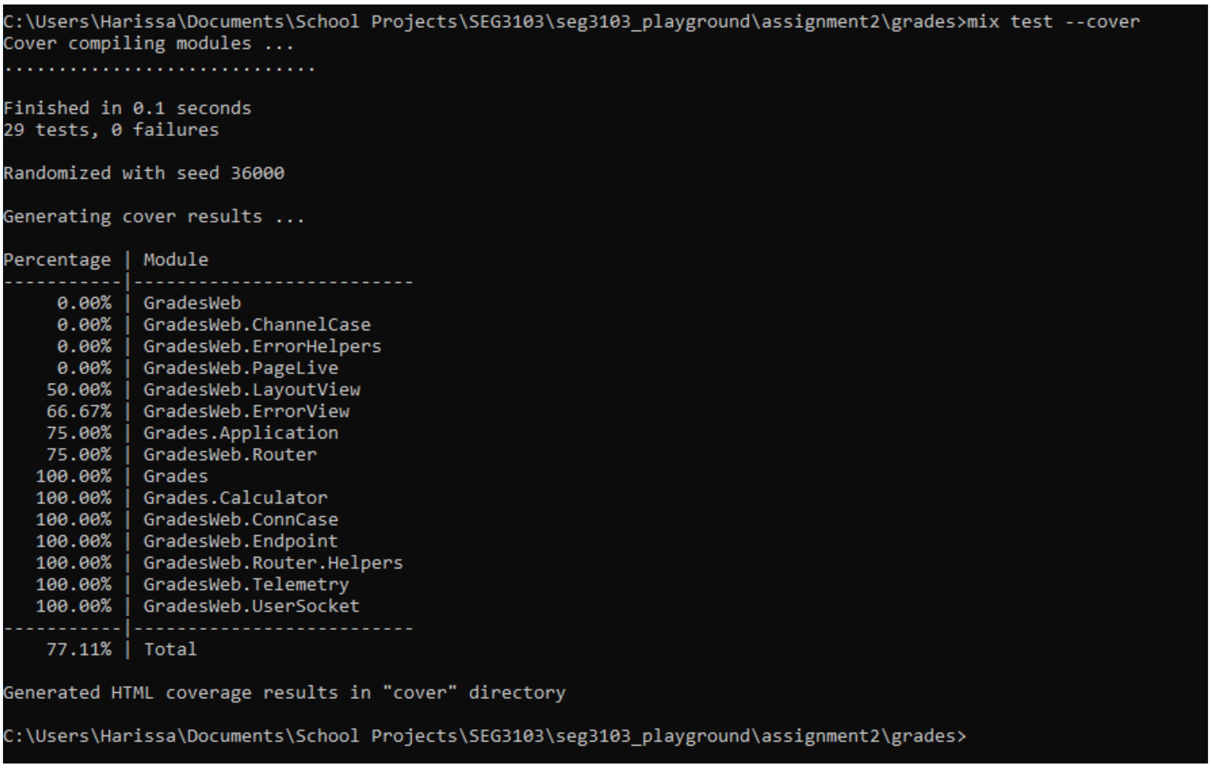
```
def letter_grade(%{homework: homework, labs: labs, midterm: midterm, final: final}) do
  avg_homework =
    if Enum.count(homework) == 0 do
      0
    else
      Enum.sum(homework) / Enum.count(homework)
    end

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

  avg_exams = (midterm + final) / 2

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

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

    cond do
      mark > 0.895 -> "A+"
      mark > 0.845 -> "A"
      mark > 0.795 -> "A-"
      mark > 0.745 -> "B+"
      mark > 0.695 -> "B"
      mark > 0.645 -> "C+"
      mark > 0.595 -> "C"
      mark > 0.545 -> "D+"
      mark > 0.495 -> "D"
      mark > 0.395 -> "E"
      :else -> "F"
    end
  end
end
```

**Numeric Grade Test Case Annotations:**



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

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

  avg_exams = (midterm + final) / 2

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

  if avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 do
    0
  else
    mark = 0.2 * avg_labs + 0.3 * avg_homework + 0.2 * midterm + 0.3 * final

    cond do
      mark > 0.895 -> 10
      mark > 0.845 -> 9
      mark > 0.795 -> 8
      mark > 0.745 -> 7
      mark > 0.695 -> 6
      mark > 0.645 -> 5
      mark > 0.595 -> 4
      mark > 0.545 -> 3
      mark > 0.495 -> 2
      mark > 0.395 -> 1
      :else -> 0
    end
  end
end
```

Question 1.4 - **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?**



```
C:\Users\Harissa\Documents\School Projects\SEG3103\seg3103_playground\assignment2\grades>mix test --cover
Cover compiling modules ...
.............................

Finished in 0.1 seconds
29 tests, 0 failures

Randomized with seed 36000

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

C:\Users\Harissa\Documents\School Projects\SEG3103\seg3103_playground\assignment2\grades>
```

As can be seen in the screenshot above, I did achieve a statement coverage of 100% for the grades.calculator class.

The problem with this testing tool is that it doesnt provide the branch coverage of my tests. For example. I know for a fact that I do not have 100% test case coverage for the following statement in the numeric and letter grade methods:

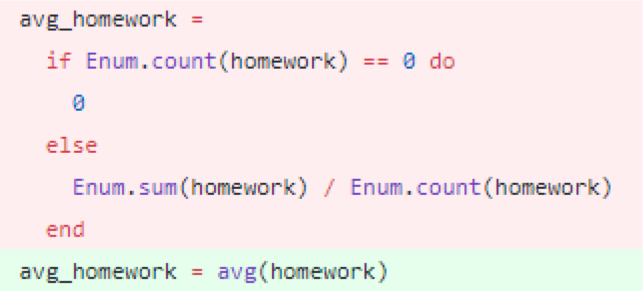
**avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3**

However, this is not indicated in the testing tool and only shows that I got 100% coverage. If this were a real program there could still be some bugs or unexpected behaviors that would pass to the consumers due to the uncovered branches.

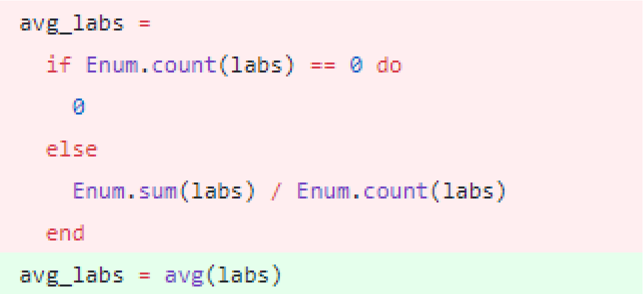
Please note that the reason I didn't cover the branch 100% is that the question only asked to get 100% **statement** coverage in the least amount of test cases possible which means adding test cases to get 100% branch coverage might lead to a lower grade.

**Question 2.1 - Extract a helper method avg to clean up the duplicate code like?**

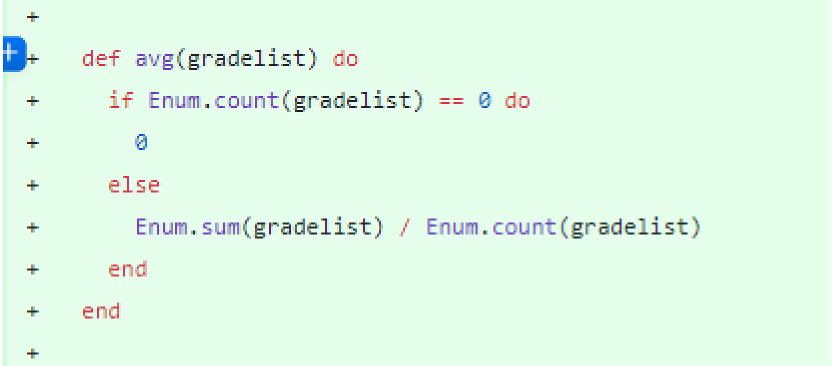
**Note: Red code means its been removed and green code means its been added**



```
avg_homework =
  if Enum.count(homework) == 0 do
    0
  else
    Enum.sum(homework) / Enum.count(homework)
  end
avg_homework = avg(homework)
```



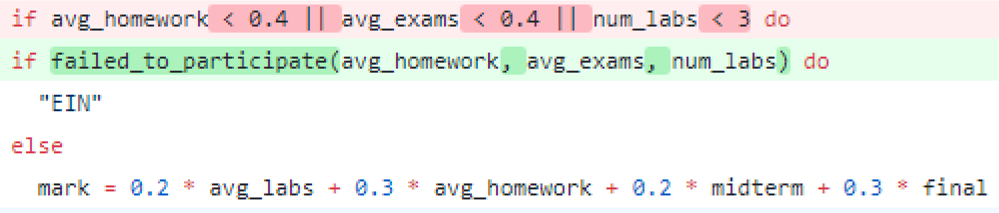
```
avg_labs =
  if Enum.count(labs) == 0 do
    0
  else
    Enum.sum(labs) / Enum.count(labs)
  end
avg_labs = avg(labs)
```



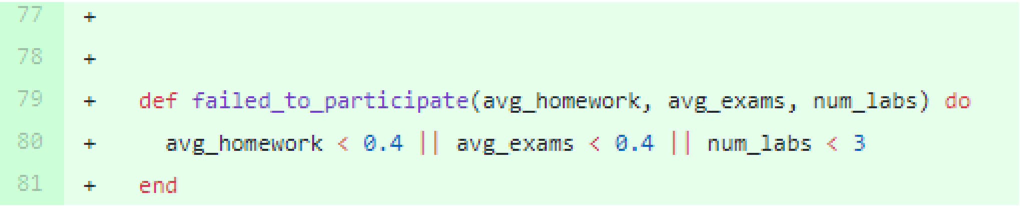
```
+
+   def avg(gradelist) do
+     if Enum.count(gradelist) == 0 do
+       0
+     else
+       Enum.sum(gradelist) / Enum.count(gradelist)
+     end
+   end
+
```

To refactor this I made a helper method to remove all average calculating code in all three methods that have been duplicated

Question 2.1 - **Extract a helper method failed_to_participate? to clean up duplicate code like**



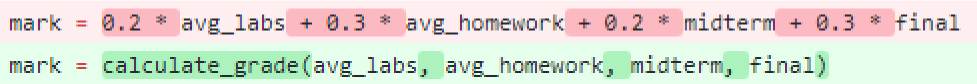
```
if avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3 do
if failed_to_participate(avg_homework, avg_exams, num_labs) do
  "EIN"
else
  mark = 0.2 * avg_labs + 0.3 * avg_homework + 0.2 * midterm + 0.3 * final
```



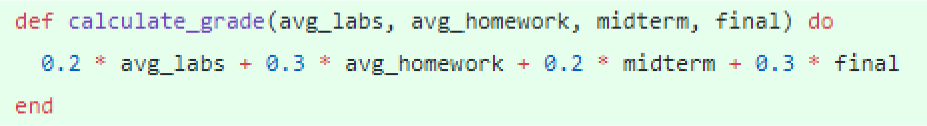
```
+
+
+   def failed_to_participate(avg_homework, avg_exams, num_labs) do
+     avg_homework < 0.4 || avg_exams < 0.4 || num_labs < 3
+   end
```

To refactor this I made a helper method to remove all participation checking code in the numeric and letter grade methods that has been duplicated.

Question 2.3 - **Extract a helper method calculate_grade to clean up duplicate code like**



```
mark = 0.2 * avg_labs + 0.3 * avg_homework + 0.2 * midterm + 0.3 * final
mark = calculate_grade(avg_labs, avg_homework, midterm, final)
```

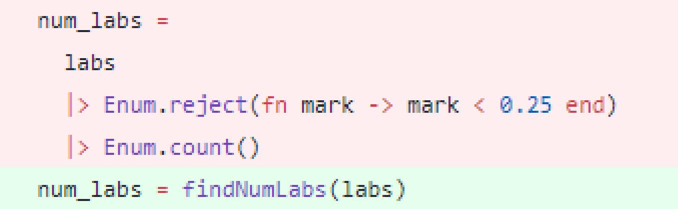


```
def calculate_grade(avg_labs, avg_homework, midterm, final) do
  0.2 * avg_labs + 0.3 * avg_homework + 0.2 * midterm + 0.3 * final
end
```

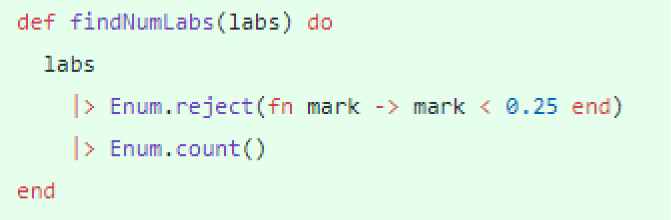
To refactor this I made a helper method to remove all final mark calculating code in all 3 of the grade methods that have been duplicated.

Question 2.4 - Provide at least 2 additional refactoring to the code. Your refactoring should not require additional testing, however if you encounter any bugs in the original code then please fix them separately (ensuring your tests continue to pass) before continuing to refactor.

**Part 1:**



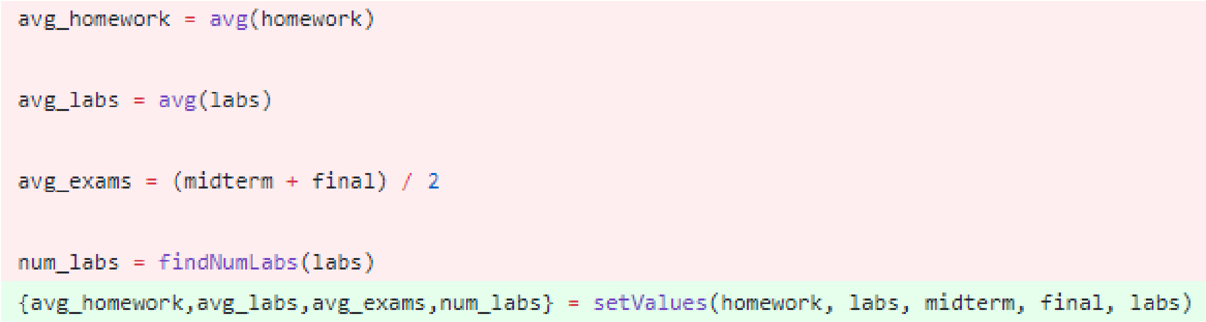
```
  num_labs =
    labs
    |> Enum.reject(fn mark -> mark < 0.25 end)
    |> Enum.count()
  num_labs = findNumLabs(labs)
```



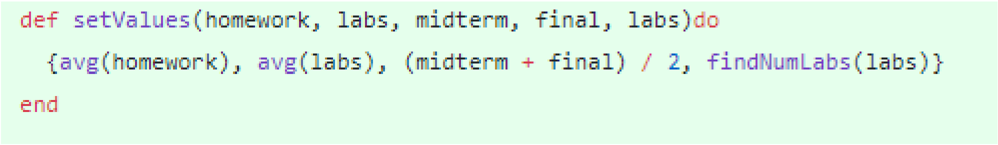
```
def findNumLabs(labs) do
  labs
    |> Enum.reject(fn mark -> mark < 0.25 end)
    |> Enum.count()
end
```

I noticed that we use this method of filtering the labs twice in both the numeric and letter grade methods. I created a helper method to keep the duplicated code in one place.

**Part 2:**



```
avg_homework = avg(homework)

avg_labs = avg(labs)

avg_exams = (midterm + final) / 2

num_labs = findNumLabs(labs)
{avg_homework,avg_labs,avg_exams,num_labs} = setValues(homework, labs, midterm, final, labs)
```



```
def setValues(homework, labs, midterm, final, labs)do
  {avg(homework), avg(labs), (midterm + final) / 2, findNumLabs(labs)}
end
```

I noticed that in both the numeric and letter grade methods we set the same variables and call the same methods in the same exact manner. Therefore I simplified this by creating a helper method that set all values at once and removed duplicate code.

**To see the commit history of these refactors please check:**[https://github.com/YoucefBenAli/seg3103_playground/tree/master/assignment2/grades](https://github.com/YoucefBenAli/seg3103_playground/tree/master/assignment2/grades)