Week 6 Assignment Specification
Module 2 – Array & Pointer Revision, File Input & Output, User Defined Header Files

---

This document contains the specification for all the assignments of the week.
Each question found here should have the following structure

- Question Preamble or Context (for application questions)
- Your Programming Task
- Input format
- Input constraints (i.e. the range of possible values of inputs to your program)
- Output format
- Starter Files (and how you can find them)
- Test Cases (at least one per question)

Because making a stress test or a test suite for each of the code is really too tedious, submit to me all the work you have completed and I will help you test and review your code.

There are **6** programming assignments for this week.
You are required to complete at least **2** out of **6** assignments for this week.

2 of these assignments highlighted in yellow, you are encouraged to do these first and you can complete the rest of the assignments as you revise in your own time.

1 of these assignments are highlighted in blue, you are encouraged to attempt these as a form of revision, and time yourself as you attempt the question.

Table of Contents

FOREWORD
Your programming assignments are getting longer, more demanding in terms of assignment specifications, so this would be the new format of your programming assignments. Remember to not shy away from Google, Stack Overflow, W3Resource, TutorialsPoint, Programiz.com and others– these websites are there to help you if you need specific targeted syntax as it is not efficient to read through the 100 slides just to find one specific thing. You will encounter issues along the way, don't give up, and I'll be one text away!

---

1 Average Weight for a Population of Elephant Seals

Adapted from Coursera's MOOC: C For Everyone Course 1 of 4
Programming Fundamentals – Week 5

Preamble
Some text files are long – and most of the time you will have no clue about the length of the data within the file.

Your Task:
A **text file** of unknown length is used to store the weights of elephant seals. These weights have already been rounded to the nearest integer. **By reading the data from the file into the array,** find the average weight for a population of elephant seals.

Hint:
If you don't know the size of the text file, you won't know what size the array should be. For this reason, you should not create an array first? How would you write a routine that somehow allows you to count the number of words within the text file?

File Input Format:
The file would consists of integers only, these files would also have unknown length. You can find these files provided in your starting folder.

Constraints:
The weight of an elephant would not exceed 10^5 kg.
The length of the text file can range from a length of 1 integer to a length of 10^6 integers.
Output Format:
Output just 1 line, a floating point number, which is the average weight of elephant seals calculated.

Starter File:
`q1/1_avgweight.c`
Using this starter code is optional.
This code file implements the input and output formats as specified above, and passes the input to an empty function which you have to implement. The code file would then take the result of your implementation and print it to the console in the specified format.

If you have a different idea in mind, feel free to create a new code file for this question.

[Turn Over for test cases]

TEST CASES FOR Q1

Test Case 1

Input file:

```
testcase1.txt
```

Output:

```
1372.900
```

Test Case 2

Input file:

```
testcase2.txt
```

Output:

```
1350.600
```

Test Case 3

Input file:

```
testcase3.txt
```

Output:

```
1599.521
```

END OF Q1

2 Classifying Elephant Seals

Note: Only Q1 is from Coursera, I am on a roll now,
so I came up with more applications for you to attempt!

Prerequisites
You will have to complete Question 4 before attempting this question.

Preamble
The average weight of an elephant seal that you derived in Question 1 seems to be erroneous. It turns out there were both male and female data mixed together in all of the datasets provided to you in Q1.

According to general statistics, there is a pretty distinct discrepancy between the weight of a male elephant seal and a female elephant seal.

- Male elephant seal weight should fall around 1500 – 2300 kg range.
- Female elephant seal weight should fall around 400 – 900 kg range.

For this question only, we assume that weights which do not fall within either of these ranges to be erroneous data.

Your Task:
In this question, write a program in C, to find the number of male elephant seals, female elephant seals, and erroneous data given an integer dataset. **You need not use an array this time.**

Input Format:
An input text file that is filled with integer weights in kilograms.

Constraints:
The weight of an elephant would not exceed 10^5 kg.
The length of the text file can range from a length of 1 integer to a length of 10^6 integers.
In addition, we assume that weights which do not fall within the ranges above to be erroneous.

Output Format:
Output 3 integers separated by a space, the number of males followed by the number of females followed by the number of erroneous data.

Starter File:
`q2/2_classification.c`
Using this starter code is optional.
This code file implements the input and output formats as specified above, and passes the input to an empty function which you have to implement. The code file would then take the result of your implementation and print it to the console in the specified format.

If you have a different idea in mind, feel free to create a new code file for this question.

[Turn Over for test cases]

TEST CASES FOR Q2

Test Case 1

Input

```
testcase1.txt
```

Output:

```
28 20 32
```

Explanation:

28 males, 20 females, and 32 erroneous data.

Test Case 2

Input:

```
testcase2.txt
```

Output:

```
15026 9279 18696
```

Explanation:

15026 males, 9279 females, and 18696 erroneous data

Test Case 3

Input:

```
testcase3.txt
```

Output:

```
285587 178732 535680
```

~~Explanation~~ Confession:

You must be wondering what happened, and why there are so many erroneous data, the truth is I didn't take this data from any source, this is just a series of random integers within a broad range of values, and this entire application story is all a ruse. As an aspiring data scientist, this is considered inexcusable and unforgivable. I am trying to manage your expectations here, so your coding practical dataset surprises you.

END OF Q2

3 Bowling Score Calculator

Preamble
The goal of bowling is to knock down as many pins as possible. The more pins you knock down, the more points you score.

A single game consists of 10 "frames", with each frame consisting of two chances to knock down all 10 pins (with the exception of the last frame). Each pin you knock down earns you a point, while you can also score extra points by hitting "strikes" or "spares".

Strikes and Spares
A **strike** is when you knock down all ten pins at your first attempt in a single frame. A strike on the score sheet is commonly indicated with an 'X'.
A **spare** is when you fail to knock down all ten pins at your first attempt in a single frame, but manage to clear the remaining pins at your second attempt. A spare is commonly indicated with a '/'.
A **double** is when you get two strikes in consecutive frames.
Strikes and Spares are scored slightly different from regular frames:

- **Spare:** 10 points + the number of pins you knock down for your **first attempt** on your next frame.
- **Strike:** 10 points + the number of pins you knock down on **your next two attempts.** This means if you get a strike on your next frame, the points in the **frame after** would also be counted together with your current frame score.

In a game of bowling, a player obtains the following score:

1	2	3	4	5	6	7	8	9	10
X	9 /	5 /	7 2	X	X	X	9 -	8 /	9 / X
20	35	52	61	91	120	139	148	167	187

The numbers in the top row show the breakdown of the shots, where the first number would be the first attempt, and the second number would be the second attempt. If the player gets a strike, he only gets 1 shot for the frame, and he does not get a second shot. The cross ('X') is written on the second space. The numbers below show the total score of the player.

In frame 1, the player gets a strike which gives him 10 points. This is because he gets a strike in the first frame – this would mean his next two attempts would be counted together in the score for frame 1. Hence, 10 + (9 + 1) = 20

In frame 2, he gets a spare, he gets 10 points. In the next frame, he gets a score of 5 on his first attempt, which brings his score for that frame 15, and his total would be 35.

In frame 6, he gets a strike, which scores him 10 points. Frame 7 is also a strike, which gives him another 10 points. His first shot in frame 8 is 9 points, which means he has a raw score of 10+10+9 = 29. This brings his total to 91 + 29 = 120.

Frame 10 is a special frame. There is no bonus in frame 10, every strike or spare in frame 10 is counted as 10, without bonus. (If a player gets a strike in frame 9, his first two shots in frame 10 would also be counted as bonus into his score in frame 9).
In **frame 10**, a player can potentially get 3 strikes. If a player does not manage to get a strike or a spare within his first two attempts in frame 10, the game ends. If he manages to get a spare or

strike within frame 10, there is 1 or 2 bonus attempts respectively within frame 10. In the above case, the player makes a spare, and he gets 1 bonus shot (which he strikes).

Your Task:
You will represent the player's breakdown, B, in the form of an integer array of size 21, where 21 is the maximum possible number of shots that can be made by any player. If a player doesn't make a shot (for instance a strike), the score on that attempt would be marked with the value 0.
For the example above, the player's breakdown B would look like this:

10	0	9	1	...	2	9	1	10

B

The first two indices (index 0 and 1) of the array B represent frame 1 (coloured blue) – in this frame a player obtained a strike, hence the value at the second attempt (index 1) is marked with a value 0.
The next two indices (index 1 and 2) of the array B represent frame 2 (coloured green) – in this frame a player obtained a spare.
The indices 18, 19 and 20 of the array B represent frame 10 (coloured orange) – in this frame a player obtained a spare in his first two attempts, followed by a strike on his 3rd attempt.

You would find the player's total score, T, in the form of an integer array of size 10, where 10 is the number of frames in a 10 frame game. The i-th value in the array T should be the total score for the player up to and including that frame, more formally $T[i] = S_i + T[i-1] + \cdots + T[0]$ for $i \in [0,9]$,
Where S_i is the raw score for frame i.
For the example above, if your implementation is correct, your array T should look like this:

20	35	52	61	...	139	148	167	187

T

where the i-th value in T is the total score for the frame $i+1$.

Write a program in C that takes input **from the console** in the form of a game breakdown, and outputs **to the console** the total score the player had in each frame.

Input Format:
The input consists of 10 lines. From line i where $i \in [1,9]$, line i represents the breakdown for frame i. Each line i would consist of a **maximum** of two integers separated by a space: which represent the score on the first attempt on frame i followed by the score on the second attempt on frame i. **If a player gets a strike on frame i, line i would only consist of 1 integer 10.**

For frame 10, if the player does not strike or spare on frame 10, line 10 would only consist of two space-separated integers. If a player gets at least a strike or a spare, there would be 3 integers on line 10.

For this reason, the input has varied length – where the shortest would be 11 integers long and the longest would be 21 integers long.

Constraints:
$0 \leq B[i] \in \mathbb{Z} \leq 10, 0 \leq T[j] \in \mathbb{Z} \leq 300, \forall\ 0 \leq i \in \mathbb{Z} \leq 20, 0 \leq j \leq 9$

Output Format:
Output 10 integers, separated by a space, where the i-th integer represents the total score for the player at that frame i.

Starter File:
`q3/3_bowling.c`
Using this starter code is optional, though you are highly encouraged to use this as a starting point. This code file has already implemented the output format, but **the input format is incomplete.** Currently, the input is implemented by a for loop and the program will require 21 integers exactly, **before it passes the array into the empty function which you have to implement.** You'll see that most inputs are **not** 21 digits long (see the test cases for more details).

If you have a different idea in mind, feel free to create a new code file for this question.

Hint:
Try to derive an expression in terms of the frame number that derives the **index** of the first shot and the index of the second shot in the array B.

TEST CASES FOR Q3
Test Case 1
Input
```
10
9 1
5 5
7 2
10
10
10
9 0
8 2
9 1 10
```
Output:
```
20 35 52 61 91 120 139 148 167 187
```
Explanation:
This test case is the same as the sample results by the player above in the preamble. In this case, the input is way shorter than 21 integers, because the player has made 4 strikes in the game that were not in frame 10.

Test Case 2
Input:
```
10
10
10
10
10
10
10
10
10
10 10 10
```
Output:
```
30 60 90 120 150 180 210 240 270 300
```
Explanation:
This is a perfect game, and the player would get a total score of 300.

Test Case 3

Input:

```
9 1
9 0
10
9 1
9 0
7 1
8 0
10
9 0
0 8
```

Output:

```
19 28 48 67 76 84 92 111 120 128
```

Explanation:

This test case is to check if you implemented your tenth-frame input output and functionality correctly.

Test Case 4

Input

```
0 10
0 10
0 10
0 10
0 10
0 10
0 10
0 10
10
0 10 0
```

Output:

```
10 20 30 40 50 60 70 90 110 120
```

Explanation:

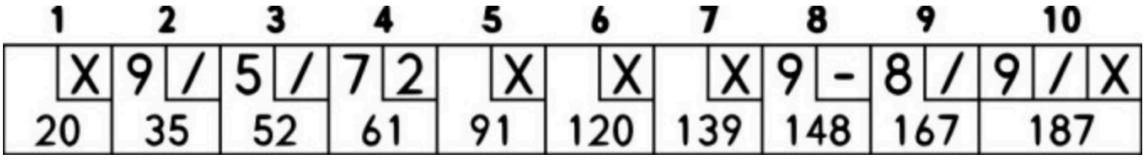
Throughout this game, this player is missing one shot and sparing all ten pins on the next. This is **not** considered a strike. This player only strikes at the 9^{th} frame, hence his score jumps by 20 from the 7^{th} to 8^{th}, and the 8^{th} to 9^{th} frame because of the bonus points.

END OF Q3

4 Bowling Score Sheet

Question Prerequisites
You need to complete Question 3 before attempting this question.

Preamble
Some bowlers like for their scores to be printed out on a sheet of paper. You are tasked to create a product that allows bowlers to print their bowling score breakdown and their total score after the completion of the game.



A strike is marked with a character `'X'`. A spare is marked with a character `'/'`.

Your Task:
In the function `createScoreSheet()` of the starter file, or otherwise, write a program in C that takes the player's shot breakdown in numbers as **input from the console** and creates a **scoresheet in a text file.** (See the output file format for more information)

You should modify your program In question 1 to suit the functionality in this question.

Input Format:
Same as Question 3.

Constraints:
Same constraints as Question 3.

Output File Format:
The output file name would be `q4output.txt` and it should contain 3 lines.
The first line should contain the numbers 1 to 10, **separated by tabs.**

The second line should contain the player's breakdown – and here are some specifics regarding how the breakdown should be implemented:

- If the player gets a strike, print the character `'X'` (without quotes) to the output file.
- If the player hits 9 pins on the first attempt, and a spare on the next attempt, print the characters `"9 /"` separated by a space (without quotes)
- If the player does not get a strike or spare, print the first attempt score, followed by the second attempt score, two integers separated by a space.
- If the player does not hit any pins, print a dash character `'-'` for the attempt that no pins were hit. **Note:** if the player gets a strike, there should be no dash characters after a strike.
- Breakdowns within the same frame should be separated by a space, breakdowns **between** frames should be **separated by tabs.**

The last line should contain the players total score, where the first integer should be the total score for frame 1, second integer should be the total score for frame 2, and so on. Total scores **between frames** should be **separated by tabs.**

Starter File:
`q4/4_scoresheet.c`
The starter file for this question is the same as Q3. Adapt your solution from Q3 and add the additional functionality to the function main. You should implement this printing process in the function `createScoreSheet()` that has already been declared for you in `4_scoresheet.c`.

TEST CASES FOR Q4

Test Case 1

Input

```
10
9 1
5 5
7 2
10
10
10
9 0
8 2
9 1 10
```

q2output.txt:

```
1     2     3     4     5     6     7     8     9     10
X     9 /   5 /   7 2   X     X     X     9 -   8 /   9 / X
20    35    52    61    91    120   139   148   167   187
```

Test Case 2

Input:

```
10
10
10
10
10
10
10
10
10
10 10 10
```

q2output.txt:

```
1     2     3     4     5     6     7     8     9     10
X     X     X     X     X     X     X     X     X     X X X
30    60    90    120   150   180   210   240   270   300
```

Test Case 3

Input:

```
9 1
9 0
10
9 1
9 0
7 1
8 0
10
9 0
0 8
```

q2output.txt:

```
1     2     3     4     5     6     7     8     9     10
9 /   9 -   X     9 /   9 -   7 1   8 -   X     9 -   - 8
19    28    48    67    76    84    92    111   120   128
```

Explanation:

Note that the last frame has no strike or spare. Hence the player had no bonus attempt. Hence there is no dash after the last 8 on the 2nd row.

Test Case 4

Input

```
0 10
0 10
0 10
0 10
0 10
0 10
0 10
0 10
10
0 10 0
```

q2output.txt

```
1     2     3     4     5     6     7     8     9     10
- /   - /   - /   - /   - /   - /   - /   - /   X     - / -
10    20    30    40    50    60    70    90    110   120
```

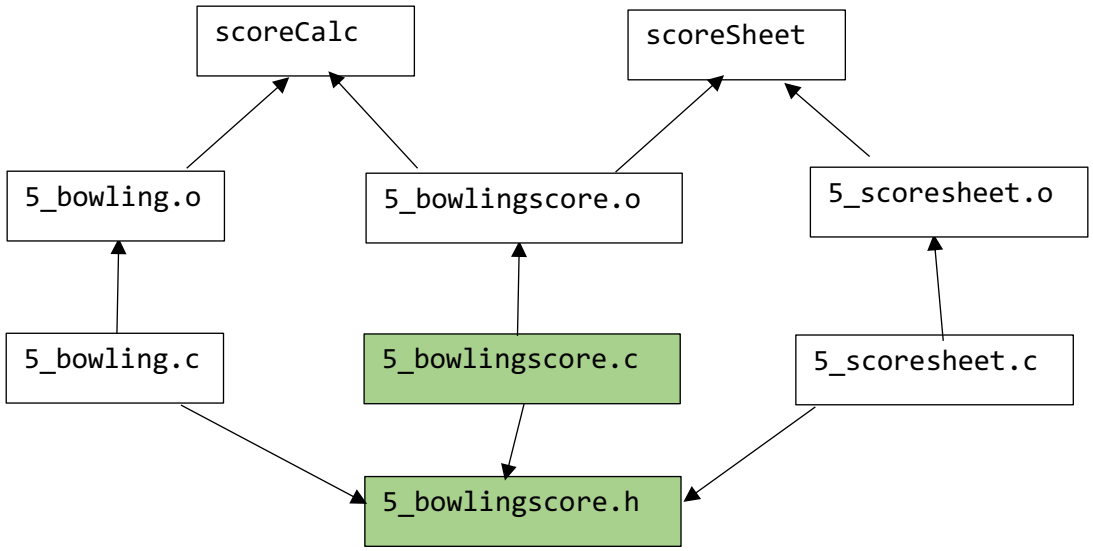
END OF Q4

5 A Bowling Application: Using User Defined Header Files

Question Prerequisites:
Complete Question 1 and Question 2 before attempting this question.
Copy the code in `3_bowling.c` into `5_bowling.c`
Copy the code in `4_scoresheet.c` into `5_scoresheet.c`

Preamble
Question 1 and 2 use the same function `calculateBowlingScores()` for implementation.
Writing the same function twice at two different areas of code is naïve implementation.

Your Task:
For this question, you are tasked to automate the building of the executables: `scoreCalc` and `scoreSheet`using Makefiles. Here is the dependency tree you will have to implement:



Source files marked in green are files that you are required to create.

Checklist:

- You will have to **create** the following files: `5_bowlingscore.c` and `5_bowlingscore.h`. Implement in these files program functionality that is shared between `5_bowling.c` and `5_scoresheet.c`. This shared functionality should only be implemented once in `5_bowlingscore.c`.

- You will have to **edit** the following files: `Makefile`. This `Makefile` is meant to automate the compilation of the executables `scoreCalc` and `scoreSheet`. This file is currently empty apart from the starting comment. Program the makefile for the following commands:

Command	What it Does
`make`	Creates the executable `scoreSheet`.
`make bowlingCalc`	Creates the executable `scoreCalc`
`make clean`	Removes the object files **if they exist.**
`make reset`	Removes the object files **and** executables **if they exist.**

END OF Q5

6 Hypothesis Testing using Z-Tests

Assume, for the purposes of this question, that the datasets that we have are accurate.

Prerequisites

You should complete Question 1 and Question 2 before attempting this question. In addition, please have at least 1 piece of foolscap paper with you and a timer.

Time Allowed: 50 minutes

Preamble

Hypothesis testing is an integral part in Data Science. It is an act in statistics whereby an analyst tests an assumption regarding a population parameter. Hypothesis testing is used to assess the plausibility of a hypothesis by using sample data. Such data may come from a larger population, or from a data-generating process.

In this question, we will be using Hypothesis testing on our datasets on elephant seal mass. Instead of using the p-value method, we would use the slightly less conventional **Z- Test** method for Hypothesis testing, by finding the **critical regions.**

Z Tests on Our Elephant Seals Data

A marine mammal centre studied claims that for northern elephant male seals has an average mass of 2000 kg. We would like to test this claim on samples of the population.
The null hypothesis, H_0, would be

$$H_0: \mu = 2000$$

where μ is the average mass (in kg) of a northern elephant male seal.
You would like to test this claim using the datasets on northern elephant male seals that you have. So the alternative hypothesis, H_1, would be

$$H_1: \mu \neq 2000$$

and you would do a **two-tail test** to test the hypothesis H_0 at k level of significance, where k is a floating point number between 0 and 1.

It can be assumed, for the purposes of this question, that the least possible northern elephant male seal weight is 1500 kg and the maximum possible elephant male seal weight is 2300kg (both ends inclusive). The population of elephant seal weights is sampled, and it's data is stored in a file of size n. **All** the samples taken from this population would have a mixture of both male weights, female weights, and erroneous data. It is important that you leave only the male entries within the dataset out before proceeding with the rest of statistical calculation. (Make more files for this question if necessary).

The population of elephant seal weights is sampled, and its data is stored in a file of size n.
For each of these samples, calculate the sample mean $\bar{x}$. Use the sample mean to calculate the sample variance s^2. You may assume that the population variance σ^2 is approximately equal to the sample variance s^2.

Sample variance s^2 is expressed as the sum of the squared differences between each value in the sample and the sample mean, divided by the total number of data entries minus 1.

$$\sigma^2 \approx s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

where x_i is the weight of one particular male elephant seal in the sample of n male elephant seals, and $\bar{x}$ is the population mean weight of all elephant seals.

The test statistic T would hence be modelled by the distribution $T \sim N(\mu, \frac{\sigma^2}{n})$.

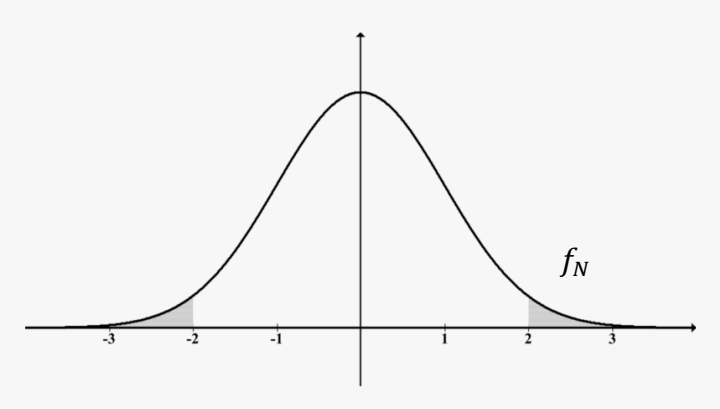
We would like to use the Z statistic to test H_0. To obtain the Z statistic, you would have to make the following normalization, where

$$Z = \frac{S - \mu}{\frac{\sigma}{\sqrt{n}}} \sim N(0,1)$$

A Critical Region (also known as the rejection region), is a set of values for the test statistic for which the null hypothesis H_0 is rejected. If the observed z-score falls within this critical region, then we accept the H_1 as being statistically significant at significance level k.

The invNorm() Function

The diagram below shows an example of a normalized normal distribution curve $Z \sim N(0,1)$. It's



It has a mean of 0 and a variance of 1.

The curve f_N, a special case of the Gaussian Distribution curve, is defined as follows:

$$f_N(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

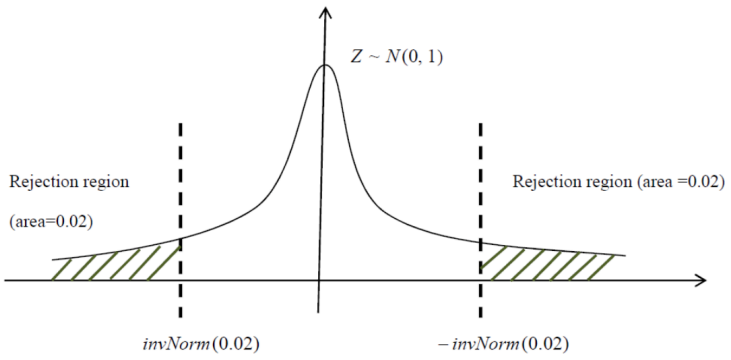
Since probability is the area under the normal distribution curve, then for some value of m,

$$P(Z > m) = \int_m^{\infty} f_N(x)\, dx$$

The **inverseNorm** method (an informal term which stands for inverse normal distribution) is a way to work backwards from a known probability to find an x-value. In the case of the equation above, given a value of $P(Z > m)$ (a floating point number between 0 and 1), the **inverseNorm** method should obtain a value for m.

In code, this represents the function `invNorm()`, which has been implemented for you.

A k level of significance would mean that we reject k values that are furthest away from the population (not sample) mean. For instance, a 5% level of significance would mean that if our **z-score** for our test statistic lies within the furthest 5% of our dataset then we reject the null hypothesis H_0 and accept the alternate hypothesis H_1.



[Refer to the next page for an analysis of this image]

The image on the previous page aims to find the critical region at 4% level of significance, which means, they have to find a value m, such that on the Z distribution,

$$P(Z < -m) = P(Z > m) = 0.02$$

such that in total we create a rejection region when our normalized sample data is found within the furthest 4% of our dataset. Hence, they run the function `invNorm(0.02)` to find the absolute value of m. Hence, in the case of a two tailed test at k significance level, because of the symmetry of the normal distribution curve, we would reject H_0 if our z-score lies within $\frac{k}{2}$ of our dataset that is greatest from the mean and smallest from the mean. The data that lies in these values would constitute our **critical region.**

Calculating z-Score

After obtaining the value of m, we calculate the z-Score, defined as follows:

$$z = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

If z falls in the range of $z < -m$ or $z > m$, claim that it is statistically significant to reject H_0 and accept H_1.

If z falls in the range $-m \leq z \leq m$, then there is insufficient evidence to reject H_0 and we reject H_1 instead.

(Note the lowercase z denotes the z-score, while the capital Z denotes the distribution.)

Questions

(a) Suggest why sample variance is approximately equal to population variance, and an assumption on the sample sizes for this to hold. **[2]**

(b) The general equation for a normal distribution curve is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Show that you are able to derive f_N from f. **[2]**

(c) See Coding Question below. **[10]**

Your Task

Given the level of significance k, a sample S, and the null hypothesis

$$H_0: \mu \geq 1900$$

where μ is the average mass (in kg) of **male** elephant seals.

This sample S is sampled from a population of male weights, female weights and erroneous entries. According to general statistics, there is a pretty distinct discrepancy between the weight of a male elephant seal and a female elephant seal.

- Male elephant seal weight should fall around 1500 – 2300 kg range.
- Female elephant seal weight should fall around 400 – 900 kg range.

Before doing the Z-test on your sample, you should filter your sample such that it will only contain male elephant seal weights.

Write a program in C to use a Z-test on the above hypothesis.

Input Format:

The first line of input would be the name of a **text file** – denoting the sample S, which is sampled from a population of male weights, female weights and erroneous entries.

The second line of input would be a floating point number k, which represents the level of significance of the dataset.

Constraints:
Length of the sample S would not exceed 10^6 integers.
$0 \leq k \in \mathbb{R} \leq 1$
Each integer i in the sample S would be $0 < i \in \mathbb{Z} < 10^5$.

Output Format:
Output two lines:
The first line would be three numbers separated by a space: the first number would be the sample mean $\bar{x}$ (to 5 decimal places), the second number would be the sample variance s^2 (to 5 decimal places) and the third is the z-score of the sample (to 5 decimal places). Note that your answer's accuracy would be graded up to 3 decimal places.
On the second line, then output 1 integer, either 0 or 1, where 0 means that we retain H_0 and 1 means that we accept H_1.

Starter File:
`q6/6_ztests.c`
Using this starter code is optional.
This code file implements the input and output formats as specified above, and passes the input to an empty function which you have to implement. The starter code has provided you a utility function called `invNorm().` You will have to figure out, by analyzing the code, how to use the `invNorm()` function to your benefit. The code file would then take the result of your implementation and print it to the console in the specified format. Note that the invNorm() function also calls another function called `f(),` which you will implement f_N from the preamble. This is required for `invNorm()` to work.

If you have a different idea in mind, feel free to create a new code file for this question.

Hint:
This is a REALLLY LONG QUESTION! You may want to outline what you need to do in order to solve this question, by looking at the preamble once more. To keep you from being overwhelmed by the 3 pages worth of information, let me summarise what you are required to do:

1) Filter out all the non-male elephant entries
2) Find the sample mean and then the sample variance of your male entry dataset.
3) Find the critical region.
4) Find the z score.
5) Conclude your Z- test.

TEST CASES FOR Q6
Test Case 1
Input:

```
sample1.txt
0.05
```

Output:

```
1905.558 50444.840 0.370
1
```

Explanation:
H_0 rejected at 5% level of significance. You can also check that there are 224 valid entries in sample1.txt.

Test Case 2

Input:

```
sample1.txt
0.645
```

Output:

```
1905.558 50444.840 0.370
1
```

Explanation:

In fact, when you plot a graph of $y_1 = \text{invNorm}()$ and $y_2 = 0.370$ (critical region value), invNorm() only becomes larger than the value of c at 0.644 (to 3dp)

Test Case 3

Input:

```
sample2.txt
0.10
```

Output:

```
1959.024 33143.174 2.076
0
```

Explanation:

Note that there are 41 valid entries in this sample. H_0 is not rejected at 10% significance level.

Test Case 4

Input

```
sample2.txt
0.975
```

Output:

```
1959.024 33143.174 2.076
0
```

Explanation: We're aiming to make H1 be accepted. Close, but still not enough!

Test Case 5

Input

```
sample2.txt
0.985
```

Output:

```
1959.024 33143.174 2.076
1
```

Explanation:

Only at a 98.4% (0.984 to 3dp) level of significance, H_1 will be accepted. This is derived from plotting the graphs on the GC.

END OF Q6

END OF WEEK 6 ASSIGNMENT