Lab 2

Project 1

Download the file vabirds.csv.

1. We will start with structs. Create a struct birdData in a file birdDat.cxx.

Attributes:
One member of the type will be the species (a character variable) and the other member will be the list of observations. This should take the form of an allocatable (C style) array.

Subprograms:
A. Write a function to act as the constructor for the type. Its parameters will be the species name and a vector of input data. Allocate the data vector based on the size of the input array. Use the new operator.

B. Write a stats method that takes only an instance of the type and returns the mean and standard deviation of the observations for that instance.

C. Write a minmax method that takes an instance of the type and the array of years and returns the maximum observed, the minimum observed, and the years for maximum and minimum. You may use the maxval, minval, maxloc, and minloc intrinsics.

D. Bonus: This is a struct without methods, so we don't really have "constructors" or "destructors." However, if we wanted to delete unused "birds" we will need to write a function to do so, since memory deallocation will *not* happen automatically. Write a function to delete an instance of birdDat.

2. Write a main program that includes your birdDat.cxx file.

Read the file name from the command line.

Still in read_data, fill a vector of birdData structs. Loop through this array calling your constructor for each species. The read_data routine should return the array of years and the birdData vector.

Request a species name from the user. Find the species in your array of types and print its mean, standard deviation, and results from minmax.

Compute a vector of the means for all species. Sort this vector. You also need the *permutation vector*, which is a vector or array of the indices of the original positions. For example, if after the sort the permutation vector is (17,3,55,11,23, and so forth) that means that the element that was previously 17 is now the first in the new array, and so on. Note that sort return in ascending order (smallest to largest).

To get a permutation index from the C++ sort, use a construct like sort(index.begin(), index.end(), [&] const int &a, const int &b) {return (data[a] < data[b]); }); where index is a vector of integers you set up corresponding to the original indices, and "data" is the vector whose actual sorted order we want. We'd then need to use the permuted index to get the new order of the original data as well. (Note that C++ sort is destructive; it overwrites the input vector.)

From the sorted mean vector and the permutation indices, print the names of the 10 most common (by mean) species over the years of observations.

Test the user input portion for

TurkeyVulture
TuftedTitmouse
ElegantTrogon

Project 2

Convert your type from Project 1 into a class. Separate it into a .h file with the class definition and the method prototypes (this is called the interface) and a .cxx file with the bodies of the methods (this is called the implementation). Only your .h file should be included, with a syntax
#include "birdDat.h"
Use the makemake utility to set up a Makefile.

Project 3

Download the file bodyfat.csv. This is a dataset of body fat, age, height, and weight for a set of participants in a study.

BMI categories are as follows:

Severely underweight	BMI < 16.0
Underweight	16 <= BMI < 18.5
Normal	18.5 <= BMI < 25

Overweight	25 <= BMI < 30
Obese Class I	30 <= BMI < 35
Obese Class II	35 <= BMI <40
Obese Class III	BMI > 40

Write a bmistats.cxx file containing functions/subroutines for the following:

1. Convert pounds to kilograms. Use the actual conversion factor, not the approximate one. Look it up on Google.
2. Convert feet/inches to meters. Look up the conversion factor, do not guess at it.
3. Compute BMI
4. Determine where the user falls in the table supplied and return that information in an appropriate form.

Write a file stats that implements the following:
1. mean of an array (or vector)
2. standard deviation of an array (or vector)
3. outlier rejection using Chauvenet's criterion. Pseudocode given further down.

Write a main program that implements the following:

1. Includes your headers
2. reads the input file into appropriate vectors. Don't assume you know the length of the file (but you can assume the number of header lines is fixed).
3. pass appropriate vectors to a subroutine that computes an array of BMI data based on height and weight and returns the array.
4. Rejects the outlier(s). The function should return an array of Booleans that you can apply to the original data Create new vectors with the outlier(s) deleted.
5. Print a crude histogram of BMI values. To do this, divide the data into an appropriate number of bins as defined by the categories, obtain the count for each bin, and print out a line of asterisks representing the number of BMIs in each bin. It should look something like the example below (I have not computed the results, this just sketches the general appearance of the output)
6. Write a file that contains the corrected data for age, weight, height, bodyfat, and BMI. Use Excel or whatever to plot BMI as a function of percentage body fat. Be sure to plot it as a *scatter plot* (points only, no connecting lines).
7. Create a parameters file and use it to read in the cutoffs for the various bodyfat categories. Pass that information to the routine that computes the BMI categories.

Histogram example:

*****

****
****
****
****
****
****

Chauvenet's criterion:

It's not the state of the art but works pretty well.

1. Compute the mean and standard deviations of the observations.
2. Compute the absolute values of the deviations, i.e. abs(A-mean(A))/std(A)
3. Use the tails devs=devs/sqrt(2.)
4. Compute the probabilities prob=erfc(devs)
   Fortran: erfc is an intrinsic in any fairly recent compiler.

5. The criterion is that we retain data with prob>=1./(2*N_obs) (number of observations)