CS265 Intro to Programming Languages
Program 4 – Spring 2023

Objectives
Work with lists/arrays
Work with functions/methods

Write each of the following programs in C++, Python, and Java. Make sure to use the appropriate extension for each file (.c, .cpp, .py, and .java, respectively). Name each file Prog3ABC.extension, where ABC are your initials. My 3 files would be Prog4TLS.cpp, Prog4TLS.py, and Prog4TLS.java.

```
public static void main (String[] args)
int main()
def main():
```

Assignment

Write each of the following functions. The function header must be implemented **exactly** as specified. For Python the type of value that should be returned from each function is in parenthesis in the function description. Write a main function that tests each of the required functions/methods. You do not have to use keyboard input in `main`, just ensure that each function/method is called more than one. Remember that the functions themselves will not display any output, they will return values that can then be written to the screen.

Specifics
If there are built in functions that will accomplish the tasks lists below YOU CANNOT USE THEM. The purpose of this program is to get practice working with arrays/lists, not to get practice searching for modules or libraries. DO NOT sort the list, that changes the order of the items. The order of the values in the list should be the same as the order of the values in the file.

Do not use a Java `ArrayList` or a C++ `List` or `Vector` (or any variation), use standard arrays.

In main create several arrays/lists and populate them with values. You will an array/list of integers and an array/list of String/string. You can initialize them when you create them. Think of the integer lists as a sequence of quiz scores and the string list as a sequence of student names. Make sure both lists are the same size. They are parallel lists, meaning the data across common indexes is related, the quiz score for the student at index 0 in string list in at index 0 in the quiz list. You can assume all lists/arrays, in all languages will be completely filled. I would test each function with lists of different sizes. DO NOT assume a default size for any list/array.

Required Functions

```
(int) def findMaxScore (scoresList):
int findMaxScore (int[] scoresList)
int findMaxScore (int scoresList[], int size)
```
Return the largest integer value (int) found in the list.DO NOT make any assumptions about the range of values in the list. size is the number of items in the list/array.

```
(int) def findMinScore (scoresList):
int findMinScore (int[] scoresList)
int findMinScore (int scoresList[], int size)
```
Return the smaller integer value (int) found in the list.DO NOT make any assumptions about the range of values in the list. size is the number of items in the list/array.

```
(boolean) def hasPerfectScore (scoresList):
boolean hasPerfectScore (int[] scoresList)
bool hasPerfectScore (int scoresList[], int size)
```
Returns the appropriate true reserved word if any of the scores is 100 or greater, otherwise returns the appropriate false reserved word.

```
(float) def calcAverage (scoresList):
double calcAverage (int[] scoresList)
double calcAverage (int scoresList[], int size)
```
Return the average (double) of all the scores found in the list. Return -1 if unable to determine an array – don't let the program crash.

```
(str) def findStudentWithMax (scoresList, namesList):
String findMaxScore (int[] scoresList, String[] namesList)
string findMaxScore (int scoresList[], string namesList[], int size)
```
Return the name of the student with the largest score. You can assume there will be only one student with the largest score. IF there were two students with the largest score (there won't be) you can return the name of the first student with that score.

```
([]) def mergeLists (list1, list2):
int[] mergeLists (int[] list1, int[] list2)
int* mergeLists (int list1[], int list2[])
```
Return a new array/list made up of the values from the two lists, values from list1 first, then the values from list2. This might not find into the theme of the other functions, but we'll still include it in the list of required functions.

```
(float) standardDeviation (scoresList)
Double standardDeviation (int[] scoresList)
double standardDeviation (int scoresList[], int size)
```
Return the standard deviation of the values. To find the standard deviation start with the average of the list – DO NOT repeat the code for calculating the average. Then for each value in the list subtract the average from that value and square the result. Find the average of the squared differences and then take the square root of that average.

For example, if the list contains the values 4, 5, 6, and 3 the average of those values would be 4.5. Subtract 4.5 from each value and square that results. The differences would be -0.5, 0.5, 1.5 and -1.5 respectively. The square of each of those differences would be 0.25, 0.25, 2.25, and 2.25 respectively. The average of these values is 1.25 ((0.25 + 0.25 + 2.25 + 2.25) / 4).

The following is copied from a spreadsheet used to calculate the standard deviation.

4	5	6	3		values
				4.5	average
-0.5	0.5	1.5	-1.5		difference from average
0.25	0.25	2.25	2.25		difference squared
			1.25		average of "difference squared"
			1.118034		square root of above average

```
(int) def findInList (scoresList, valueToFind, startingIndex=0):
int findInList (int[] scoresList, int valueToFind)
int findInList (int[] scoresList, int valueToFind, int startingIndex)
int findInList (int scoresList[], int valueToFind)
int findInList (int scoresList[], int valueToFind, int startingIndex)
```
Return the index of the first occurrence of valueToFind. IF specified, starting searching at the specified index, otherwise start searching at index 0. This requires a default parameter in Python and overloaded functions/methods in Java and C++. In any of the functions return -1 if the value is not found. The pseudocode for using these functions might be

```
index = findInList (scoresList, 100)
while index != -1
   print (“Found 100 at index “, index)
   index = findInList (scoresList, 100, index + 1)
```