

CIS2107_Lab 06: "Arrays of Pointers to Functions"

Points: 50 points

Project 01: Arrays of Pointers to Functions

Objective:

- To design and implement array of function pointer.

Instructions:

- Be sure to document your code (add comments on top of each function).
- In the comments add your name, date, course, homework number, and statement of problem.
- Once you are done, upload your final solution through Blackboard.
- No need for input validation.
- Write a project called **ArraysofPointers.c**

Steps:

- Rewrite the program of Fig. 6.22 to use a menu-driven interface. The Program should offer the user four options as follows:

```
Enter a choice:
0  Print the array of grades
1  Find the minimum grade
2  Find the maximum grade
3  Print the average on all tests for each student
4  End Program
```

- One restriction on using arrays of pointers to functions is that all the pointers must have the same type. The pointers must be to functions of the same return type that receive arguments of the same type.
- **[10 points]** For this reason, the functions in Fig. 6.22 must be modified so that they each return the same type and take the same parameters.
- **[20 points]** Modify functions `minimum` and `maximum` to print the minimum or maximum value and return nothing. For option 3, modify function `average` of Fig. 6.22 to output the average of each student (not a specific student).
- **[10 points]** Function `average` should return nothing and take the same parameters as `printArray`, `minimum`, and `maximum`.
- **[10 points]** Store the pointers to the four functions in array `processGrades` and use the choice made by the user as the index into the array for calling each function.

(Testing main) **ArraysofPointers**

- First declare a 2-dim array. How to do that? You need to read the number of rows and the number of columns from the user, and then it reads a corresponding entries to that size. E.g., if a user enters 3 for the number of rows, and enters 3 for the number of columns, then we declare an array of 9 and then read 9 entries and store them in the array. (Remember the Run-time Array Length!).
- Call all functions in part 1 to match the sample menu above. So after each call, you are going to throw the menu again, and keep doing that as long as the user is not entering 4 to end program.
- Use blank lines to separate outputs and make them more readable.