Name:

|  |  |  |
| --- | --- | --- |
| Problem | Points | Score |
| 1 | 50 |  |
| 2 | 50 |  |
| Total | 100 |  |

Notes:

1. The first step in this exam is to create a workspace in the following directory:

/data/courses/ece\_1111/current/exams/ex\_02

Your directory should be your last name all lowercase, followed by an underscore, following by your first name (e.g, “picone\_joseph”). Set the permissions using “chmod u+rwx,g-rwx,o-rwx <lastname>” so only you have read and write permission to this directory. Create subdirectories within this directory: p01, p02, … You will use these for problems 1 and 2, … respectively. Put ALL your code in these directories. Do not touch your files after the exam is over.

1. For this exam you must only use concepts discussed in this class. If you try to use ChatGPT to solve these problems, and it produces code more advanced than what we have covered, you will be penalized. You might also have to come to my office to explain your code.🧑‍💻Your code must be your own original work.
2. All your solutions should use a make file, a header file, a driver program (e.g., p01.cc), and an implementation file (e.g., p01\_00.cc). Your executable should be named \*.exe (e.g., p01.exe).

**(50 pts) Problem No. 1 (/p01)**:

This file:

/data/courses/ece\_1111/current/exams/ex\_02/picone\_joseph/p01/coefficients.dat

contains coefficients of a vector. The first line always contains the length of the vector. Assume any coefficients not specified are set to $0$.

Write a filter program that does the following:

* loads this coefficient file into a vector $a[]$, and load the order into a variable $N$;
* reads a file of $16$-bit short integers $N$ values at a time into an array $s[]$;
* computes a dot product between the signal and the vector (sum the products of the elements of $a[]$ and $s[]$: $a[0]s[0] + ... + a[N-1]\*s[N-1]$);
* prints the buffer number, starting with 0 and the value of the sum.

Your program should follow this interface (use a %3d format for the index and %10.4f for the format of the sum):

p01.exe coefficients.dat signal.dat

buffer no. 0: sum = ????.????

buffer no. 1: sum = ????.????

buffer no. 2: sum = ????.????

buffer no. 3: sum = ????.????

....

You must read the signal data N samples at a time – do not read the entire signal into memory. If a signal buffer does not have enough data, fill the missing values with zeroes (e.g., if the order is $7$, and the file length is $16$, then the last buffer will only have two samples).

In the same directory mentioned above, you can find an example file named signal.dat which you can use for testing. Your program should work for any signal file and any coefficient file.

You must implement the dot product function using a function named my\_dot\_product implemented in a file p01\_00.cc and declared in p01.h. It must operate on a single buffer of the signal – do not pass the entire signal to this function. You are free to define the arguments to this function, but you cannot use global variables.

**(50 pts) Problem No. 2 (/p02):**

Write a program named p02.cc that uses a function my\_allocate() to allocate a 4-dimensional array, populates it with values of $-1$, and then prints the element values to stdout. The function should allocate space and populate the array. The main program, which we refer to as the driver program, should call my\_allocate() and then print the values of the array to stdout (printing cannot be done inside my\_allocate()).

Your program should work like this:

p02.exe 1 2 3 4

a[0][0][0][0] = -1.0000

a[0][0][0][1] = -1.0000

a[0][0][0][2] = -1.0000

....

where the arguments “$1 2 3 4$” are the four dimensions of the array. The first dimension is “$1$” and the last dimension is “$4$”: float a[1][2][3][4] would be the correct dimension for the above example.