PIC-10A: Homework 6

Due 11/19/21 11:59pm via CCLE

Problem 1

Write a function

```
void bar_chart(double values[], int size)
```

that displays a vertical bar chart of the `values` in values, using asterisks. You can assume the array `values` only contains non-negative numbers. First figure out the maximum value in `values`. That value's bar should be drawn with 20 asterisks. Shorter bars should use proportionally fewer asterisks. For example, if `values` contains 5 numbers 45, 90, 100, 80.8, 59.1, and 24.8, then the function should display

```
   *
   *
  **
  **
  ***
  ***
  ***
  ***
  ****
  ****
  ****
*****
*****
*****
*****
******
******
******
******
******
```

Download the file *hw6_1.cpp*, write your function in this file. Save and submit your file as *hw6_1.cpp*

Problem 2:

Write a function:

```
void mat_vec_mul(double mat[][DIM2], double vec[], double result[],
                                                    int dim1)
```

that computes the matrix-vector multiplication between matrix `mat` of size `dim1 x DIM2` and vector `vec` of size `DIM2`. The function then stores the result in the array `result`.

Note:

- Matrix-vector multiplication between a matrix `A` of size `n1 x n2` and a vector `b` of size `n2` is a vector `x` of size `n1` where:
- The i_th element of vector `x` is defined as a dot product between the i_th row of matrix `A` and vector `b`
- The dot product between two vectors $x = [x_1, x_2, ..., x_n]$ and $y = [y_1, y_2, ..., y_n]$ of the same size `n` is defined as follows:

$$< x, y > \; = x_1 y_1 + x_2 y_2 + ... + x_n y_n$$

- An illustration of matrix-vector multiplication is shown in the below figure

$$\begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ b_1 & b_2 & b_3 & \dots & b_n \\ c_1 & c_2 & c_3 & \dots & c_n \\ \dots & \dots & \dots & \dots & \dots \\ m_1 & m_2 & m_3 & \dots & m_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} (a_1 x_1) + (a_2 x_2) + (a_3 x_3) + \dots + (a_n x_n) \\ (b_1 x_1) + (b_2 x_2) + (b_3 x_3) + \dots + (b_n x_n) \\ (c_1 x_1) + (c_2 x_2) + (c_3 x_3) + \dots + (c_n x_n) \\ \dots \\ (m_1 x_1) + (m_2 x_2) + (m_3 x_3) + \dots + (m_n x_n) \end{bmatrix}$$

You might need a helper function for this program.

Download the file *hw6_2.cpp*, implement your function in this file. Save and submit your file as *hw6_2.cpp*

Problem 3:

In this problem, you implement a function `safe_strnAppend(str1,str2,capacity1,n)` that appends at most `n` characters from a C-String `str2` after the end of the characters in a C-String `str1`

```
void safe_strnAppend( char str1[], char str2[], int capacity1, int n
                                                                    )
```

where `capacity1` is the capacity of the character array `str1`. You can assume that `capacity1` at least equals the length of the C-string `str1` + 1. The number of appending characters cannot exceed the number of remaining spaces in `str1` and the length of `str2`.

Your function needs to:

- Check whether the character array `str1` still has spaces for appending, let say there are `m` spaces left. For example, if your char array `str1` has capacity 20, and length = 10, then m=9 (recall one space is used for null character `'\0'`).
- Calculate the length of `str2`, let say `p`. For example if `str2 = "string 2"` then p=8.
- The actual number of appending characters is the smallest value among `m`, `p`, and `n`
- Recall that C-String ends with the null character **`'\0'`**.
- Do not use `strncat` or `strcat` in this problem. `strlen` is fine.

Download the file *hw6_3.cpp*, write your function in this file. Save and submit your file as *hw6_3.cpp.*