# Assignment 1

## Receiving Inputs:

* The function which I created to read inputs, is of the following format:

**int insertArtists(char artists[][80], char songsArtist1[][80], char songsArtist2[][80],  
 char songsArtist3[][80], char songsArtist4[][80], int numSongsPerArtist[4]);**

This function takes in 6 arrays as inputs and returns an integer value (the total number of Artists entered). This integer value is then stored in the **numOfArtists** variable in the main function.

* To read inputs from the command line, I used the function, **fgets( )**, within my created function. I used this function because it takes three important arguments: the memory location where the string must be stored; the maximum length of the string; and the file from which it must read the input (which is **stdin** – inputs taken from the keyboard)
* **fgets( )** is a better function to use in comparison to **scanf( )** as **fgets( )** is particularly used for reading strings.

## Sorting:

* I used the Insertion Sort algorithm for sorting.
* I chose Insertion Sort because it is a more efficient algorithm in comparison to Selection sort and Bubble sort. Insertion Sort checks through each song in the array, while also ensuring that as it moves through the array, the already checked songs are in order. Therefore, it only checks through additional previous songs if the song just before the current song is in the wrong position alphabetically. Otherwise, the checked songs are sorted! The algorithm doesn’t go through any unnecessary checks. As a result, Insertion Sort is also quicker compared to Selection sort and Bubble sort.
* Design Implementation:

I copied the content of the artists array using **memcpy( )** into the **sortedArtists** array.

When the songs and artists were stored in arrays in the input function, the songs of the first artist were stored in **songsArtist1[][80],** the songs of the second artist were stored in **songsArtist2[][80]** …etc.

Therefore, when sorting the array of sortedArtists, it was vital to rearrange the **numSongsPerArtist[]** array and the various songs arrays too. The rearrangement of songs array was done using **memcpy(** ) function. So that once the artists are sorted, the first artist in the **sortedArtists[4][80]** had its songs in **songsArtist1[][80]**, the songs of the second artist according to the **sortedArtists[4][80]** has its songs in **songsArtist2[][80]** …etc.

Once the rearrangement of the songs array was done, each **songsArtist** array was then sorted separately using Insertion Sort algorithm.

* Finally, using a **for** loop and **printf( )** function, I printed out the sorted list of songs.

## Shuffling:

* Design Implementation:

I created a pre-shuffle array **preShuffledPlaylist[24][165];** where each element consisted of artist name, followed by a hyphen and then the song name. To do this, I used the strncat( ) function.

This array contains every song twice, as per the requirement.

Then, I copied the elements of this array using **memcpy( )** into a ‘Shuffle’ array- char **ShuffledPlaylist[24][165]**. I then shuffled this array using the Fisher Yate’s algorithm.

* The shuffling involved three parts: generating a random number (for new swapping position); checking the neighbouring five songs for repetition; and finally doing the swap.
* I used the **rand( )** function with the **time(NULL)** seed along with the modulo function to generate a random number less than the total count of songs.
* To check for the presence of the current song (to be swapped), in its 5 neighbouring songs, I used a Boolean variable- **bool swapAllowed**; and multiple **if**/**else** statements. Before the check, I also ensured that the total number of songs in the ShuffledPlaylist is greater than 12, so that the requirement of repeating a song after at least 5 songs have been played can be met. Otherwise, the program will print out a warning with an improper shuffled playlist.
* If the result of conditions satisfies the requirements, **swapAllowed** remains **true**. The swap happens using the **strcpy( )** function and an additional char array named as **swapShuffle**.
* Finally, using a **for** loop and **printf( )** function, I printed out the Shuffled Playlist.