NIlesh Gupta

604489201

CS 31 Homework #4

One obstacle I had with this program was figuring out enum. Since I have only programmed in Java before, I had never used enum so the concept was foreign. However, since it is a relatively simple idea, understanding it did not take long.

The hardest part of this assignment was determining the appropriate algorithm for each function. In particular, the merge and delete duplicates function took the longest to figure out. Alternatively merging elements from two arrays into one was not too difficult; the fact that the resulting array had to be in non increasing order was tricky. I was stuck on this function for quite a while before I figured out that I should sort all elements from the smaller array and the same number of elements from the larger array first. Thus, my first for loop arranges 2\*min number of elements in non increasing order from the arrays. Then, the second for loop takes care of all the elements unaccounted for. The biggest breakthrough I had with that function was realizing the need for the i— in the case where a1[i] was not the greatest element. This ensures that all elements of the first array get inserted into result[].

(See below for test cases)

| Input | Reason |
| --- | --- |
| song[1001] | These arrays all have elements greater than TOTAL\_NO\_SONGS (1000) |
| singer[1001] | These arrays all have elements greater than TOTAL\_NO\_SONGS (1000) |
| spi[1001] | These arrays all have elements greater than TOTAL\_NO\_SONGS (1000) |
| votes[1001] | These arrays all have elements greater than TOTAL\_NO\_SONGS (1000) |
| grammy[1001] | These arrays all have elements greater than TOTAL\_NO\_SONGS (1000) |
| SPI\_Class = SPI\_BRONZE | This is not a valid value defined for SPI\_Class |
| SPI\_Change = SPI\_NO\_CHANGE | This is not a valid value defined for SPI\_Change |
| string jazz[4] = {"Take Five", "Angela", "Affirmation", "Angela”};  int i = deleteDuplicates(jazz, 20); | The program can’t account for misinformation on the part of the user |
| SPI\_CLASS some\_spis[4] = { SPI\_SILVER, SPI\_COPPER, SPI\_GOLD,  SPI\_COPPER };  int some\_votes[4] = { 5001, 4999, 4001, 1 };  bool some\_grammy[4] = { false, false, false, true };  SPI\_Change some\_changes[4];  int i = computeSPI(some\_spis, some\_changes, some\_votes, some\_grammy,  4); | This test data adeptly tests the computeSPI() function of my program. After the function runs, some\_spis is now {SPI\_GOLD, SPI\_SILVER, SPI\_SILVER, SPI\_PLATINUM} and some\_changes is now {SPI\_CHANGE\_UP, SPI\_CHANGE\_UP, SPI\_CHANGE OWN, SPI\_CHANGE\_UP}  The function returns 0 |
| spi[4] = {SPI\_GOLD, SPI\_GOLD, SPI\_GOLD,  SPI\_GOLD}  int songList[];  int i = getRankingList(spi, SPI\_GOLD, songList, 4) | This tests the getRankingList function. After the function runs, songList[] will be {0,1,2,3,-1}  The functions returns 0 |
| spi\_change[4] = {SPI\_CHANGE\_UP, SPI\_CHANGE\_NO, SPI\_CHANGE\_UP,  SPI\_CHANGE\_DOWN}  int songList[];  int i = getChangeList(spi\_change, SPI\_CHANGE\_UP, songList, 4) | This tests the getChangeList function. After the function runs, songList[] is {0,2,-1}  The function returns 0 |
| grammy[3] = {true, true, false}  int songList[];  int i = getGrammyList(grammy, songList, 3) | Testing the getGrammyList function. After function runs, songList is {0,1,-1}  The function returns 0 |
| string jazz[4] = {"Take Five", “So What?”, "Angela”, "Affirmation"};  string rap[3] = {"YG my hitter", "Drake trophies", "B.o.b headband"};  string z[30];  int n = merge(jazz, 4, rap, 3, z, 30); | This tests the merge function. After merge is called, z will be "YG my hitter" "Take Five" “So What?” "Drake trophies” "B.o.b headband" "Angela” “Affirmation”  The function returns 7 |
| string favorite[8] = {  “Style", “If”, "Clean”, “Clean”, “Heaven”, “Heaven”, "If", “If”  };  int n = deleteDuplicates(favorite, 8); | Testing the deleteDuplicates function. After the function is called, favorite[0-4] will be {“Style”, “If”, “Clean”, “Heaven”, “If”}  The function returns 5 |
| string favorite[6] = {  “Style", “If”, "So What?”, “Clean”, “Heaven”, "Take Five"  };  int r = moveToStart(favorite, 6, 1); | After the function is called, favorite will be {“If”, “Style”, "So What?”, “Clean”, “Heaven”, "Take Five”}  “If” has been moved to the start  The function returns 1 |
| string favorite[6] = {  “Style", “If”, "So What?”, “Clean”, “Heaven”, "Take Five"  };  int r = moveToEnd(favorite, 6, 0); | After the function is called, favorite will be {“If”, "So What?”, “Clean”, “Heaven”, "Take Five”, “Style”}  “Style” has been moved to the end  The function returns 0 |
| int i = computeSPI(some\_spi, some\_changes, some\_votes, some\_grammy,  -1); | This header is invalid and the function will return -1 since size < 0 |
| int i = getRankistList(spi, SPI\_PLATINUM, songList, -1) | This header is invalid and the function will return -1 since size < 0 |
| int i = getChangeList(spi\_changes, SPI\_CHANGE\_NO, songList, -1) | This header is invalid and the function will return -1 since size < 0 |
| int i = getGrammyList(grammy, songList, -1) | This header is invalid and the function will return -1 since size < 0 |
| int n = merge(jazz, -1, rap, 3, z, 30); | This header is invalid and the function will return -1 since size1 < 0 |
| int n = merge(jazz, 4, rap, -1, z, 30); | This header is invalid and the function will return -1 since size2 < 0 |
| int n = merge(jazz, -1, rap, 3, z, 30); | This header is invalid and the function will return -1 since size < 0 |
| int n = merge(jazz, 4, rap, 3, z, 6); | This header is invalid and the function will return -1 since size1 + size2 > size |
| string jazz[4] = {“Affirmation”, ”Take Five", “So What?”, "Angela”};  int n = merge(jazz, 4, rap, 3, z, 30); | This header is invalid and the function will return -1 since jazz[] is not in nonincreasing order |
| int i = deleteDuplicates(somestring, -1) | This header is invalid and the function will return -1 since size < 0 |
| int i = moveToStart(somestring, -1, 3) | This header is invalid and the function will return -1 since size < 0 |
| int i = moveToStart(somestring, 3, 3) | This header is invalid and the function will return -1 since loc >= size |
| int i = moveToEnd(somestring, -1, 3) | This header is invalid and the function will return -1 since size < 0 |
| int i = moveToEnd(somestring, 3, 3) | This header is invalid and the function will return -1 since loc >= size |
| string album[3] = { "If", "Angela, "Heaven" };  string playlist = album[3]; | This will crash the program. There is no way to account for this error |