10050

Software Engineering Project – 1 (COMP10050)

ASSIGNMENT -1

SURABHI AGARWAL
STUDENT NUMBER - 17203535

DETAILED SPECIFICATION – ASSIGNMENT 1
IPOD MUSIC PLAYER

All the following function declarations are present in the header file folder in the project.

- **Functions used to Insert songs and artists:**
  I have implemented the insertion of songs and artists in the main itself using a nested for loop and if else statements.
  Code:

```
for(int i=0; i<4 ; i++) {
    for(int j=0; j<3 ; j++) {

        //the first condition for the first artist
        if(i == 0) {
            if(j == 0) {
                printf("Insert an Artist/Group name ");
                fgets(artistName, 80 , stdin); //fetching artist 1 from user
                strcpy(artists[i],artistName); //copying fetched artistName to array artists for artist 1
                 numOfArtists++; //incrementing number of artists to 1

            }
           //For the songs of artist 1
            printf("Insert song %d for %s ", j+1,artistName);
            //using
            fflush(stdin); //flushes out any buffered data and takes input from the user
            fgets(songName, 80 , stdin);  //fetching song names for artist 1
            strcpy(songsArtist1[j],songName); //copying fetched songName from user to array of songs for artist 1
             numSongsPerArtist[i]++; //incrementing array of number of songs per artist every time new songs entered



        } else if(i == 1) {  //Condition for the second artist
```

- **Functions used to sort songs and artists:**
  Made use of the following functions to sort the array of artists and songs.
  void sortSongs(char songsOfAnArtist[][80], int numOfSongs);
  void sortArtists(char sortedArtists[][80] , int numOfArtists);
  Passing sub array and an integer value i.e number of songs/artists as arguments of the function.
- **Algorithm implemented to sort songs:**
  Made use of Quick Sort Algorithm instead of other algorithms because:
  - The quicksort algorithm partitions an array of data into items that are less than the pivot value and those that are greater than or equal to the pivot item.
  - Results in O(n log n) quadratic complexity.
  Code:

```
 void sortArtists(char artists[][80], char sortedArtists[][80] , int numOfArtists){
  //Declaring some variables
  int i =0;
  int j=0;
  int minIndex = 0;
  char swap [80];

  for(i=0; i<numOfArtists; i++){
      strcpy(sortedArtists[i], artists[i]);
  }


  /* i initially points to the first element of the array, it iterates all the elements accept the last
     and minIndex is set to i */
  for(i=0; i< (numOfArtists-1); i++){
      minIndex = i;  //minIndex assigned as i
      for(j=i+1; j<numOfArtists; j++){
         //finds the smallest string(alphabetically) in the array sortedArtists[j,numOfArtists]
          if(strcmp(sortedArtists[j], sortedArtists[minIndex]) < 0) //compares using strcmp function
              minIndex = j; //minIndex is now assigned to j
      }
 memset(swap, '$', 80-2);  //swap is initialised using function memset to avoid runtime errors
 swap[80-1]='\0';
 //using strcpy function strings at position i and minIndex are swapped
 strcpy(swap,sortedArtists[i]);
strcpy(sortedArtists[i], sortedArtists[minIndex]);
 strcpy(sortedArtists[minIndex], swap);
}

}
```

- We have taken a variable called minIndex and at first initialised it to i, i iterates through the array accept the last element. Then we increment counter j, to find the smallest string alphabetically, which we find through comparing the string sub array j and minIndex.
- **Advantages** include the efficient average case compared to any a forementioned sort algorithm, as well as the elegant recursive definition, and the popularity due to its high efficiency. The quick sort produces the most effective and widely used method of sorting a list of any item size.

- **Functions used to shuffle songs:**

Made use of the following functions to sort the array of artists and songs.
void shuffleSongs(char songsToBeShuffled[][80], int numOfSongs);
Passing sub array and an integer value i.e number of songs as arguments of the function.

- **Algorithm implemented to shuffle songs:**

Made use of Fisher Yates algorithm to shuffle songs.

```
To initialize an array a of n elements to a randomly shuffled copy of source, both 0-based:
  for i from 0 to n − 1 do
      j ← random integer such that 0 ≤ j ≤ i
      if j ≠ i
          a[i] ← a[j]
      a[j] ← source[i]
```

This is a pseudo code which we are implementing of inside out algorithm. We can still generate a uniformly random list of data even if we do not know the parameter n, hence this sets as one of the advantages of the source.
Code:

```
void shuffleSongs(char songsToBeShuffled[][80], int numOfSongs){
  //declaring some variables
   int  i, j;
   srand(time(NULL)); //called once to initialise random number generation
   char swap[80];


   for(i=1;i<numOfSongs;i++){

       j = rand()%(i+1); //generated random number comprised between 0 and i

       if(j!=i){
           //if i is not equal to j then strings at positions i and j are swapped
          strcpy(swap, songsToBeShuffled[j]);
          strcpy(songsToBeShuffled[j],songsToBeShuffled[i]);
           strcpy( songsToBeShuffled[i], swap);


       }
   }
   for(i=0; i<numOfSongs; i++){
       printf(" - %s", songsToBeShuffled[i]); //prints shuffled songs
   }
  }
```