DESIGN DOCUMENT: Project 1

NAME: Naveela Sookoo
STUDENT ID: nnsookoo

The assigned task was to create a C++ program that manages a playlist of songs, that is, adding songs to the end of the playlist, selecting a song to be played and erasing a song at a given position.

Two classes were created – ***class Song*** which stores the properties of each individual song and provides services such as setting the title and setting the artist's name and ***class Playlist*** which stores the list of songs using vector as well as provides services such as adding a song, deleting a song and playing a song.

class Song:

This class contained only public members as some attributes will be needed within the class Playlist. The public members included ***string songTitle*** (to store the title of the song), ***string artistName*** (to store the artist's name of the song) and ***int position*** (to store the position of the song).

The member functions include:

1. Constructors:
   a. To create an empty song: ***Song();***
   b. To create a defined song: ***Song(const string& Title, const string& Artist, const int number);*** This was designed to take the attributes of the song to create a new song object.
2. Destructor: ***~Song();***
   This destructor clears/deletes the attributes of the song
3. ***setSongTitle(const string& Title);***
   This 'void' function (does not return anything) is used to set the title of the song. A string with the title is passed.
4. ***setArtistName(const string& Artist);***
   This 'void' function (does not return anything) is used to set the artist's name of the song. A string with the artist's name is passed.

class Playlist:

This class contained member objects (private) and member functions (public). Its private member objects included ***vector<Song*> song_list***, a vector array which contained the various songs found in the playlist and a ***double length***, which contained the amount of songs in the playlist.

The member functions included:

1. Constructors:
   a. To create an empty song: ***Playlist();***
   b. To create a copy of an already defined playlist: ***Playlist(const Playlist& p);*** This was designed to take the attributes of the song to create a new song object.
2. Destructor: ***~Playlist();***
   This destructor clears/deletes the attributes of the playlist. It calls on the destroy() function to do this.
3. ***destroy();***
   This function clears/deletes the playlist, that is, it clears the vector which contains the song as well as sets the length of the playlist object to 0. It only clears the playlist if it is not empty. The empty() function was used to test if the vector was empty. This was chosen to be implemented over the size() function as empty() is implemented in constant time whereas size() depends on the number of objects in the vector.

NAME: Naveela Sookoo
STUDENT ID: nnsookoo

4. ***addSong(const string title, const string artist);***
   This bool function adds a new song to the end of the playlist given two strings containing the title of the song and the artist name and returns a true if the addition was successful and false if it was not. Several edge cases were included such as if the song was already included in the playlist or if the string containing the title/artist passed into the function was empty. This function utilised the push_back() function found in the vector STL library. The time taken to execute this function is a constant and therefore does not take a great amount of time.

5. ***eraseSong(const int n);***
   This bool function erases a song given its position in the playlist. It returns a true if the function was successful in erasing the object and a false if the function was not successful in doing so. Several edge cases were also included in the structure of the function. First, it was checked if the position, n, passed into the function was negative (or equal to zero). In this case, it would have returned false. Secondly, it was checked if the position, n, passed into the function was greater than the length of the playlist. This would have meant that that song did not exist within the playlist and hence would have returned false. The song was found using its index in the vector array and then erased using erase() function found in the vector STL library. The time complexity of this erase() function however, would have depended on the location of the song within the playlist. In the worst case, the running time would have depended linearly on the number of songs in the playlist when the first element is deleted whereas in the best case, the running time would have been constant if the last element was deleted.

6. ***playSong(const int n);***
   This bool function finds a song given its position in the playlist and "plays" it. It returns a true if the function was successful in locating the song and a false if the function was not. Again, several edge cases were included such as if the position, n, was a negative number (or equal to zero) and if the position, n, was greater than the length of the playlist. Else, it would have meant that the song was found in the playlist and would have returned true.

7. ***returnTitle(const int n);***
   This function, given the position of the song, returns a string with the title of the song.

8. ***returnArtist(const int n);***
   This function, given the position of the song, returns a string with the artist's name of the song.