#### 

**A MINI PROJECT REPORT**

#### ON

# SavTunes

## *Submitted in partial fulfillment for the award of the degree of Bachelor of Engineering*

#### *In*

**COMPUTER SCIENCE AND ENGINEERING**

***Submitted by***

#### Savion Mario Sequeira

#### 1NH17CS745

#### 3 C

***Reviewed by***

#### Ms. Asha Rani Borah

#### [Sr. Asst. Professor, Department of CSE]

## New Horizon College of Engineering

Certificate

# This is to certify that the mini project work titled

SavTunes

##### *Submitted in partial fulfillment for the award of the degree of Bachelor of Engineering*

#### Savion Mario Sequeira

#### 1NH17CS745

**During the academic year**

***2018-2019***

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Signature of Reviewer** |  | **Signature of HOD** |

**Semester End Examination**

|  |  |
| --- | --- |
| *Name of the Examiner*  ***1. .…………………………….*** | *Signature with date* ***……………………………*** |

***2. ..…………………………… ……………………………***

# ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be, but impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned my efforts with success.

I thank the management, **Dr. Mohan Manghnani**, Chairman of NEW HORIZON EDUCATIONAL INSTITUTIONS for providing necessary infrastructure and creating good environment.

I also record here the constant encouragement and facilities extended to me by **Dr. Manjunatha**, Principal, NHCE, **Dr. Prashanth.C.S.R,** Dean Academics and **Dr. B. Rajalakshmi**, Head of the Department of Computer Science and Engineering. I extend my sincere gratitude to them.

I express my gratitude to **Ms. Asha Rani Borah** , my project reviewer for constantly monitoring the development of the project and setting up precise deadlines. Her valuable suggestions were the motivating factors in completing the work.

Finally a note of thanks to all the teaching and non-teaching staff of Computer Science and Engineering Department for their cooperation extended to me and my friends, who helped me directly or indirectly in the course of the project work.

**Savion Mario Sequeira**

**1NH17CS745**

i

**ABSTRACT**

SavTunes is a working model of how a music management application would work in the real world. This project aims to create a working model of the app which will allow you to dynamically allocate memory for songs which means if the space is sufficient one could store as many songs as they wish to.

The project also want to showcase the sorting feature of a music management application and how you can sort the songs as per artists name, song name or even genre the user can decide that just like in the real world.

Lastly the project will to showcase a searching feature ,in the real world one would have more than 100 songs stored in the application and it’s not feasible to make the user to scroll through each song until they find the correct one, hence with the searching feature they can enter any keyword in the song like the artists name or song name and the application will present all the songs that match the keyword.

ii

|  |  |  |  |
| --- | --- | --- | --- |
| **TABLE OF CONTENTS** | | | |
| **CHAPTERS** | | | |
| **CHAPTER NO** | **TITLE** | | **PAGE NO** |
|  | **Abstract** | | i |
|  | **Acknowledgment** | | ii |
| 1 | **Introduction** | | 1 |
| 2 | **Analysis and Design** | | 2 |
| 3  4  5  6 | 2.1 Objectives of the Project  2.2 Requirement Specification  2.2.1 Hardware Specification  2.2.2 Software Specification  2.3 Algorithm/Pseudo Code  2.3.1 Flowchart  2.3.2 Algorithm  2.3.3 Data structures  **Implementation**  3.1 Dynamic Memory Allocation  3.2 Structures  3.3 Sorting  3.4 Linear Search  3.5 Delete  3.6 Shuffle  **Sample Output**  **Conclusion**  **Bibliography** | | 2  2  2  2  3  3  5  6  7  7  7  7  8  9  9  9  13  14 |
|  | |  |  |
|  | | iii  **LIST OF FIGURES** |  |

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure name** | **Page No.** |
| 2.3.1 | Flowchart | 3 |
| 3.2 | Structures | 6 |
| 3.3 | Sorting | 7 |
| 3.4 | Linear search | 8 |
| 3.5 | Delete | 8 |
| 3.6 | Shuffle | 9 |

**Chapter 1**

**Introduction**

SavTunes is a take on how current music apps work , inspiration being Itunes. This program tries to give a very basic outline of the software and how it works.

SavTunes has all the basic functionalities of a Music Player:

1.Insert

2.Sort

3.Search

4.Delete

5.Shuffle

With SavTunes you can experience adding how many ever songs you choose to, sorting them based on artist , song name or genre. The search functionality allows you to just type the name or artist or genre of any song and it will display all possible results that exist. Added a song by mistake ?, well you can use the delete function to delete any song you choose to. With the shuffle feature you the software at random displays you the songs you have entered in the application.

**Chapter 2**

**Analysis and Design**

**2.1 Objectives of the project**

The goal of SavTunes is to let the user experience a rundown simulation of a music application and to understand the working behind it. It allows the user to use all the basic functionalities that would exist in any music player application like Itunes.

**2.2 Requirement Specification**

**2.2.1 Hardware Requirement**

* Processor : Intel i3/i5/i7 3rd Generation or more
* Ram : 512 MB or more
* Hard Disk : 250 GB or more

**2.2.2 Software Requirement**

* Windows XP or more / MAC Os / Linux
* CodeBlocks with mingw compiler
* Data Structures like Linked Lists , Arrays

**2.3 Algorithm / Psuedo Code**

**2.3.1 Flowchart**

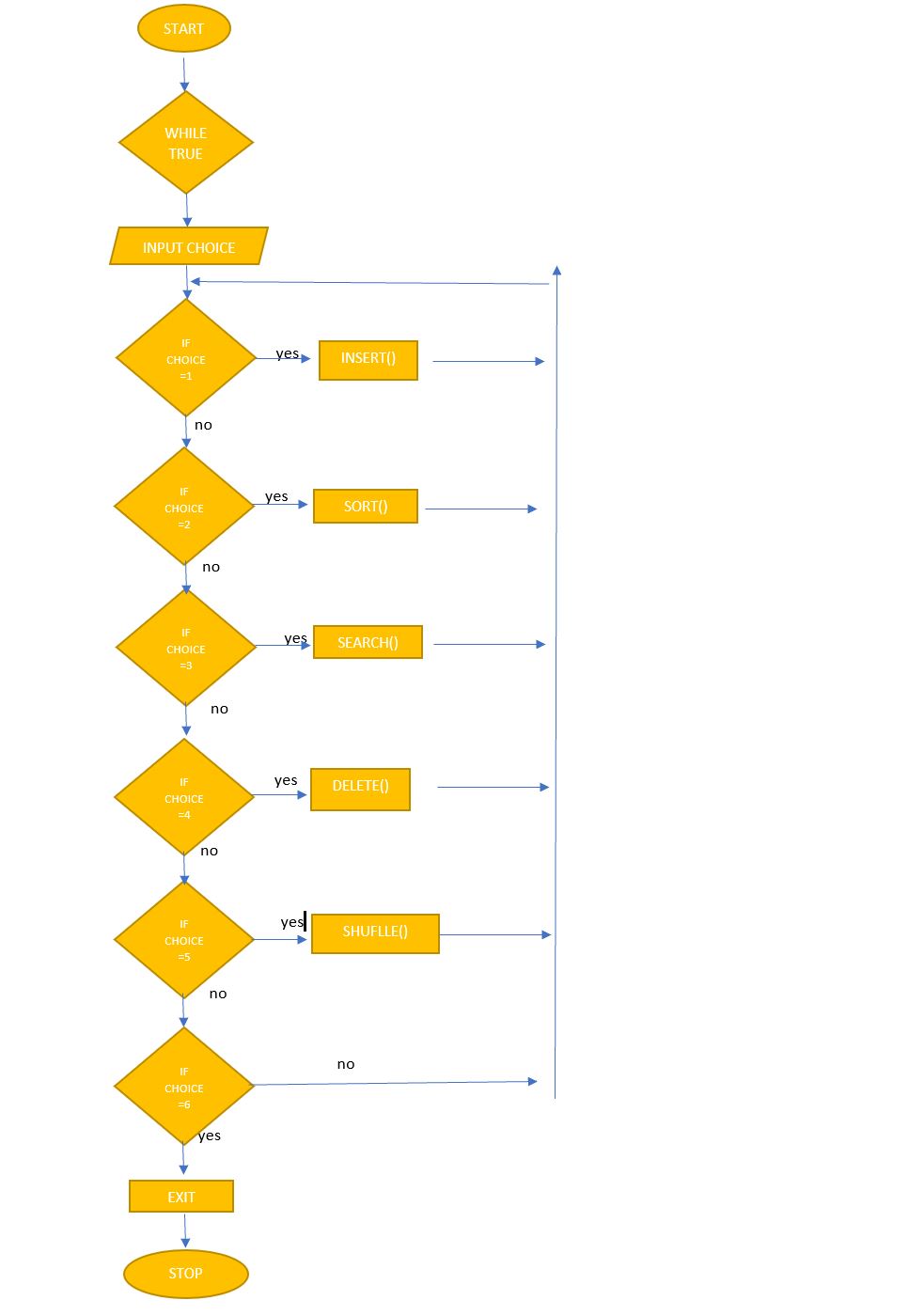
****

Figure 2.3.1

**2.3.2 Algorithm**

1. START
2. Create a structure with parameters song ,artist and genre
3. Declare necessary variables for the program
4. Display message for application for introduction
5. Input Username and Password and confirm it
6. Display options for the actions to be performed in the application
7. Start a while loop to regenerate options until while condition fails
8. If Option 1 is selected
   * + Dynamically allocate memory to the structure by inputting the value for n from the user and reallocate second time if new value of n is detected.
     + Input the song name, artist name and genre from the user.
9. If Option 2 is selected

* + - Ask user as to how they would like to view the song details in the order of artist, song or genre.

1. If Option 3 is selected
   * + Input the song, artist or genre from the user that they would like to search and the display the corresponding result. If no matches are present display appropriate message.
2. If Option 4 is selected
   * + Input the song number from the user as to which they would like to delete then perform delete operation on the structure, is the number does not exits display appropriate message.
3. If Option 5 is selected
   * + Use the randomize function to randomly print a song in the program from the data inputted from the user.
4. If Option 6 is selected
   * + Display thank you message and end the while condition , thereby ending the program
5. If none of the above options are selected then display default message
6. STOP

**2.3.3 Data Structure**

1. Structures | Linked List : Using the concept of Linked Lists in my program I have dynamically allocated my structures using malloc and realloc functions thereby creating a linked list of sorts but I haven’t stored the address of any location so I call it a partial linked list. Inside the structure is present three data fields song, artist and genre.
2. Arrays: To store the data entered from the user in the program I Use 3 arrays inside the structure to store the data.
3. Bubble sort: Using this classic sorting technique of bubble sort is sort my data entered from the user when called.
4. Liner search: Linear search is used in the program to search for a particular item entered by the user.

**Chapter 3**

**IMPLEMENTATION**

**3.1 Dynamic Memory allocation**

Allocating memory dynamically is one of the most important things to do for any real world application. Normally in a program while using data structures we tend to use fixed values which makes it less user friendly. Hence by using memory allocation techniques like malloc (), calloc (), realloc () and free (), we can overcome this issue.

The program uses malloc , realloc and free.

Syntax and functionality

pointer = (data type \*)malloc(size \* sizeof(data type);

malloc assigns a block of memory every time it is called , the block of memory is filled with junk values.

pointer= (data type\*)realloc(newsize \* sizeof(data type);

realloc reassigns the same pointer with a new block of memory.

free(pointer)

free() frees the assign blocks of memory so that to empty the stored values.

**3.2 Structures**

In order to store multiple inputs from the user in a single unit a structure is used. This structure was dynamically allocated so as to let the user multiple inputs as per his/her wish. The structures contain multiple data types to store different parameters.

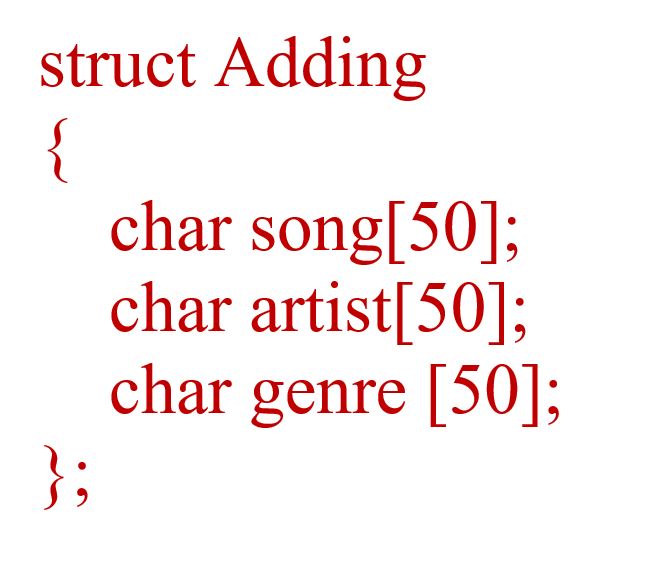


Figure 3.2

**3.3 Sorting**

One of the features of the application is to sort the music list in any desired order the user decides, in order to do this a bubble sorting technique is implemented. This technique compares two data elements and decides which is smaller and then replaces it and then compares that with the next item and so on.

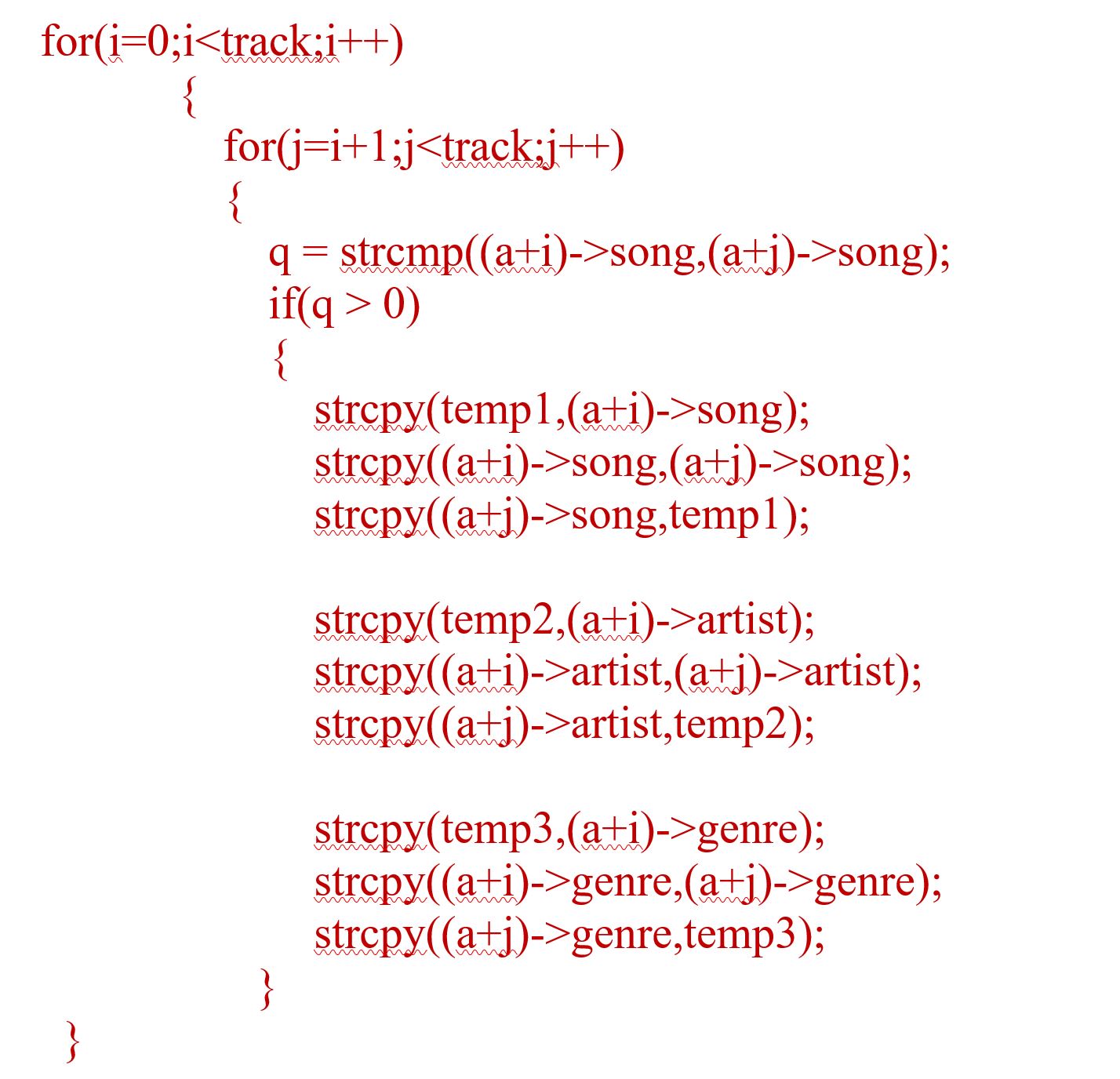


Figure 3.3

**3.4 Linear Search**

A feature of the music application is to search the songs, artists and genre and display the required output. In order to do this a linear search technique is implemented where the user inputs what he wants to search and the program finds for that particular keyword and then displays the output.

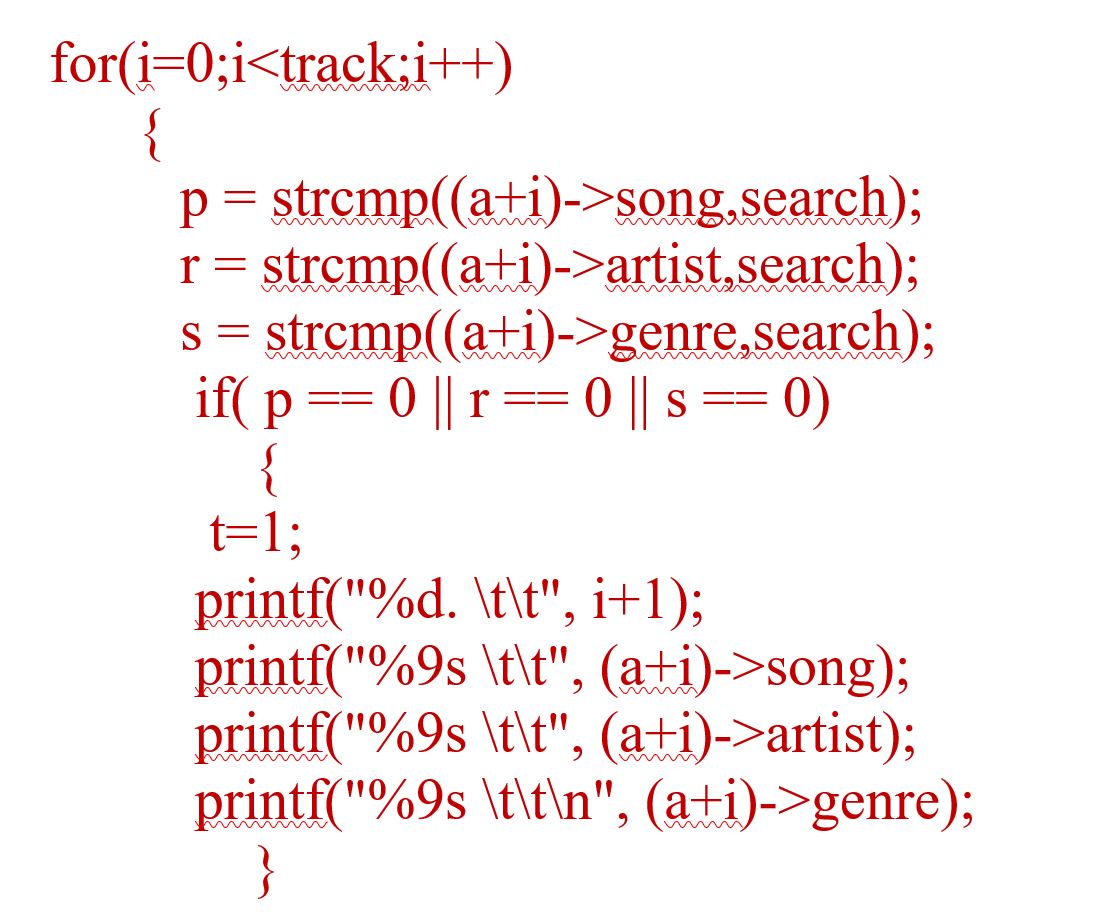


Figure 3.4

**3.5 Delete**

The user can input which line number he wants to delete and the program then searches for the line number entered by the user by implementing linear search and then copies the next element to the previous element to the current one until it reaches the last line.

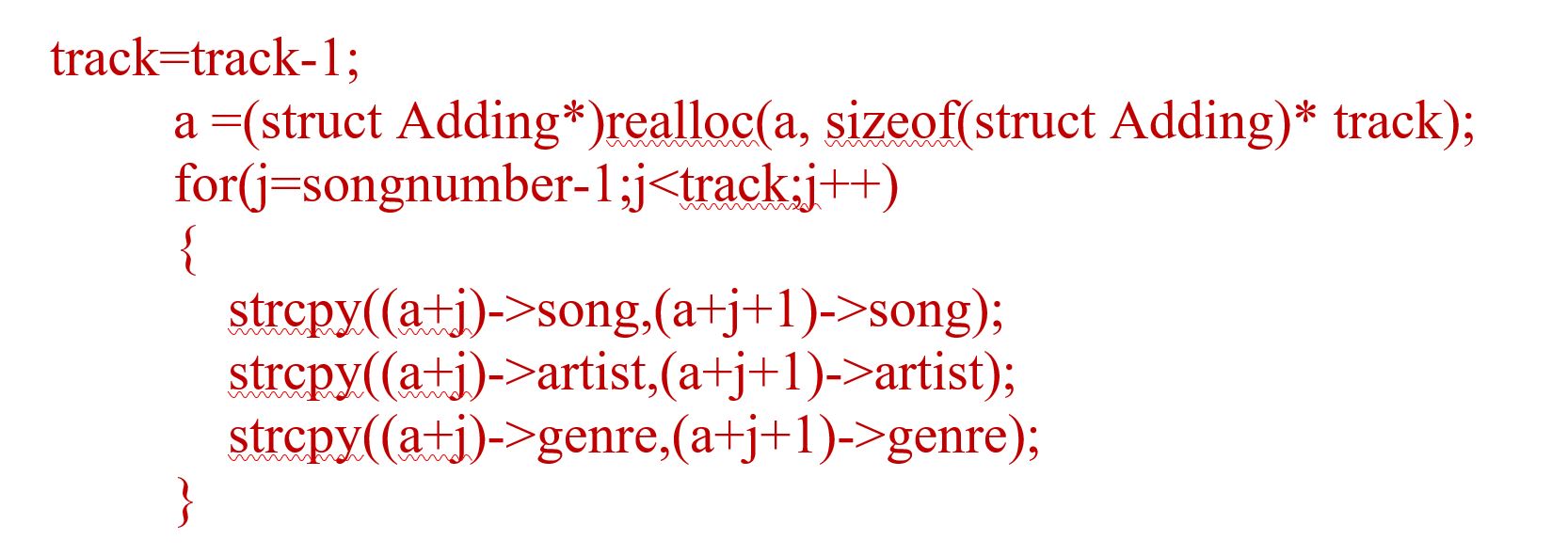


Figure 3.5

**3.6 Shuffle**

Shuffle is a way to randomly print out a song of based on the inputs entered by the user. C has a built in randomize function which randomly generates a number from 1 to the total number of songs present in the application and then prints out the line with the details of the song name , artist and genre.

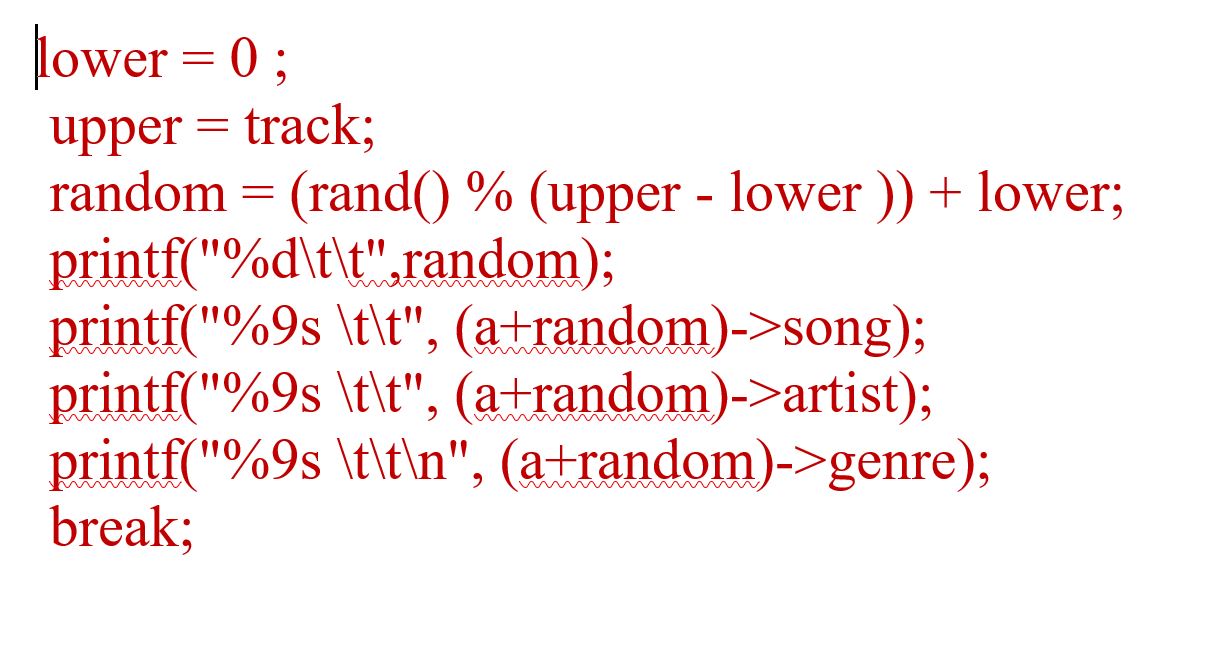
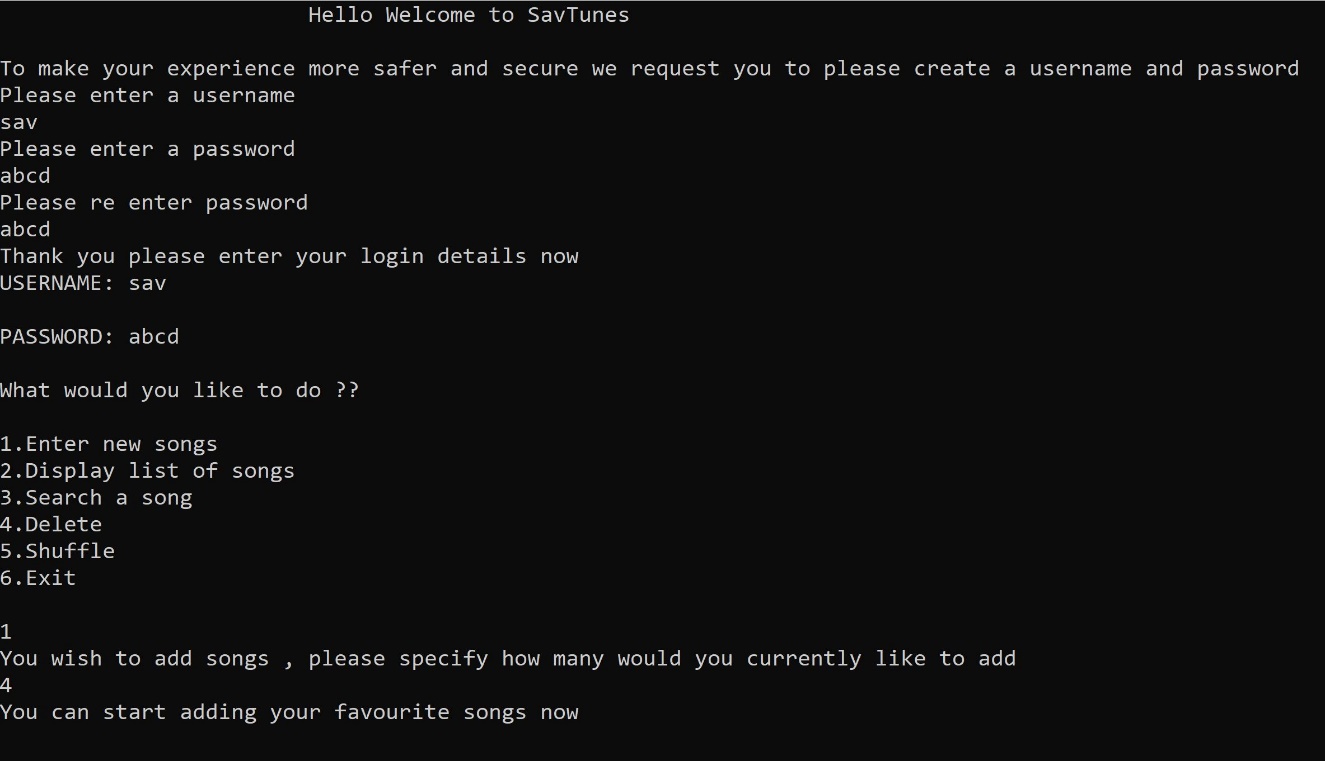
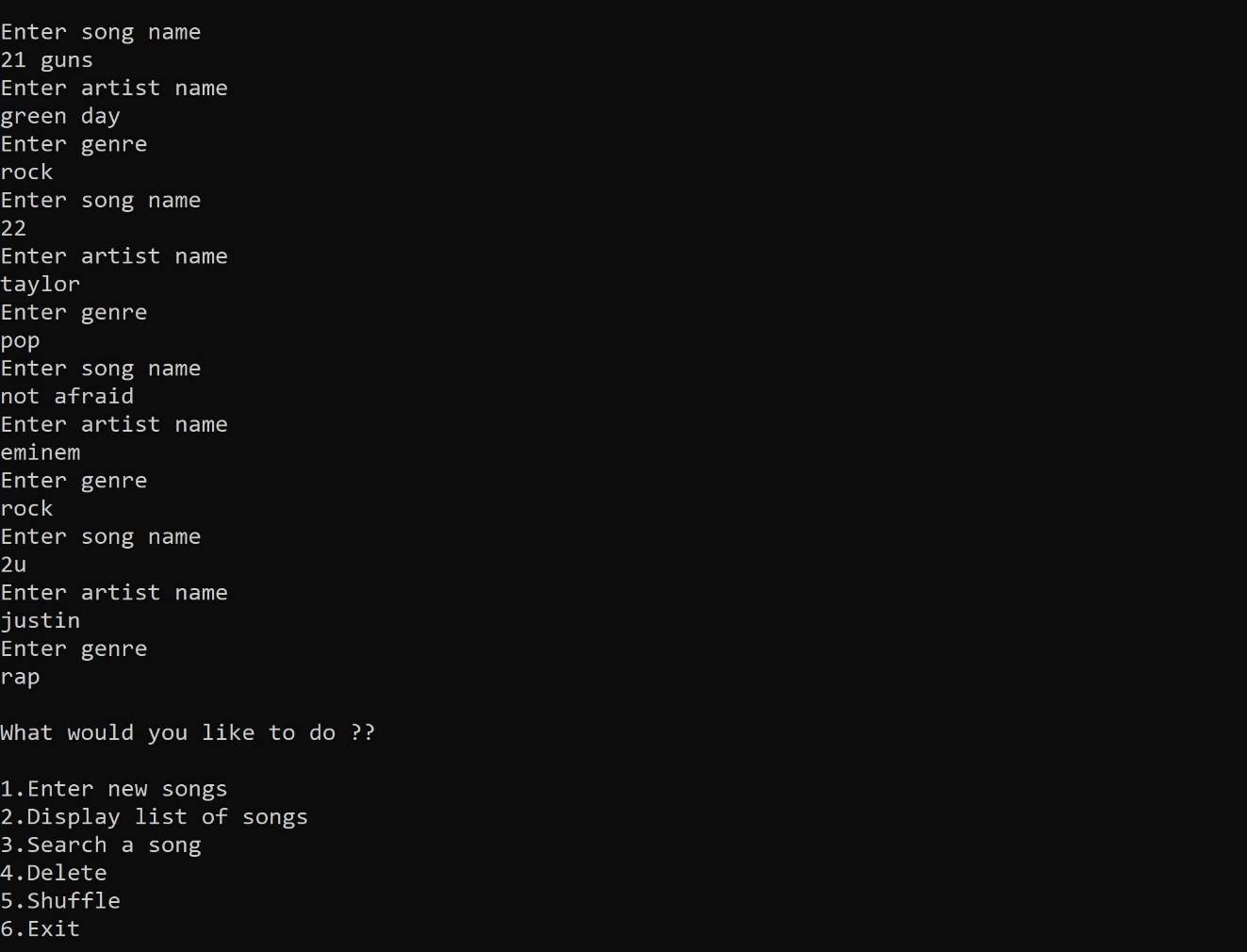


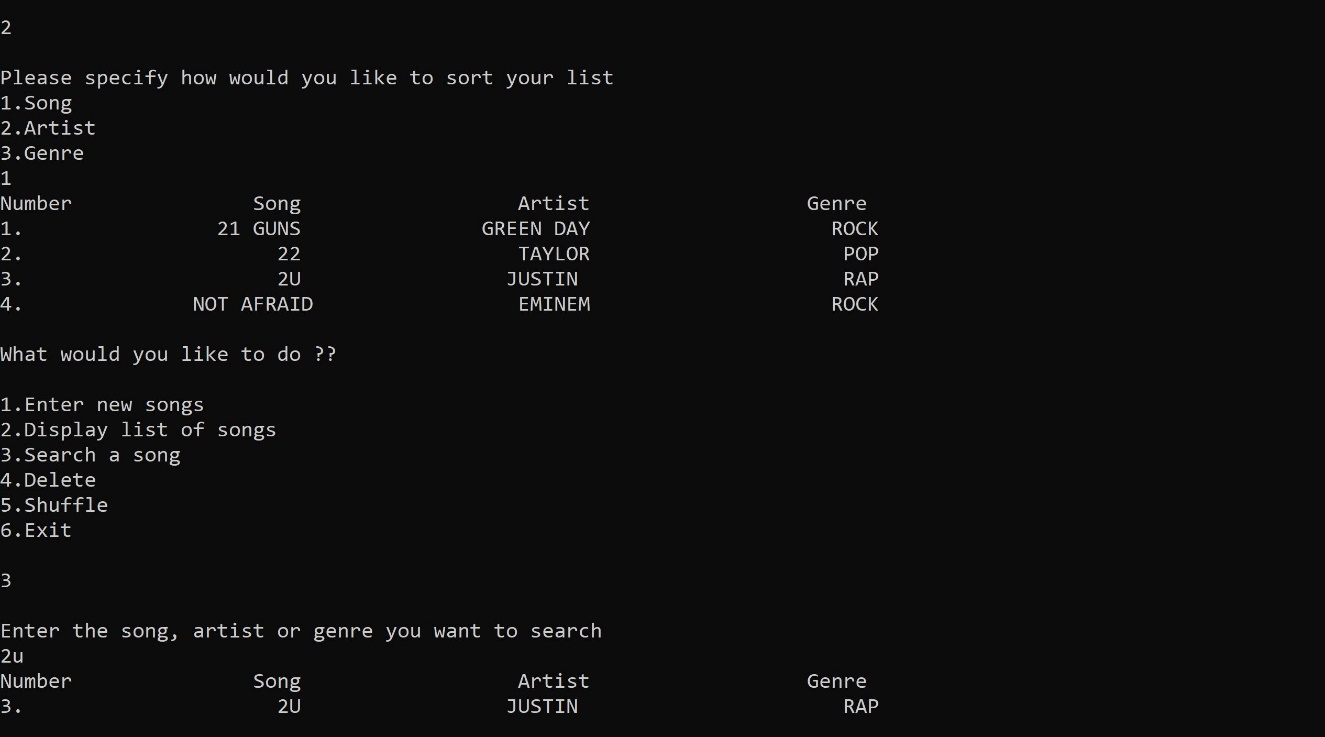
Figure 3.6

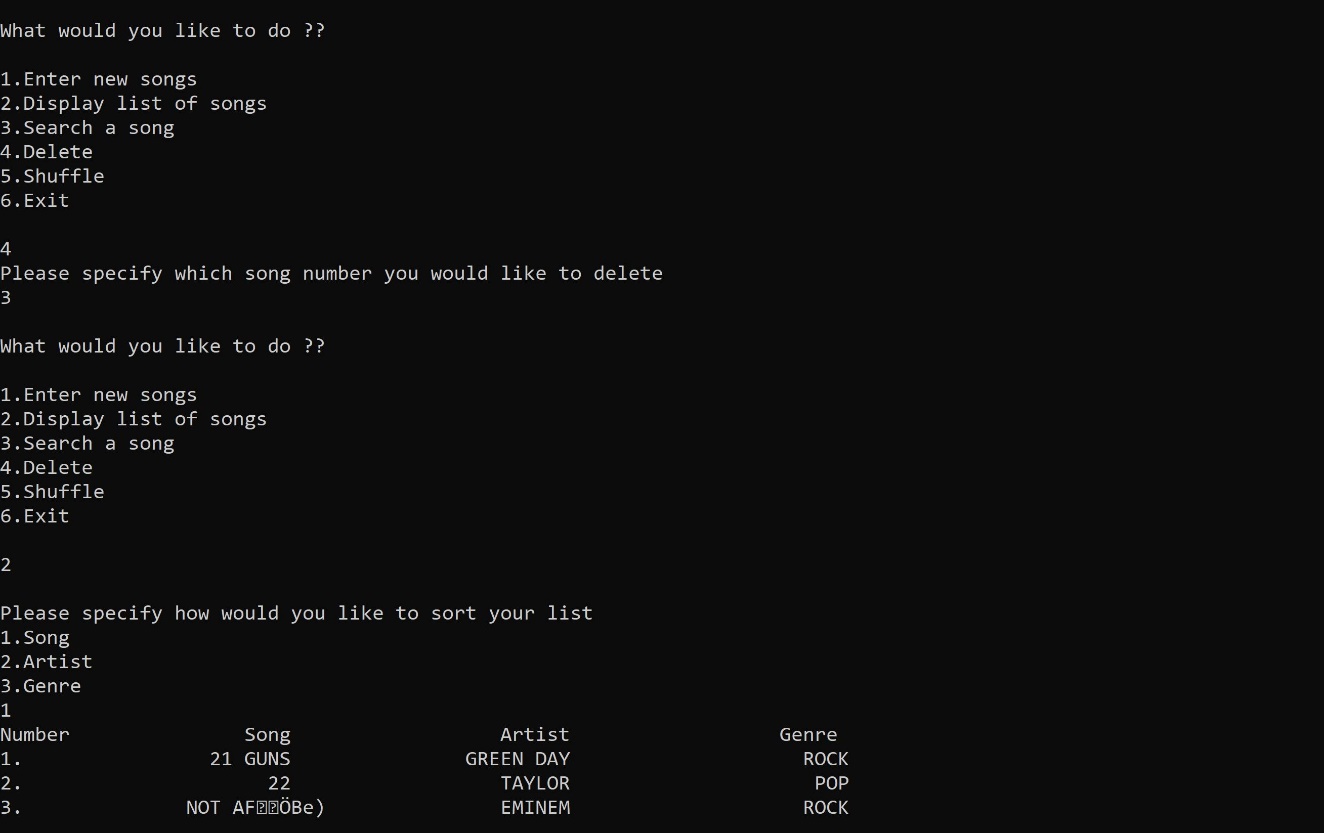
**Chapter 4**

**OUTPUTS**

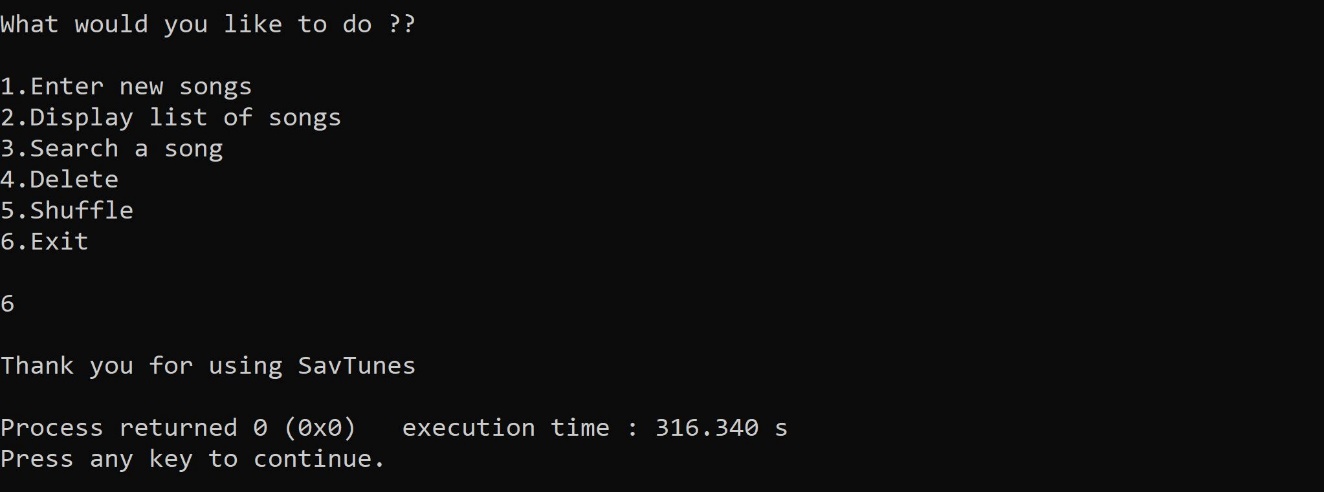




****







**Chapter 5**

**CONCLUSION**

The goal of SavTunes is to let the user experience using a music application which is essentially a watered down version of the current existing ones like Itunes.

SavTunes makes use of the dynamic memory allocation in C and allows the user to add n number of songs. The other functionalities such as sorting, search, delete and shuffle make use of the data structures and techniques available in C.

SavTunes was inspired by Apple’s Itunes as a curiosity to understand the working of a music application. The application will continue further development in the future to make it more suitable for the current generation and to be used as a real world application.

**BIBLIOGRAPHY**

1. C programming language by Dennis Ritchie

2. Geeks for Geeks

3. Stackoverflow