International Journal of Advance Research in Science and Engineering
Vol. No.6, Issue No. 04, April 2017
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

NEW IMPLEMENTATION OF DIGITAL JUKEBOX IN DATA STRUCTURE BY USING C++ PROGRAMMING LANGUAGE

Manigandan Raamanathan

Student, BSc in Software Engineering, Asia Pacific University, Kuala Lumpur, Malaysia

ABSTRACT

Data structures is a popular method used in organizing data in a manner where it deals about searching for the best ways for the storing of data. It also talks about how to store a cluster of objects in the memory, the operations that can be done on the algorithms, the algorithms for the operations and also how efficient is it in saving spaces and time. In this paper, for the creating of "Digital Jukebox", the implementation of linked list using stack and queue is needed by using C++. To flexibility of customers here will be able to pick songs from a list of songs that are displayed as well as they are able to view it. The main purpose in to help Burpee Burgers in changing the traditional playlist into a more revolutionary playlist using stack and queue.

Keywords: Algorithm, Data Structure, Jukebox, Linked List, Queue, Stack

I. INTRODUCTION

Data structures are used in almost all application in the hope of enhancing the accuracy and usefulness of it. There are two main function of using data structures that is putting data in and removing data from the application incorporating it. The most common data structures are linked list, arrays, queues, stacks, trees and graphs. It is very much helpful in managing memory better in applications using data structures. Other than that, the song of the year 2016 are charged RM 1 and will be placed on the top while songs of the other years are free. The main purpose in to help Burpee Burgers in changing the traditional playlist into a more revolutionary playlist using stack and queue.

II. DATA STRUCTURES

2.1 Linked List

Linked List is a linear data structure and it is very common data structure which consists of group of nodes in a sequence which is divided in two parts. Each node consists of its own data and the address of the next node and forms a chain. Linked Lists are used to create trees and graphs.Linked list is concept where it is defined as a linear data structure which is most commonly used in building applications where it consists of nodes which hold their data as well as the particulars of another node. It is also used as an alternative to an array as it helps in using memory freely whereas an array is only fixed to a size [1].

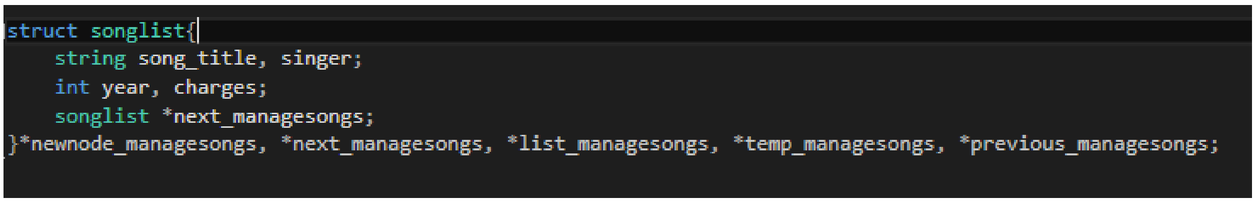
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

2.2 Queue

Both stacks and queues are like lists (ordered collections of items), but with more restricted operations. They can both be implemented either using an array or using a linked list to hold the actual items. Stack is stack is an ordered list in which all insertions and deletions are made at one end, called the top. A queue is an ordered list in which all insertions take place at one end, the rear, while all deletions take place at the other end, the front [2].

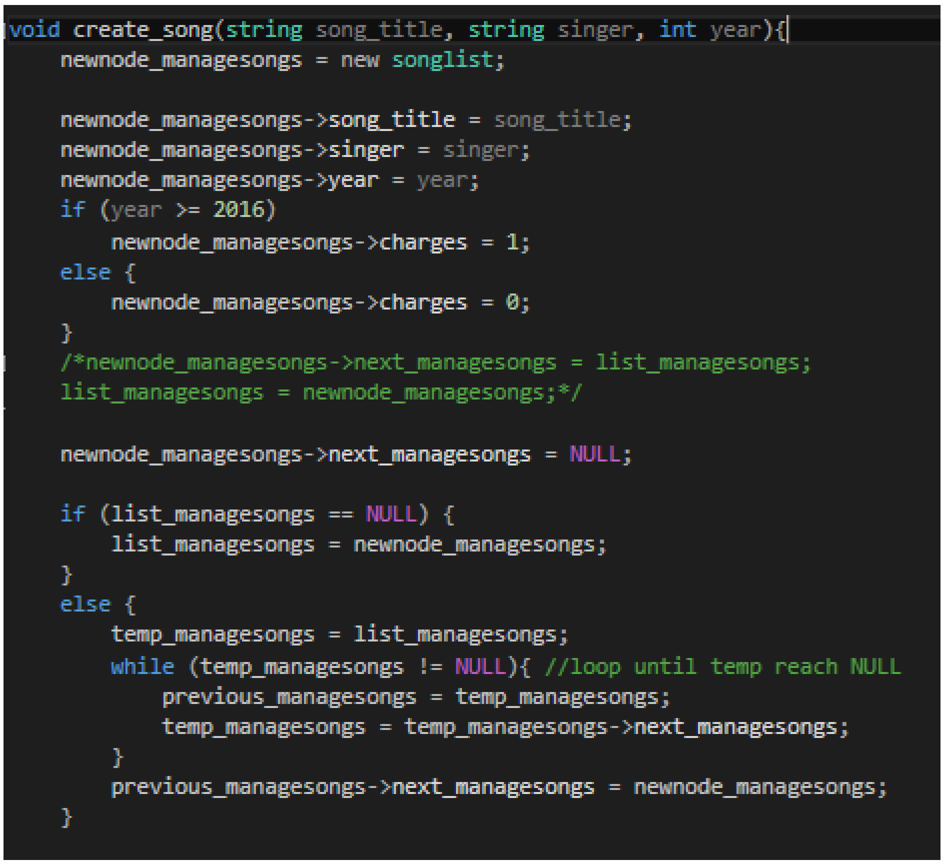
III. IMPLEMENTATION

3.1 Song List



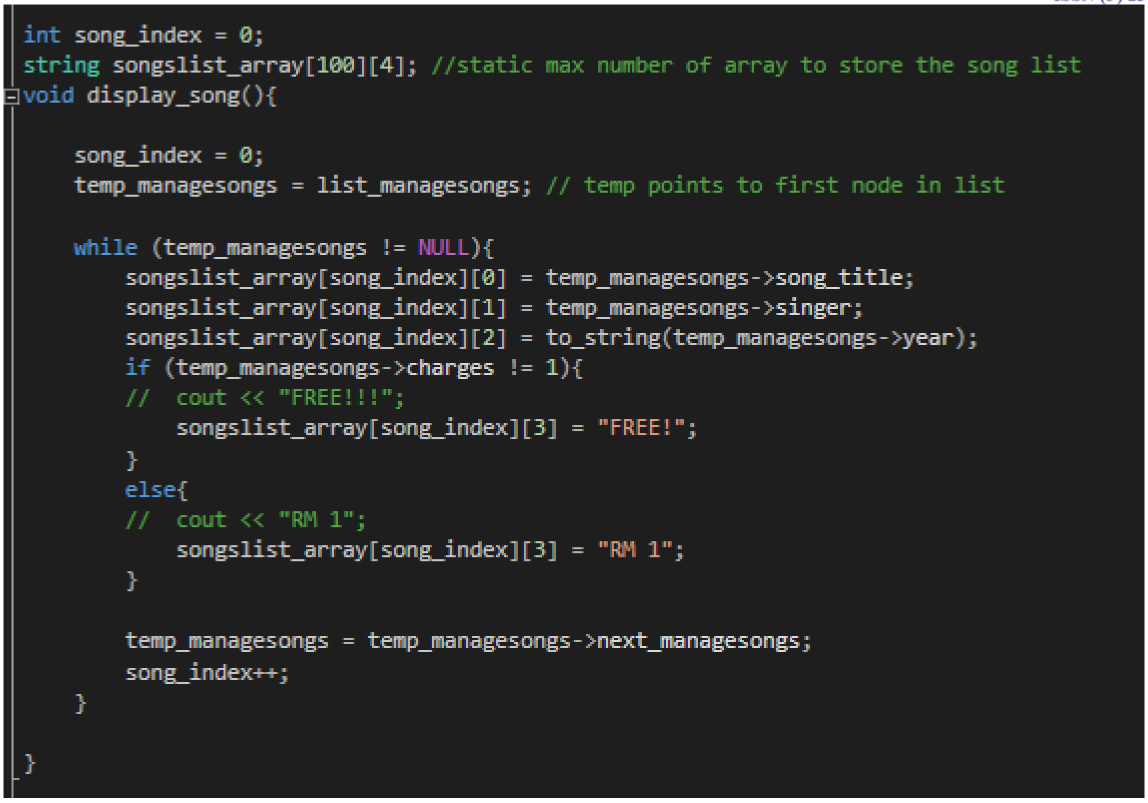
```
struct songlist{
    string song_title, singer;
    int year, charges;
    songlist *next_managesongs;
}*newnode_managesongs, *next_managesongs, *list_managesongs, *temp_managesongs, *previous_managesongs;
```

The above shows the structure that are used for song list.



```
void create_song(string song_title, string singer, int year){
    newnode_managesongs = new songlist;

    newnode_managesongs->song_title = song_title;
    newnode_managesongs->singer = singer;
    newnode_managesongs->year = year;
    if (year >= 2016)
        newnode_managesongs->charges = 1;
    else {
        newnode_managesongs->charges = 0;
    }
    /*newnode_managesongs->next_managesongs = list_managesongs;
    list_managesongs = newnode_managesongs;*/

    newnode_managesongs->next_managesongs = NULL;

    if (list_managesongs == NULL) {
        list_managesongs = newnode_managesongs;
    }
    else {
        temp_managesongs = list_managesongs;
        while (temp_managesongs != NULL){ //loop until temp reach NULL
            previous_managesongs = temp_managesongs;
            temp_managesongs = temp_managesongs->next_managesongs;
        }
        previous_managesongs->next_managesongs = newnode_managesongs;
    }
```

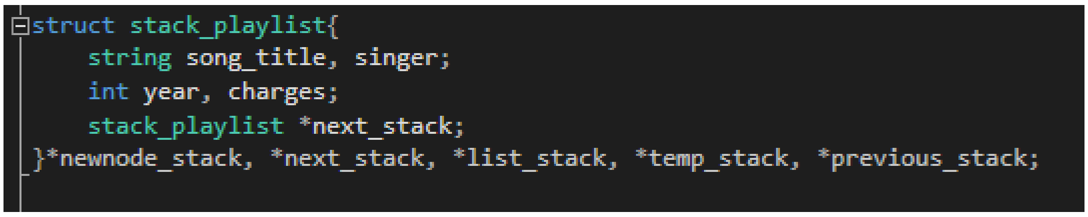
The above code shows the create song function that are used to insert the newnode which is new songs that are being created to pass into it using the queue method which is by adding the nodes at the end of all other nodes. In this way we can keep track of the song lists based smaller to larger index number. Here, the data are added into newnode pointer and then the system assigning the charges of the songs based on the year.



```
int song_index = 0;
string songslist_array[100][4]; //static max number of array to store the song list
void display_song(){

    song_index = 0;
    temp_managesongs = list_managesongs; // temp points to first node in list

    while (temp_managesongs != NULL){
        songslist_array[song_index][0] = temp_managesongs->song_title;
        songslist_array[song_index][1] = temp_managesongs->singer;
        songslist_array[song_index][2] = to_string(temp_managesongs->year);
        if (temp_managesongs->charges != 1){
        //  cout << "FREE!!!";
            songslist_array[song_index][3] = "FREE!";
        }
        else{
        //  cout << "RM 1";
            songslist_array[song_index][3] = "RM 1";
        }

        temp_managesongs = temp_managesongs->next_managesongs;
        song_index++;
    }

}
```

Above shows the functions of display song list codes. Here we are saying the temporary pointer to point to list pointer which always point to first node. Then, song details are inserted into array using while loop and where later will be displayed and decorated in the main class.

3.2 Stack

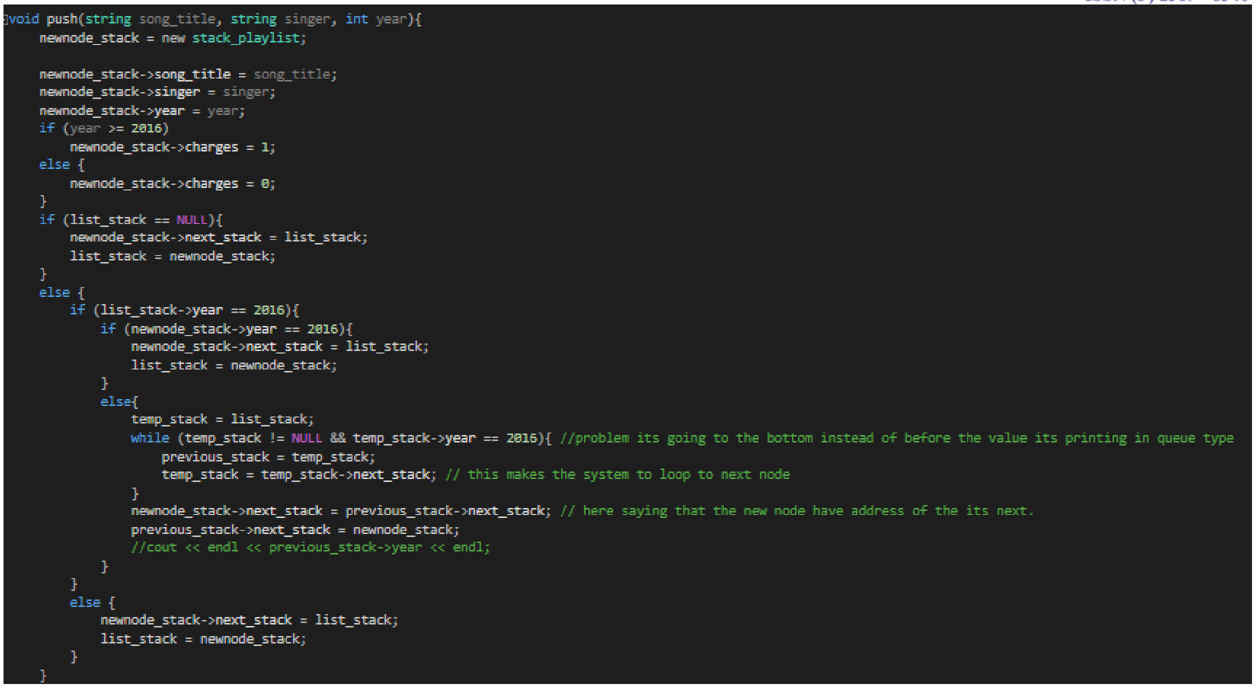


```
struct stack_playlist{
    string song_title, singer;
    int year, charges;
    stack_playlist *next_stack;
}*newnode_stack, *next_stack, *list_stack, *temp_stack, *previous_stack;
```

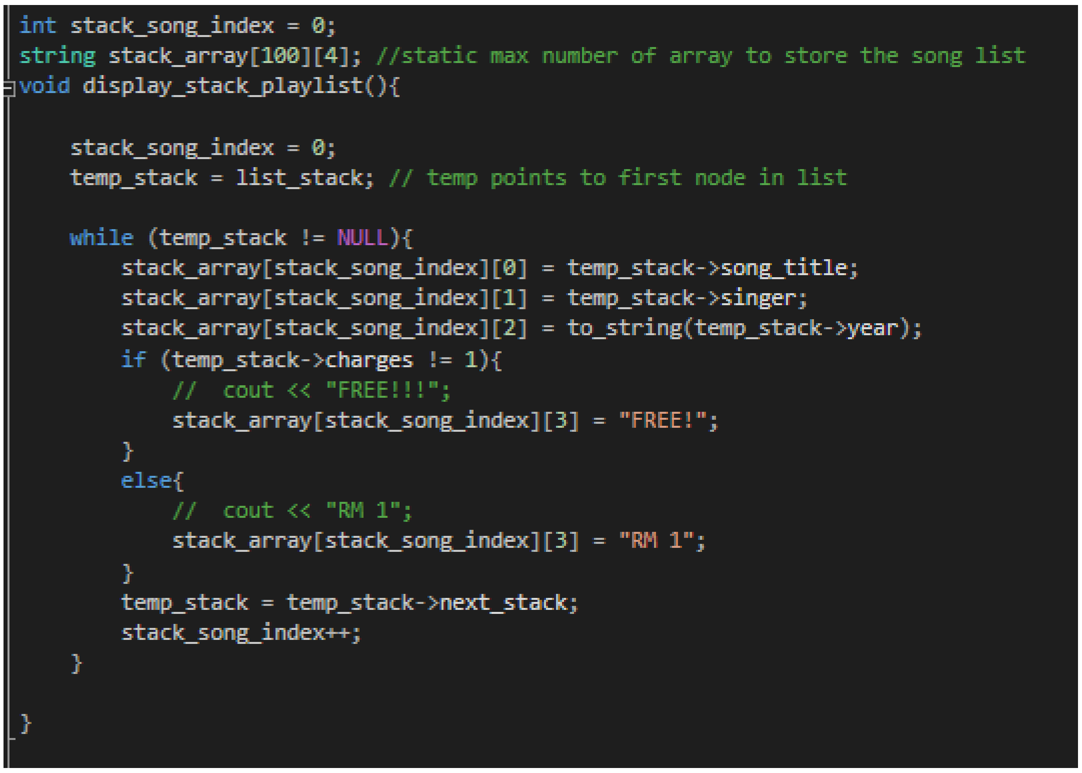
The above shown structure is used for stack playlist.

The below shows the push function that are used for the system to insert the song into the system. Here the system check whether the linked list is empty or not. If the linked list is empty and then the songs will be added as normal. If the entered song is year 2016 then the songs will be inserted into the beginning. The system will also search for the traces of 2016 song before inserting in the list, if the 2016 node found and then the system will run a while to find the node that are not 2016 and then insert the song that are less than 2016.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

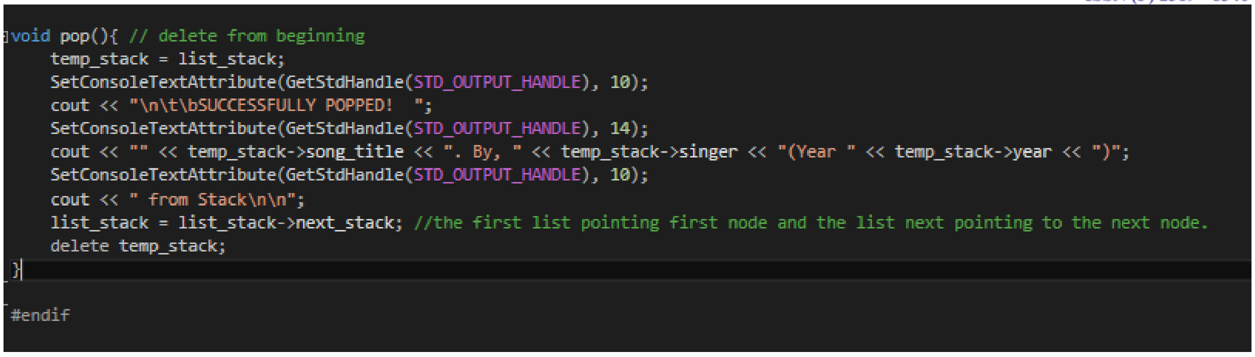


```
void push(string song_title, string singer, int year){
    newnode_stack = new stack_playlist;

    newnode_stack->song_title = song_title;
    newnode_stack->singer = singer;
    newnode_stack->year = year;
    if (year >= 2016)
        newnode_stack->charges = 1;
    else {
        newnode_stack->charges = 0;
    }
    if (list_stack == NULL){
        newnode_stack->next_stack = list_stack;
        list_stack = newnode_stack;
    }
    else {
        if (list_stack->year == 2016){
            if (newnode_stack->year == 2016){
                newnode_stack->next_stack = list_stack;
                list_stack = newnode_stack;
            }
            else{
                temp_stack = list_stack;
                while (temp_stack != NULL && temp_stack->year == 2016){ //problem its going to the bottom instead of before the value its printing in queue type
                    previous_stack = temp_stack;
                    temp_stack = temp_stack->next_stack; // this makes the system to loop to next node
                }
                newnode_stack->next_stack = previous_stack->next_stack; // here saying that the new node have address of the its next.
                previous_stack->next_stack = newnode_stack;
                //cout << endl << previous_stack->year << endl;
            }
        }
        else {
            newnode_stack->next_stack = list_stack;
            list_stack = newnode_stack;
        }
    }
```



```
int stack_song_index = 0;
string stack_array[100][4]; //static max number of array to store the song list
void display_stack_playlist(){

    stack_song_index = 0;
    temp_stack = list_stack; // temp points to first node in list

    while (temp_stack != NULL){
        stack_array[stack_song_index][0] = temp_stack->song_title;
        stack_array[stack_song_index][1] = temp_stack->singer;
        stack_array[stack_song_index][2] = to_string(temp_stack->year);
        if (temp_stack->charges != 1){
            //  cout << "FREE!!!";
            stack_array[stack_song_index][3] = "FREE!";
        }
        else{
            //  cout << "RM 1";
            stack_array[stack_song_index][3] = "RM 1";
        }
        temp_stack = temp_stack->next_stack;
        stack_song_index++;
    }

}
```

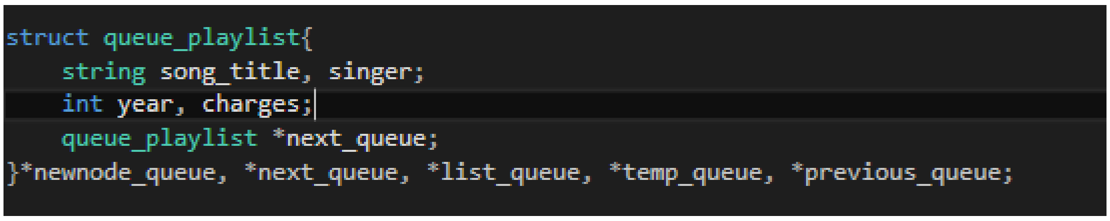
The above shown is the function used to display the stack playlist to the main menu. Here we are using a while loop and print the songs. Where, at the same time the songs are inserted into the array song that in future can be used to enter in main screen to show side by side with different play list.



```
void pop(){ // delete from beginning
    temp_stack = list_stack;
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 10);
    cout << "\n\t\bSUCCESSFULLY POPPED!  ";
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 14);
    cout << "" << temp_stack->song_title << ". By, " << temp_stack->singer << "(Year " << temp_stack->year << ")";
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 10);
    cout << " from Stack\n\n";
    list_stack = list_stack->next_stack; //the first list pointing first node and the list next pointing to the next node.
    delete temp_stack;
}

#endif
```

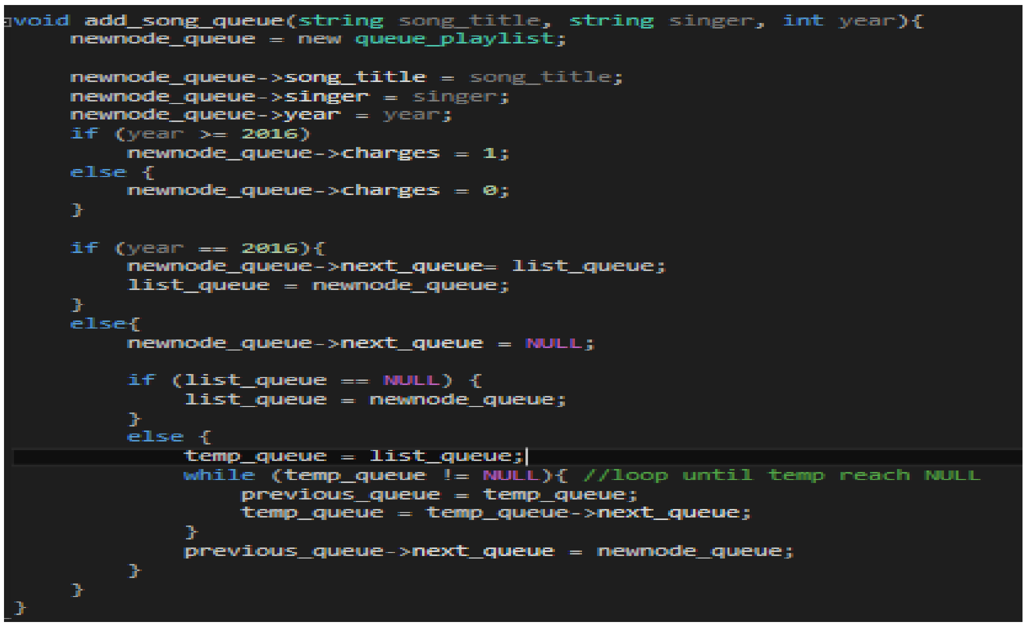
The above shown are the code that used for the system to pop or remove the first node from the linked list when the user pressed delete from the main class. Once the node is deleted then the system will print out successful message.

3.3 Queue



```
struct queue_playlist{
    string song_title, singer;
    int year, charges;
    queue_playlist *next_queue;
}*newnode_queue, *next_queue, *list_queue, *temp_queue, *previous_queue;
```

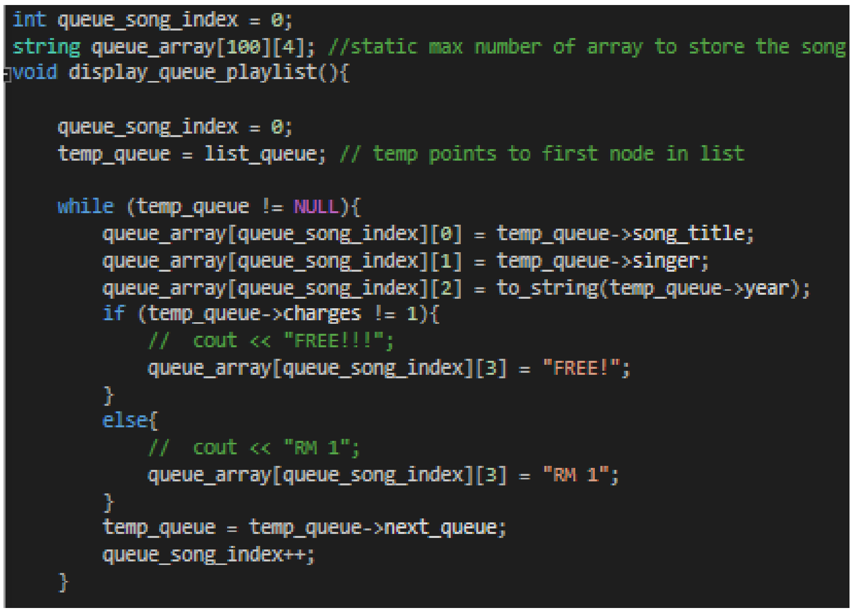
The above shown is the structure that are used for queue linked list.



```
void add_song_queue(string song_title, string singer, int year){
    newnode_queue = new queue_playlist;

    newnode_queue->song_title = song_title;
    newnode_queue->singer = singer;
    newnode_queue->year = year;
    if (year >= 2016)
        newnode_queue->charges = 1;
    else {
        newnode_queue->charges = 0;
    }

    if (year == 2016){
        newnode_queue->next_queue= list_queue;
        list_queue = newnode_queue;
    }
    else{
        newnode_queue->next_queue = NULL;

        if (list_queue == NULL) {
            list_queue = newnode_queue;
        }
        else {
            temp_queue = list_queue;
            while (temp_queue != NULL){ //loop until temp reach NULL
                previous_queue = temp_queue;
                temp_queue = temp_queue->next_queue;
            }
            previous_queue->next_queue = newnode_queue;
        }
    }
}
```

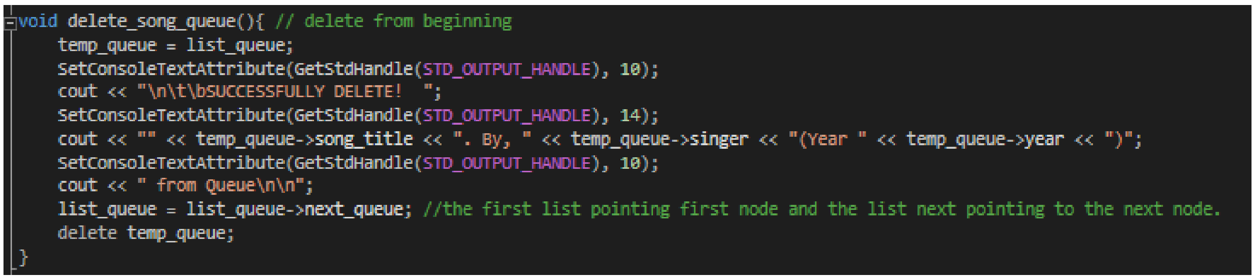
Here the above shown function is the function that are used to add song nodes to insert into the queue linked list. Here the system will be checking whether the song year is 2016 or not, if the song year is equal to 2016 then the system will add that node in the beginning of the linked list. If the song is not 2016 or less, then the system will add that song into the end of the linked list.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346



```
int queue_song_index = 0;
string queue_array[100][4]; //static max number of array to store the song
void display_queue_playlist(){

    queue_song_index = 0;
    temp_queue = list_queue; // temp points to first node in list

    while (temp_queue != NULL){
        queue_array[queue_song_index][0] = temp_queue->song_title;
        queue_array[queue_song_index][1] = temp_queue->singer;
        queue_array[queue_song_index][2] = to_string(temp_queue->year);
        if (temp_queue->charges != 1){
            //  cout << "FREE!!!";
            queue_array[queue_song_index][3] = "FREE!";
        }
        else{
            //  cout << "RM 1";
            queue_array[queue_song_index][3] = "RM 1";
        }
        temp_queue = temp_queue->next_queue;
        queue_song_index++;
    }
```

The above shown is the function used to display the queue playlist to the main menu. Here we are using a while loop and print the songs. Where, at the same time the songs are inserted into the array song that in future can be used to enter in main screen to show side by side with different play list.

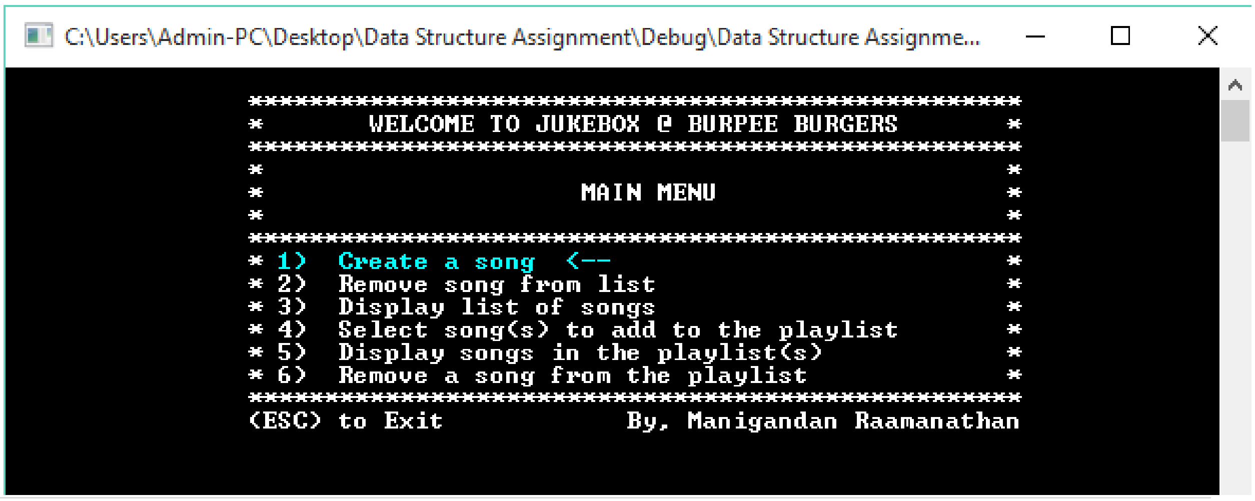


```
void delete_song_queue(){ // delete from beginning
    temp_queue = list_queue;
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 10);
    cout << "\n\t\bSUCCESSFULLY DELETE!  ";
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 14);
    cout << "" << temp_queue->song_title << ". By, " << temp_queue->singer << "(Year " << temp_queue->year << ")";
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 10);
    cout << " from Queue\n\n";
    list_queue = list_queue->next_queue; //the first list pointing first node and the list next pointing to the next node.
    delete temp_queue;
}
```

The above shown are the code that used for the system to remove the first node from the linked list when the user pressed delete from the main class. Once the node is deleted then the system will print out successful message.

IV. PROGRAM OUTPUTS

4.1 Main Menu

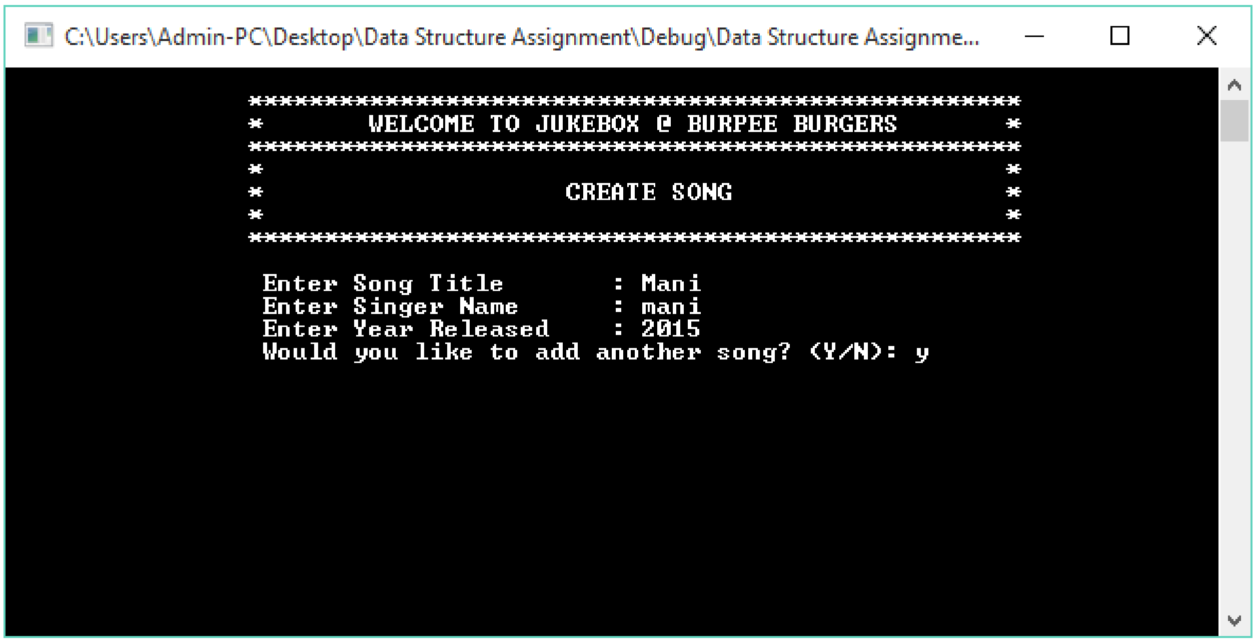


```
C:\Users\Admin-PC\Desktop\Data Structure Assignment\Debug\Data Structure Assignme...

***************************************************
*       WELCOME TO JUKEBOX @ BURPEE BURGERS       *
***************************************************
*                                                 *
*                    MAIN MENU                    *
*                                                 *
***************************************************
* 1)  Create a song  <--                          *
* 2)  Remove song from list                       *
* 3)  Display list of songs                       *
* 4)  Select song(s) to add to the playlist       *
* 5)  Display songs in the playlist(s)            *
* 6)  Remove a song from the playlist             *
***************************************************
(ESC) to Exit           By, Manigandan Raamanathan
```

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

Above shows the view of the main menu where user are able to select their desired options using the up and down arrows based on the color changes in the main menu.

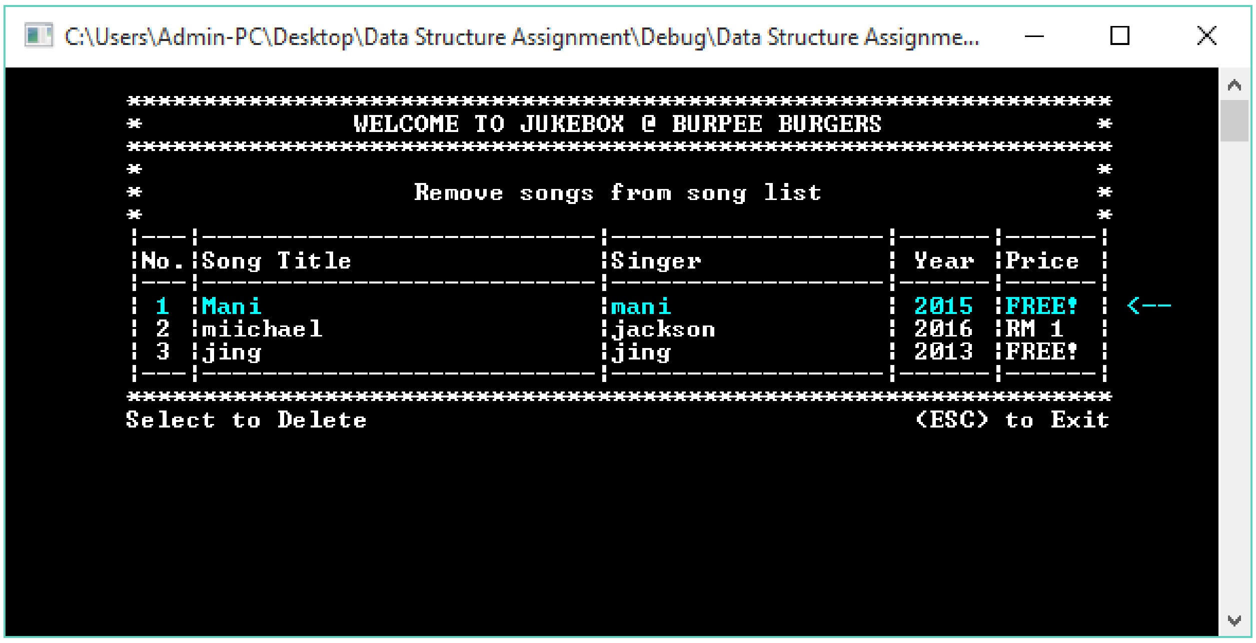
4.2 Create Song



```
C:\Users\Admin-PC\Desktop\Data Structure Assignment\Debug\Data Structure Assignme...

****************************************************
*       WELCOME TO JUKEBOX @ BURPEE BURGERS        *
****************************************************
*                                                  *
*                    CREATE SONG                   *
*                                                  *
****************************************************

 Enter Song Title        : Mani
 Enter Singer Name       : mani
 Enter Year Released     : 2015
 Would you like to add another song? (Y/N): y
```

The above shows the create song panel where the year having validation to keep the user from entering

4.3 Remove Song from Song List

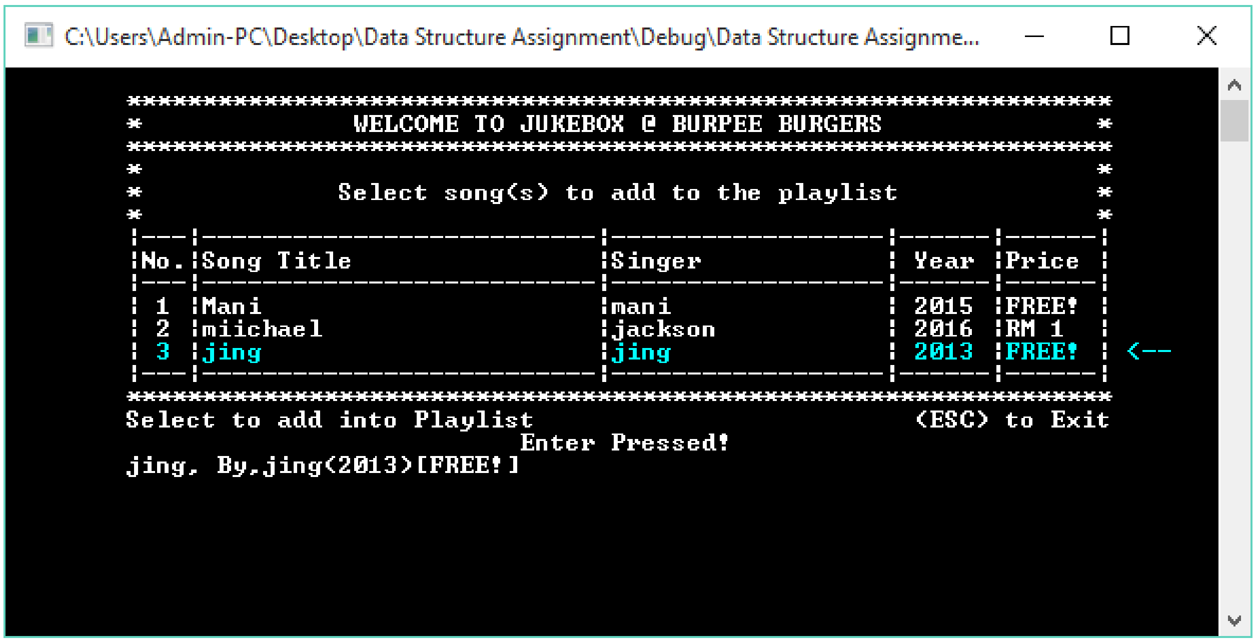


```
C:\Users\Admin-PC\Desktop\Data Structure Assignment\Debug\Data Structure Assignme...

******************************************************************
*               WELCOME TO JUKEBOX @ BURPEE BURGERS              *
******************************************************************
*                                                                *
*                    Remove songs from song list                 *
*                                                                *
|---|--------------------------|-------------------|------|------|
|No.|Song Title                |Singer             | Year |Price |
|---|--------------------------|-------------------|------|------|
| 1 |Mani                      |mani               | 2015 |FREE! | <--
| 2 |miichael                  |jackson            | 2016 |RM 1  |
| 3 |jing                      |jing               | 2013 |FREE! |
|---|--------------------------|-------------------|------|------|
******************************************************************
Select to Delete                                     (ESC) to Exit
```

The above shows, the panel that allows the user to remove the songs in the songs list using the arrow which is operated using the up and down arrows.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

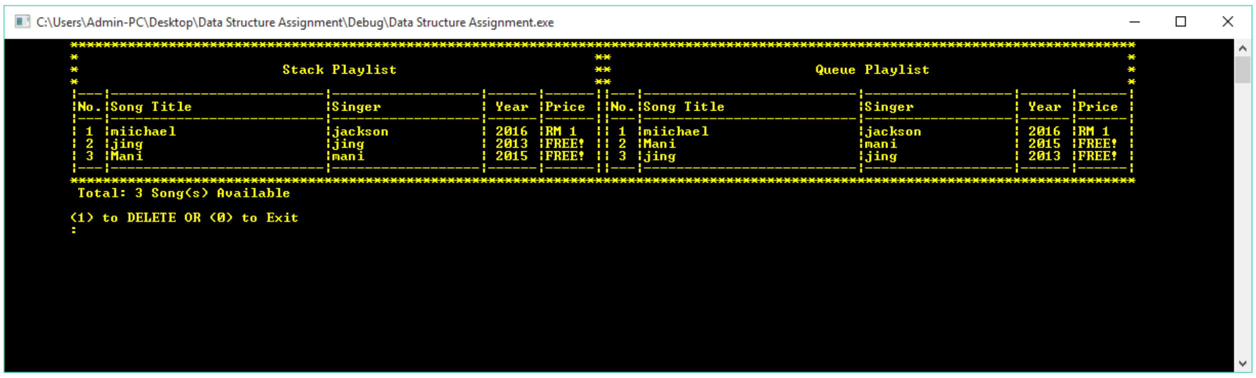
4.4 Select Song to Add in Playlist



```
C:\Users\Admin-PC\Desktop\Data Structure Assignment\Debug\Data Structure Assignme...

****************************************************************
*                WELCOME TO JUKEBOX @ BURPEE BURGERS            *
****************************************************************
*                                                               *
*              Select song(s) to add to the playlist            *
*                                                               *
|---|--------------------------|------------------|------|------|
|No.|Song Title                |Singer            | Year |Price |
|---|--------------------------|------------------|------|------|
| 1 |Mani                      |mani              | 2015 |FREE! |
| 2 |miichael                  |jackson           | 2016 |RM 1  |
| 3 |jing                      |jing              | 2013 |FREE! |  <--
|---|--------------------------|------------------|------|------|
****************************************************************
Select to add into Playlist                          (ESC) to Exit
                          Enter Pressed!
jing, By,jing(2013)[FREE!]
```

The above shows the panel that allows the users to add songs into stack and queue playlist.

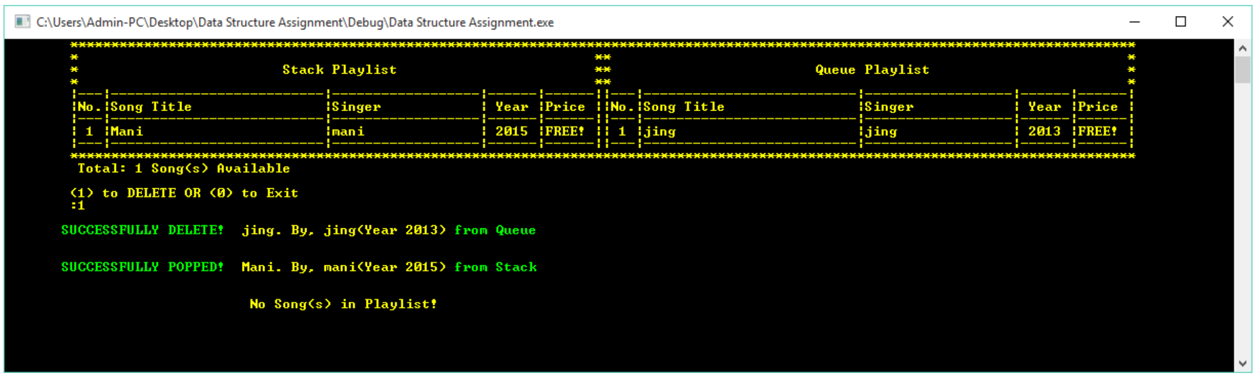
4.5 Display Songs in the Playlist



```
C:\Users\Admin-PC\Desktop\Data Structure Assignment\Debug\Data Structure Assignment.exe

**********************************************************************************************************************************
*                                                                  **                                                            *
*                          Stack Playlist                          **                        Queue Playlist                      *
*                                                                  **                                                            *
|---|--------------------------|------------------|------|------||---|--------------------------|------------------|------|------|
|No.|Song Title                |Singer            | Year |Price ||No.|Song Title                |Singer            | Year |Price |
|---|--------------------------|------------------|------|------||---|--------------------------|------------------|------|------|
| 1 |miichael                  |jackson           | 2016 |RM 1  || 1 |miichael                  |jackson           | 2016 |RM 1  |
| 2 |jing                      |jing              | 2013 |FREE! || 2 |Mani                      |mani              | 2015 |FREE! |
| 3 |Mani                      |mani              | 2015 |FREE! || 3 |jing                      |jing              | 2013 |FREE! |
|---|--------------------------|------------------|------|------||---|--------------------------|------------------|------|------|
**********************************************************************************************************************************
 Total: 3 Song(s) Available

(1) to DELETE OR (0) to Exit
:
```

The panel shows both stack and queue playlist side by side for the users to compare and take decision wisely. This panel will also allow the user to delete the songs.

4.6 Remove Song from the Playlist



```
C:\Users\Admin-PC\Desktop\Data Structure Assignment\Debug\Data Structure Assignment.exe

**********************************************************************************************************************************
*                                                                  **                                                            *
*                          Stack Playlist                          **                        Queue Playlist                      *
*                                                                  **                                                            *
|---|--------------------------|------------------|------|------||---|--------------------------|------------------|------|------|
|No.|Song Title                |Singer            | Year |Price ||No.|Song Title                |Singer            | Year |Price |
|---|--------------------------|------------------|------|------||---|--------------------------|------------------|------|------|
| 1 |Mani                      |mani              | 2015 |FREE! || 1 |jing                      |jing              | 2013 |FREE! |
|---|--------------------------|------------------|------|------||---|--------------------------|------------------|------|------|
**********************************************************************************************************************************
 Total: 1 Song(s) Available

(1) to DELETE OR (0) to Exit
:1

SUCCESSFULLY DELETE!  jing. By, jing(Year 2013) from Queue

SUCCESSFULLY POPPED!  Mani. By, mani(Year 2015) from Stack

                       No Song(s) in Playlist!
```

Shows the user panel that will allow the users to pop/remove the songs directly from the display playlist panel.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

V. CONCLUSION

In conclusion, it can be said that it was very much difficult for me in trying to program what is required for this project of “Digital Jukebox”. However, I manage to complete it using linked list in C++ by paying attention in class and referencing on what that have been taught and learnt into my assignment. Another problem is the understanding of the stack and queue in linked list but I managed to cope with it well. Time constraint was also seen as another limitation. In conclusion, it can be said that I have learnt a lot from this assignment which may be useful later in working life.

VI. ACKNOWLEDGMENT

The author would like to share gratitude to Mr Umapathy Eaganathan, Lecturer in Computing, Asia Pacific University, Malaysia also Miss Angel Aron for her constant support and motivation helped me to participate in this International Conference and also for journal publication.

REFERENCES

[1] Anon., 2007. *linked list basics.* [Online]
Available at: http://cslibrary.stanford.edu/103/LinkedListBasics.pdf
[Accessed 15 August 2016].
[2] Anon., 2005. *Stacks and Queues.* [Online]
Available at: http://www.dcs.gla.ac.uk/~pat/52233/slides/StacksQueuesLists1x1.pdf
[Accessed 15 August 2016].