PIMPRI CHINCHWAD EDUCATION TRUST's.
PIMPRI CHINCHWAD COLLEGE OF ENGINEERING
(An Autonomous Institute)

Class : SY BTech **Acad. Yr. 2025-26** **Semester :** I

Name of the student: Samruddhi Ramswami Bansode. **PRN :** 124B1B019

Department: Computer Engineering **Division :** A

Course Name : Data Structure Lab

Course Code: BCE23PC02

Completion Date :

Assignment No. 4

Problem Statement:

Design a music playlist system using a linked list where:

- Songs can be added to the beginning/end
- Songs can be deleted
- Next and previous songs can be navigated

Source Code :

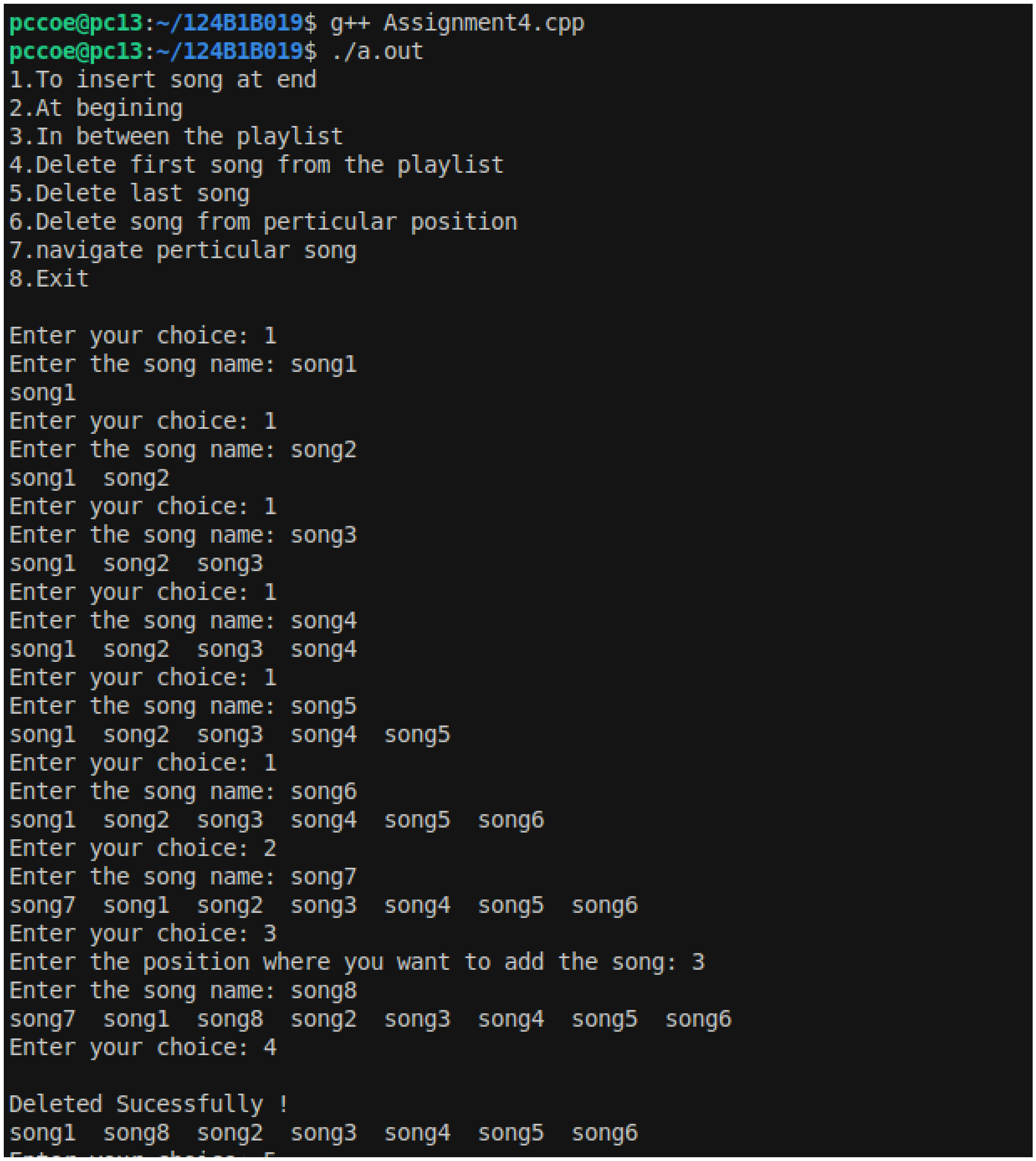
```
#include<iostream>
using namespace std;

class Music{
   public:
   string song;
   Music *next;
   Music(){
      song = " ";
      next = NULL;
   }
   Music(string s){
      song = s;
   }
};

class Music_playlist{
```

```
private:
Music *head;
public:
Music_playlist(){
   head = NULL;
}

//add song at end
void addSong_end(string song){
   Music *new_song = new Music(song);
   if(head == NULL){
      head = new_song;
   } else{
      Music *t = head;
      while(t-> next != NULL){
         t = t-> next;
      }
      t-> next = new_song;
   }
}

//add song at begining
void addSong_beg(string song){
   Music *new_song = new Music(song);
   if(head == NULL){
      head = new_song;
   } else{
      new_song-> next = head;
      head = new_song;
   }
}

//add song in between
void addSong_mid(string song, int pos){
   if (pos <= 1 || head == NULL) {
      addSong_beg(song);
      return;
   }
   Music *new_song = new Music(song);
   Music *t = head;
   for (int i = 1; i < pos - 1 && t->next != NULL; i++) {
      t = t->next;
   }
   new_song->next = t->next;
   t->next = new_song;
}
```

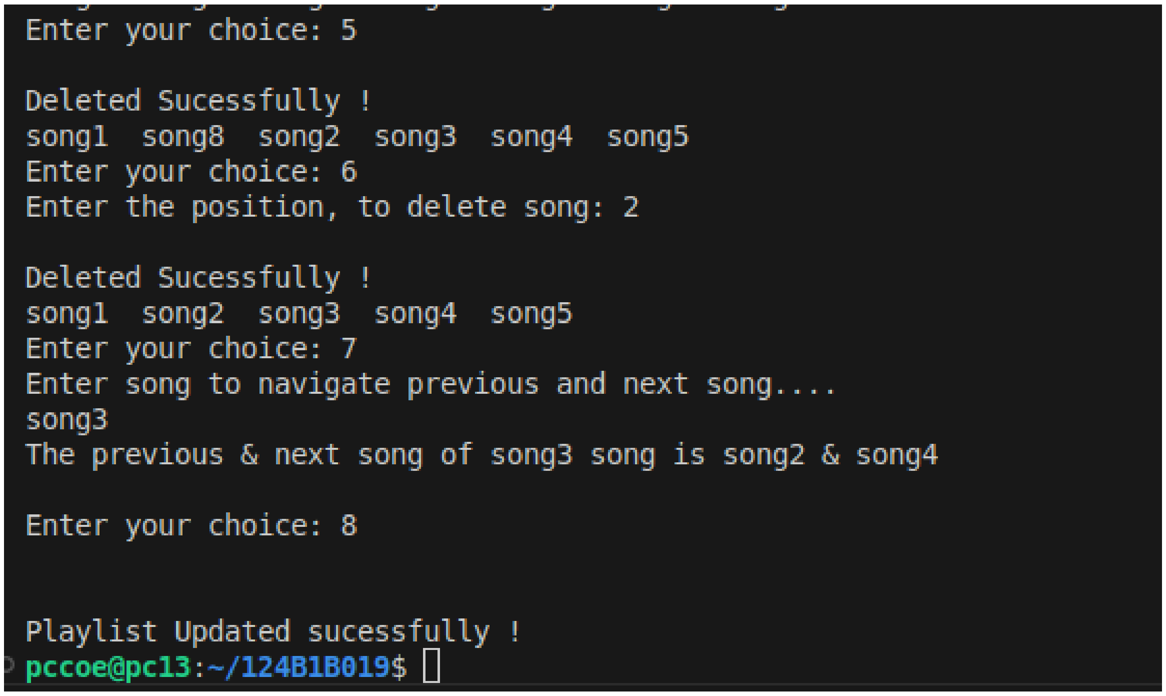
```
// delete first song in the playlist
void Delete_fsong(){
    if(head == NULL){
        cout << "Playlist is empty.";
    }
    else{
        Music* temp = head;
        head = head -> next;
        delete temp;
    }
}

// delete last song in the playlist
void Delete_esong(){
    if(head == NULL){
        cout << "Playlist is empty.";
    }
    else{
        Music* temp = head;
        while(temp -> next -> next != NULL){
            temp = temp -> next;
        }
        Music* prev = temp -> next;
        delete prev;
        temp -> next = NULL;
    }
}

// delete song which is at perticular position
void Delete_pos(int pos){
    if(head == NULL){
        Music* temp = head;
        head = head -> next;
        delete temp;
    }
    else{
        Music* temp = head;
        Music* curr = head;
        for(int i = 1; i < pos - 1 && temp -> next != NULL; i++){
            curr = curr -> next;
            temp = temp -> next;
        }
        temp = temp -> next;
        curr -> next = temp -> next;
        delete temp;
        }
    }

    // to navigate prev and next song of given position
    void nev_song(string key_song){
        Music* temp = head;
        Music* prev = head;
        while(temp -> song != key_song){
            prev = temp;
            temp = temp->next;
        }
          cout << "The previous & next song of " << key_song << " song is " << prev->song << " & " <<
temp -> next -> song;
        cout << endl;
    }

    //display playlist
    void display(){
        Music *t = head;
        if(head == NULL){
            cout<<"Playlist is empty"<< endl;
        }
        while(t != NULL){
            cout << t->song << "  ";
            t = t->next;
        }
    }
};

int main(){
    Music_playlist playlist;
    int pos,choice;
    string song;
     cout << "1.To insert song at end\n2.At begining\n3.In between the playlist\n4.Delete first song from
the playlist\n5.Delete last song\n6.Delete song from perticular position\n7.navigate perticular
song\n8.Exit\n";
    do{
        cout << "\nEnter your choice: ";
        cin >> choice;
        switch(choice){
            case 1:
                cout << "Enter the song name: ";
                cin >> song;
                playlist.addSong_end(song);
                playlist.display();
                break;
            case 2:
                cout << "Enter the song name: ";
                cin >> song;
                playlist.addSong_beg(song);
                playlist.display();
                break;

            case 3:
                cout << "Enter the position where you want to add the song: ";
                cin >> pos;
                cout << "Enter the song name: ";
                cin >> song;
                playlist.addSong_mid(song,pos);
                playlist.display();
                break;

            case 4:
                playlist.Delete_fsong();
                cout << "\nDeleted Sucessfully !\n";
                playlist.display();
                break;

            case 5:
                playlist.Delete_esong();
                cout << "\nDeleted Sucessfully !\n";
                playlist.display();
                break;

            case 6:
                cout << "Enter the position, to delete song: ";
                cin >> pos;
                playlist.Delete_pos(pos);
                cout << "\nDeleted Successfully !\n";
                playlist.display();
                break;

            case 7:
                cout << "Enter song to navigate previous and next song....\n";
                cin >> song;
                playlist.nev_song(song);
                break;

            case 8:
                cout << "\n\nPlaylist Updated successfully !\n";
        }
    }while(choice != 8);
```

```
return 0;
}
```

Screen Shot of Output :



```
pccoe@pc13:~/124B1B019$ g++ Assignment4.cpp
pccoe@pc13:~/124B1B019$ ./a.out
1.To insert song at end
2.At begining
3.In between the playlist
4.Delete first song from the playlist
5.Delete last song
6.Delete song from perticular position
7.navigate perticular song
8.Exit

Enter your choice: 1
Enter the song name: song1
song1
Enter your choice: 1
Enter the song name: song2
song1  song2
Enter your choice: 1
Enter the song name: song3
song1  song2  song3
Enter your choice: 1
Enter the song name: song4
song1  song2  song3  song4
Enter your choice: 1
Enter the song name: song5
song1  song2  song3  song4  song5
Enter your choice: 1
Enter the song name: song6
song1  song2  song3  song4  song5  song6
Enter your choice: 2
Enter the song name: song7
song7  song1  song2  song3  song4  song5  song6
Enter your choice: 3
Enter the position where you want to add the song: 3
Enter the song name: song8
song7  song1  song8  song2  song3  song4  song5  song6
Enter your choice: 4

Deleted Sucessfully !
song1  song8  song2  song3  song4  song5  song6
```



```
Enter your choice: 5

Deleted Sucessfully !
song1  song8  song2  song3  song4  song5
Enter your choice: 6
Enter the position, to delete song: 2

Deleted Sucessfully !
song1  song2  song3  song4  song5
Enter your choice: 7
Enter song to navigate previous and next song....
song3
The previous & next song of song3 song is song2 & song4

Enter your choice: 8


Playlist Updated sucessfully !
pccoe@pc13:~/124B1B019$
```

Conclusion:

This program manages a playlist using a linked list, allowing users to add or remove songs at the beginning, end, or any position via a menu. It efficiently demonstrates core linked list operations and dynamic data management in C++. Overall, it serves as a practical example of using linked lists for playlist management.