UNIVERSITY OF SIALKOT

A CHARTERED UNIVERSITY

Department of Software Engineering

University of Sialkot

Lab manual 2

Submitted by : Hubab Sherazi

Submitted to: Mam Azka

Subject: DSA

Roll no: 059

Semester: BS Software Engineering 3rd semester

Section: grey

BS SOFTWARE ENGINEERING

CODE

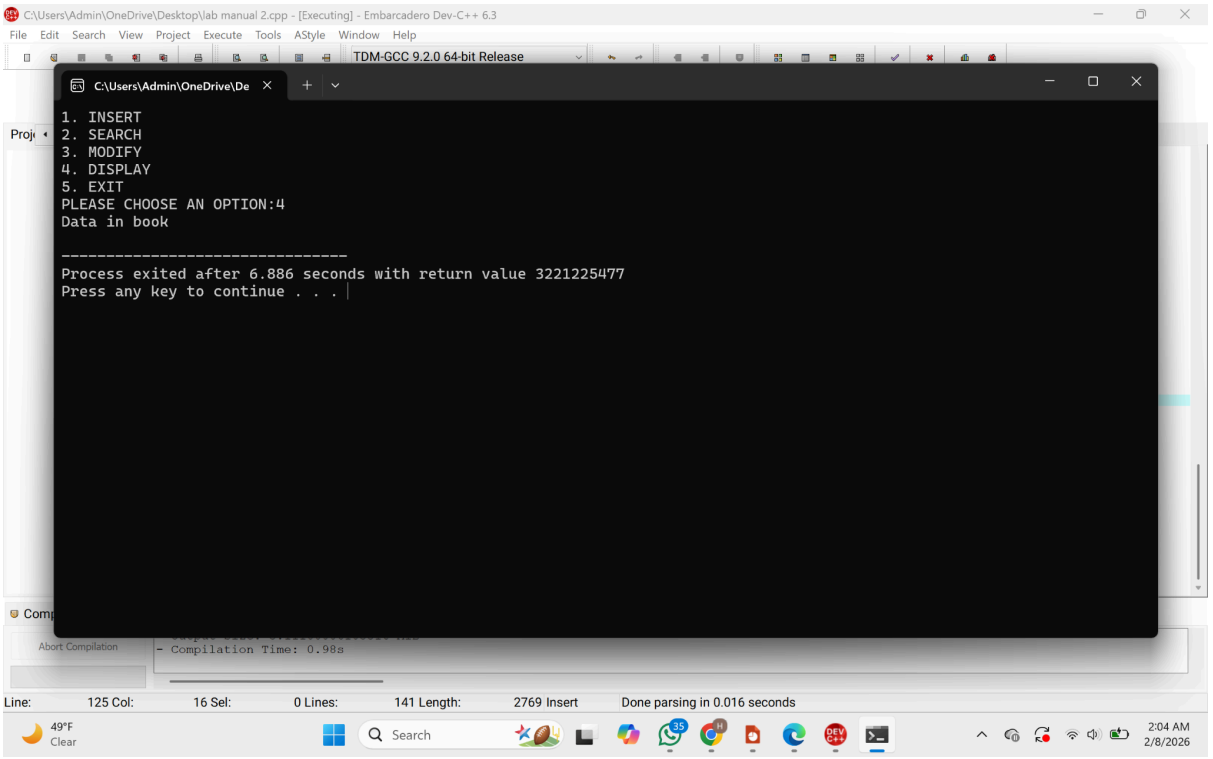
Exercise 2.1 Consider a scenario where a library wants to manage its collection of books. Each book has a unique ISBN, title, author, and publication year. The data is maintained in a doubly linked list. Create the following functions for the book collection: 1. Insert: Insert a new book record into the collection. 2. Search: Search for a book record using ISBN or title. 3. Modify: Update the details of an existing book record. 4. Display: Display all book records and the total number of books in the collection.

```
#include <iostream>
using namespace std;
/* run this program using the console pauser or add your own getch, system("pause")
or input loop */
struct book{
	int isbn;
	int year;
	string title;
	string author;
	book*p;
	book*n;
};
class doubly{
	private:
		book*start,*curr,*temp;
		public:
		doubly(){
			start=NULL;
		}
		void insert(){
			if(start==NULL){
				start=new book;
				cout<<"Enter ISBN:";
				cin>>start->isbn;
				cout<<"Enter year:";
				cin>>start->year;
				cout<<"Enter title:";
				cin>>start->title;
				cout<<"Enter Author:";
				cin>>start->author;
				start->p=NULL;
				start->n=NULL;
			}
			else{
				curr=start;
				while(curr->n!=NULL){
					curr=curr->n;
```

```
                }
                temp=new book;
                cout<<"Enter ISBN:";
                cin>>temp->isbn;
                cout<<"Enter year:";
                cin>>temp->year;
                cout<<"Enter title:";
                cin>>temp->title;
                cout<<"Enter Author:";
                cin>>temp->author;
                temp->n=NULL;
                temp->p=curr;
                curr->n=temp;
        }
}
void search(){
        int x;
        cout<<"Enter ISBN to search:  ";
        cin>>x;

        temp=start;
        while(temp!=NULL){
                if(temp->isbn==x){
                        cout<<"book found"<<endl;
                        cout<<temp->isbn<<endl;
                        cout<<temp->year<<endl;
                        cout<<temp->title<<endl;
                        cout<<temp->author<<endl;
                }
                temp=temp->n;
        }
        cout<<"book not found"<<endl;
}
void modify(){
        int w;
        cout<<"Enter ISBN to modify:  ";
        cin>>w;
        temp=start;
        while(temp!=NULL){
                if(temp->isbn==w){
                cout<<"Enter new title:";
                cin>>temp->title;
                cout<<" Enter new year:";
                cin>>temp->year;
                cout<<"Enter new author";
                cin>>temp->author;
                cout<<"Record update"<<endl;
                }
```

```
                        temp=temp->n;

                    }
                    cout<<"book not found"<<endl;
                }
                void display(){
                    cout<<"Data in book"<<endl;
                    curr=start;
                    while(curr->n!=NULL){
                        cout<<curr->isbn<<endl;
                        cout<<curr->year<<endl;
                        cout<<curr->title<<endl;
                        cout<<curr->author<<endl;
                        curr=curr->n;
                    }
                    cout<<curr->isbn<<endl;
                        cout<<curr->year<<endl;
                        cout<<curr->title<<endl;
                        cout<<curr->author<<endl;
                }
};
int main(int argc, char** argv) {
        doubly d;
        int choice;
        do{
                cout<<"1. INSERT"<<endl;
                cout<<"2. SEARCH"<<endl;
                cout<<"3. MODIFY"<<endl;
                cout<<"4. DISPLAY"<<endl;
                cout<<"5. EXIT"<<endl;
                cout<<"PLEASE CHOOSE AN OPTION:";
                cin>>choice;

        switch(choice){
                case 1:
                d.insert();
                break;
      case 2:
                d.search();
                break;
      case 3:
                d.modify();
                break;
      case 4:
                d.display();
                break;
      case 5:
      cout<<"Exiting.."<<endl;
```

```
            break;
    default:
            cout << "Invalid Choice";
            break;
            }
            }
            while(choice!=5);
      return 0;
```



Code 2

Exercise 2.2 Consider a scenario where a music app wants to manage its playlist of songs. Each song has a unique ID, title, artist name, and duration. The data is maintained in a doubly linked list, allowing for efficient traversal in both forward and backward directions. This is particularly useful for features like navigating through songs in a playlist and enabling users to easily move to the previous or next track. Create the following functions for the music playlist: 1. Insert: Add a new song record to the playlist. 2. Search: Find a song record using the title or artist name. 3. Modify: Update the details of an existing song record. 4. Display: Display all songs in the playlist, the total number of songs and total duration. 5. Play Next: Move to the next song in the playlist. 6. Play Previous: Move to the previous song in the playlist. © Department of Software Engineering 5 Faculty of Computing & IT University of Sialkot Exercise 2.2 Write a function MoveToFront( ) with one argument of data type integer. This function will first search the linked list and compare the Data member of the

node with the given number as argument. If the search finds the number in the list then this function will move that node to the start of the list as shown in the example below. The order of the remaining nodes is to remain unchanged. If no node in the list contains the integer, MoveToFront( ) should leave the list unchanged. Assume that the list contains no duplicate values. The structure definition of the doubly linked list is structListNode{ ListNode *pre; int Data; ListNode *Next; };

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

struct SongNode {
    int id;
    int duration;
    string title;
    string artist;
    SongNode* prev;
    SongNode* next;
};

class Playlist {
private:
    SongNode* head;
    SongNode* current;

public:
    Playlist() {
        head = NULL;
        current = NULL;
    }

    void addSong() {
        SongNode* newSong = new SongNode;

        cout << "Enter Song ID: ";
        cin >> newSong->id;
        cout << "Enter Duration: ";
        cin >> newSong->duration;
        cout << "Enter Title: ";
        cin >> newSong->title;
        cout << "Enter Artist: ";
        cin >> newSong->artist;

        newSong->next = NULL;
        newSong->prev = NULL;

        if (head == NULL) {
```

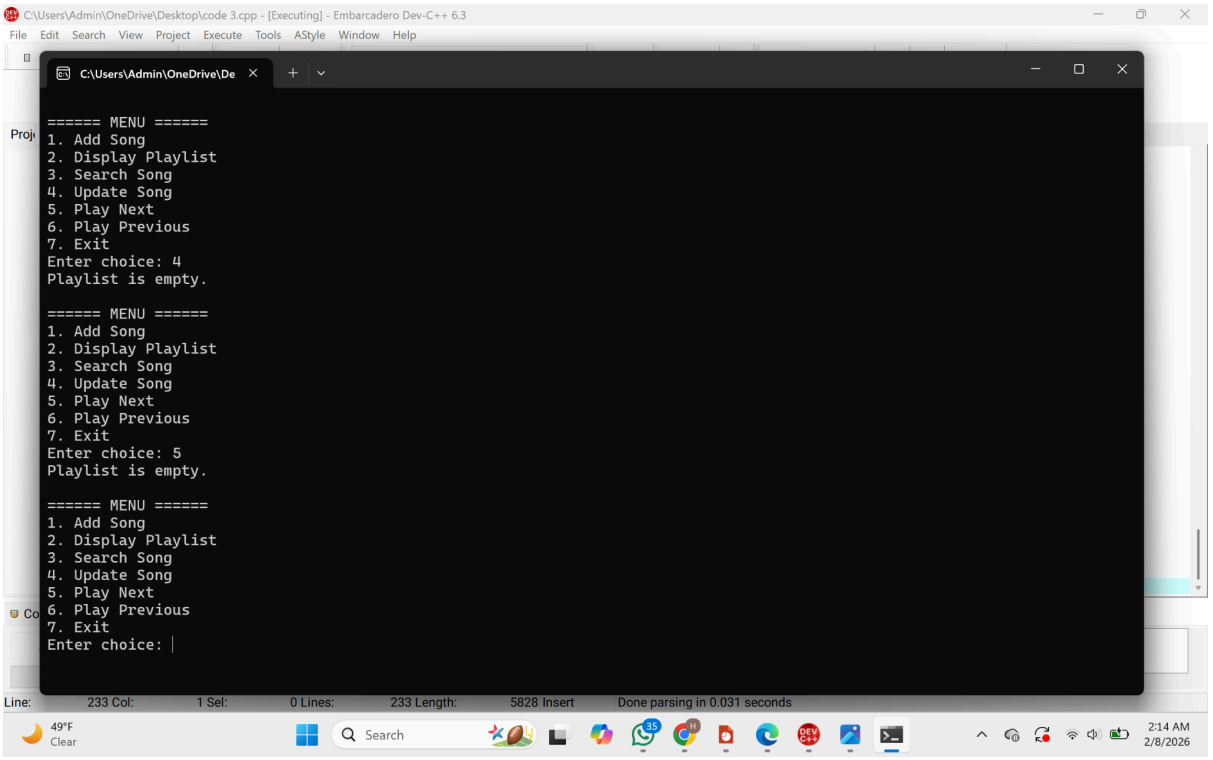
```
        head = newSong;
    } else {
        SongNode* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newSong;
        newSong->prev = temp;
    }

    cout << "Song Added Successfully!\n";
}

void displayAll() {
    if (head == NULL) {
        cout << "Playlist is empty.\n";
        return;
    }

    SongNode* temp = head;
    cout << "\n--- Playlist ---\n";
    while (temp != NULL) {
        cout << "ID: " << temp->id << endl;
        cout << "Duration: " << temp->duration << endl;
        cout << "Title: " << temp->title << endl;
        cout << "Artist: " << temp->artist << endl;
        cout << "-----------------\n";
        temp = temp->next;
    }
}

void searchSong() {
    if (head == NULL) {
        cout << "Playlist is empty.\n";
        return;
    }

    int option;
    cout << "Search By:\n1. Title\n2. Artist\nChoose option: ";
    cin >> option;

    bool found = false;
    SongNode* temp = head;

    if (option == 1) {
        string t;
        cout << "Enter Title: ";
        cin >> t;
```

```
        while (temp != NULL) {
            if (temp->title == t) {
                cout << "Song Found:\n";
                cout << temp->id << " "
                    << temp->duration << " "
                    << temp->title << " "
                    << temp->artist << endl;
                found = true;
            }
            temp = temp->next;
        }
    }
    else if (option == 2) {
        string a;
        cout << "Enter Artist: ";
        cin >> a;

        while (temp != NULL) {
            if (temp->artist == a) {
                cout << "Song Found:\n";
                cout << temp->id << " "
                    << temp->duration << " "
                    << temp->title << " "
                    << temp->artist << endl;
                found = true;
            }
            temp = temp->next;
        }
    }

    if (!found) {
        cout << "Song not found.\n";
    }
}

void updateSong() {
    if (head == NULL) {
        cout << "Playlist is empty.\n";
        return;
    }

    int searchID;
    bool found = false;

    cout << "Enter Song ID to update: ";
    cin >> searchID;
```

```
    SongNode* temp = head;

    while (temp != NULL) {
        if (temp->id == searchID) {
            cout << "Enter New Title: ";
            cin >> temp->title;
            cout << "Enter New Artist: ";
            cin >> temp->artist;
            cout << "Enter New Duration: ";
            cin >> temp->duration;

            cout << "Song Updated Successfully!\n";
            found = true;
            break;
        }
        temp = temp->next;
    }

    if (!found) {
        cout << "Song not found.\n";
    }
}

void playNext() {
    if (head == NULL) {
        cout << "Playlist is empty.\n";
        return;
    }

    if (current == NULL)
        current = head;
    else if (current->next != NULL)
        current = current->next;

    cout << "Now Playing: "
         << current->title << " by "
         << current->artist << endl;
}

void playPrevious() {
    if (current != NULL && current->prev != NULL) {
        current = current->prev;
        cout << "Now Playing: "
             << current->title << " by "
             << current->artist << endl;
    } else {
        cout << "No previous song available.\n";
    }
```

```
    }
};

int main() {
    Playlist pl;
    int choice;

    do {
        cout << "\n====== MENU ======\n";
        cout << "1. Add Song\n";
        cout << "2. Display Playlist\n";
        cout << "3. Search Song\n";
        cout << "4. Update Song\n";
        cout << "5. Play Next\n";
        cout << "6. Play Previous\n";
        cout << "7. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            pl.addSong();
            break;
        case 2:
            pl.displayAll();
            break;
        case 3:
            pl.searchSong();
            break;
        case 4:
            pl.updateSong();
            break;
        case 5:
            pl.playNext();
            break;
        case 6:
            pl.playPrevious();
            break;
        case 7:
            cout << "Exiting Program...\n";
            break;
        default:
            cout << "Invalid Choice!\n";
        }

    } while (choice != 7);

    return 0;
```

}


C:\Users\Admin\OneDrive\Desktop\code 3.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
C:\Users\Admin\OneDrive\De
====== MENU ======
1. Add Song
2. Display Playlist
3. Search Song
4. Update Song
5. Play Next
6. Play Previous
7. Exit
Enter choice: 4
Playlist is empty.
====== MENU ======
1. Add Song
2. Display Playlist
3. Search Song
4. Update Song
5. Play Next
6. Play Previous
7. Exit
Enter choice: 5
Playlist is empty.
====== MENU ======
1. Add Song
2. Display Playlist
3. Search Song
4. Update Song
5. Play Next
6. Play Previous
7. Exit
Enter choice:
Line: 233 Col: 1 Sel: 0 Lines: 233 Length: 5828 Insert Done parsing in 0.031 seconds
49°F Clear
Search
2:14 AM 2/8/2026