Assignment 03

Report

Name:

Student ID:

Lab Class:

Tutor:

Due Date:

Date Submitted:

Assignment Tile : Assignment 03

Problem – 01

1. Program Description

In this program I am creating a students details management system which can be used to insert all the student details into nested struct and handling it as the user requires.

2. Inputs and Outputs

Data to be Stored	Sample data	Type of Data	C++ Type	Input method	In / Out	Variable Names
Student Name	Zen	Text: 2 – 10 characters	String	Cin	Cout	name
Student ID	1101825	Real Numbers	Int	Cin	Cout	Id
Course Name	Computer	Text: 2 – 15 characters	String	Cin	Cout	course_name
Number of Units	4	Real Numbers	Int	Cin	Cout	number_of_ units
Marks	3	Real Number Array	Int	Cin	Cout	Marks[4]
Average	45.7	Real Numbers	Float	Cin	Cout	Avg
Count	3	Real Numbers	Int	Cin	Cout	Count

3. Source Code

```cpp
//this program is created to manage the student information using a multiple structs
//Student Name : 
//Student ID : 
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <ncurses.h>
//#include <conio.h>
using namespace std;

//global varibles
int count = 0;

//declaring the struct
struct person_tag{
string name;
string id;
};

struct course_tag{
string course_name;
int no_of_units;
int marks[4];
float avg;
};

struct student_tag{
struct person_tag student_info;
struct course_tag course_info;
};

//initializing the functions
void read_file(student_tag []);
int menu();
void display_students(student_tag [], int num);
void sort(student_tag [], int num);
void search(student_tag [], int num, string name);
void find_maximum(student_tag [], int num);
void update_file(student_tag []);
int quit();

//sorting functions initializing
void sortByMarks (student_tag [], int num);
void sortByName (student_tag [], int num);
```

```
//main method
int main(){
student_tag student_array[100];
//greeting message
cout << "Welcome to the Student Management System" << endl;

//creating an instance from the fstream class
fstream sample;
//call read_file function
read_file(student_array);
//calling the menu function
int x = menu();

//looking for the choice
switch (x)
{
case 1:{
display_students(student_array, count);
menu();
break;
}
case 2:{
sort(student_array, count);
menu();
break;
}
case 3:{
string name;
cout<<"Insert the name of the studnet : ";
cin>>name;
//calling the search function
search(student_array, count, name);
menu();
break;
}
case 4:{
find_maximum(student_array, count);
menu();
break;
}
case 5:{
update_file(student_array);
menu();
break;
}
case 6:
quit();
menu();
break;
default:
cout<<"Response is not valid"<<endl;
menu();
break;
}


return 0;
}

//read files details into the struct array
void read_file(student_tag s[]){
//creating a temparary struct and count varible
student_tag temp;
int total = 0;
//creating the file instance
fstream sample("Students.txt", ios::in | ios::out | ios::app);
//opening and checking for errors
//sample.open("Students.txt", ios::in, ios::out, ios::app);
if(sample.fail()){
cerr<<"error while opening the file"<<endl;
exit(1);
}
while (sample.eof()) {
sample>>temp.student_info.name>>temp.student_info.id>>temp.course_info.course_na
me>>temp.course_info.no_of_units>>temp.course_info.marks[0]>>temp.course_info.mar
ks[1]>>temp.course_info.marks[2]>>temp.course_info.marks[3];
for (int i = 0; i<4; i++) {
total = total + temp.course_info.marks[i];
}
temp.course_info.avg = total/4;

s[count]=temp;
count++;
}

sample.close();
}

int menu(){
int response=0;

//displaying the options
cout<<"1 --> Display students’ details"<<endl;
cout<<"2 --> Sort the students’ details"<<endl;
cout<<"3 --> Search for a student’s mark"<<endl;
cout<<"4 --> Find the details of student with the largest average"<<endl;
cout<<"5 --> Add new student to the record"<<endl;
cout<<"6 --> Quit program"<<endl;

cin>>response;

//pass the user response
```

```
return response;
}

//display function
void display_students(student_tag s[], int num){
for (int i=0; i<num; i++) {
cout<<"Student Name : "<<s[i].student_info.name<<endl;
cout<<"Student ID : "<<s[i].student_info.id<<endl;
cout<<"Course Name : "<<s[i].course_info.course_name<<endl;
cout<<"Number of Units : "<<s[i].course_info.no_of_units<<endl;
cout<<"Marks : "<<s[i].course_info.marks[0]<<", "<<s[i].course_info.marks[1]<<", 
"<<s[i].course_info.marks[2]<<", "<<s[i].course_info.marks[3]<<endl;
cout<<"Average of Marks : "<<s[i].course_info.avg<<endl;
}

//calling the menu function
int x = menu();

//looking for the choice
switch (x)
{
case 1:{
display_students(s, count);
menu();
break;
}
case 2:{
sort(s, count);
menu();
break;
}
case 3:{
string name;
cout<<"Insert the name of the studnet : ";
cin>>name;
//calling the search function
search(s, count, name);
menu();
break;
}
case 4:{
find_maximum(s, count);
menu();
break;
}
case 5:{
update_file(s);
menu();
break;
}
case 6:
```

```
quit();
menu();
break;
default:
cout<<"Response is not valid"<<endl;
menu();
break;
}
}

//sorting the arrays as the user required
void sort(student_tag s[], int num) {
cout<<"1 -> Sort by Name\n";
cout<<"2 -> Sort by Average Marks\n";
int method;
cin>>method;

if (method == 1){
//sortByName(s, 6num);
clrscr();
int i, j;
for(i=1; i<num; i++){
for(j=1; j<num; j++){
if(strcmp(s[j-1].student_info.name, s[j].student_info.name)>0){
strcpy(t, str[j-1]);
strcpy(str[j-1], str[j]);
strcpy(str[j], t);
}
}
}
cout<<"Names in alphabetical order : \n";
for(i=0; i<5; i++){
cout<<str[i]<<"\n";
}
getch();
}
else {
if (method == 2 ) {
int i,j,temp, pass = 0;
for(i = 0; i<num; i++) {
for(j = i+1; j<num; j++) {
if(s[j].course_info.avg < s[i].course_info.avg) {
temp = s[i].course_info.avg;
s[i].course_info.avg = s[j].course_info.avg;
s[j].course_info.avg = temp;
}
}
pass++;
}

cout <<"Sorted List ...\n";
for(i = 0; i<num; i++) {
cout <<s[i].student_info.name<<"\t"<<s[i].course_info.avg<<endl;
}
}
else {
cout<<"Invalid Input. Try Again\n";
sort(s, num);
}
}

//calling the menu function
int x = menu();

//looking for the choice
switch (x)
{
case 1:{
display_students(s, count);
menu();
break;
}
case 2:{
sort(s, count);
menu();
break;
}
case 3:{
string name;
cout<<"Insert the name of the studnet : ";
cin>>name;
//calling the search function
search(s, count, name);
menu();
break;
}
case 4:{
find_maximum(s, count);
menu();
break;
}
case 5:{
update_file(s);
menu();
break;
}
case 6:
quit();
menu();
break;
default:
cout<<"Response is not valid"<<endl;
```

```
menu();
break;
}
}

//search function
void search(student_tag s[], int num, string name){
//return num;
cout<<"Insert the sorting method : "<<endl;
cout<<"1 - Binary Search\n";
cout<<"2 - Linear Search\n";
int method;
cin>>method;

//binary search
if (method == 1) {
int first = 0;
int last = num-1;
int middle = (first+last)/2;
while (first <= last){
if(s[middle].student_info.name != name){
first = middle + 1;
}
else if(s[middle].student_info.name == name){
cout<<name<<" found in the location "<<middle+1<<"\n";
break;
}
else {
last = middle - 1;
}
middle = (first + last)/2;
}
if(first > last){
cout<<name<<" not found";
}
}
//linear search
else {
if (method == 2) {
for (int j = 0; j < num; j++ ){
if (s[j].student_info.name == name) {
cout<<name<<" found in the location "<<j<<endl;
}
else {
cout << "didn't find it" << endl;
}
}
}
else {
cout<<"Invalid input try again\n";
search(s, count, name);
}
}

//calling the menu function
int x = menu();

//looking for the choice
switch (x)
{
case 1:{
display_students(s, count);
menu();
break;
}
case 2:{
sort(s, count);
menu();
break;
}
case 3:{
string name;
cout<<"Insert the name of the studnet : ";
cin>>name;
//calling the search function
search(s, count, name);
menu();
break;
}
case 4:{
find_maximum(s, count);
menu();
break;
}
case 5:{
update_file(s);
menu();
break;
}
case 6:
quit();
menu();
break;
default:
cout<<"Response is not valid"<<endl;
menu();
break;
}
}

//calculating the maximum average value
```

```
void find_maximum(student_tag s[], int num){
int i,j,temp, pass = 0;
for(i = 0; i<num; i++) {
for(j = i+1; j<num; j++) {
if(s[j].course_info.avg < s[i].course_info.avg) {
temp = s[i].course_info.avg;
s[i].course_info.avg = s[j].course_info.avg;
s[j].course_info.avg = temp;
}
}
pass++;
}

cout<<"The highest scoring student's details are...\n";
cout<<s[num].student_info.name<<endl;
cout<<s[num].student_info.id<<endl;
cout<<s[num].course_info.avg<<endl;

//calling the menu function
int x = menu();

//looking for the choice
switch (x)
{
case 1:{
display_students(s, count);
menu();
break;
}
case 2:{
sort(s, count);
menu();
break;
}
case 3:{
string name;
cout<<"Insert the name of the studnet : ";
cin>>name;
//calling the search function
search(s, count, name);
menu();
break;
}
case 4:{
find_maximum(s, count);
menu();
break;
}
case 5:{
update_file(s);
menu();
```

```
break;
}
case 6:
quit();
menu();
break;
default:
cout<<"Response is not valid"<<endl;
menu();
break;
}

}

//update the text file
void update_file(student_tag s[]){
//creating a file instance
fstream sample("Students.txt", ios::in | ios::out | ios::app);
if (!sample.is_open()){
cout<<"Error while opening the file\n";
}
else {
cout<<"Insert the number of tudents you are going to add : ";
int num;
cin>>num;

//creating a temparary struct
student_tag temp;

for (int i=count; i<(count+num); i++){
cout<<"Insert the name of the student : ";
cin>>temp.student_info.name;

cout<<"Insert the Student ID : ";
cin>>temp.student_info.id;

cout<<"Insert the course name : ";
cin>>temp.course_info.course_name;

cout<<"Insert the number of units : ";
cin>>temp.course_info.no_of_units;

cout<<"Insert the marks : ";
for (int j=0; j<4; j++) {
cin>>temp.course_info.marks[j];
}

sample<<temp.student_info.name<<"\t"<<temp.student_info.id<<"\
t"<<temp.course_info.course_name<<"\t"<<temp.course_info.no_of_units<<"\
t"<<temp.course_info.marks[0]<<"\t"<<temp.course_info.marks[1]<<"\
t"<<temp.course_info.marks[2]<<"\t"<<temp.course_info.marks[3];
```

```
}
}
sample.close();

//calling the read file function
read_file(s);

//calling the menu function
int x = menu();

//looking for the choice
switch (x)
{
case 1:{
display_students(s, count);
menu();
break;
}
case 2:{
sort(s, count);
menu();
break;
}
case 3:{
string name;
cout<<"Insert the name of the studnet : ";
cin>>name;
//calling the search function
search(s, count, name);
menu();
break;
}
case 4:{
find_maximum(s, count);
menu();
break;
}
case 5:{
update_file(s);
menu();
break;
}
case 6:
quit();
menu();
break;
default:
cout<<"Response is not valid"<<endl;
menu();
break;
}
```

```
}

//quit the program
int quit(){
cout<<"Thank you for being with us\n\t\t\tGOOD BYE!\n";
return 0;
}
```

4. Screenshots showing the working program

Problem – 02

1. Program Description

This program implements the same student management system which is going to implemented using the linked list. This way it is easy to map the student information accordingly

2. Inputs and Outputs

Data to be Stored	**Sample data**	**Type of Data**	**C++ Type**	**Input method**	**In / Out**	**Variable Names**
Student Name	Zen	Text: 2 – 10 characters	String	Cin	Cout	name
Student ID	1101825	Real Numbers	Int	Cin	Cout	Id
Course Name	Computer	Text: 2 – 15 characters	String	Cin	Cout	course_name
Number of Units	4	Real Numbers	Int	Cin	Cout	number_of_ units
Marks	3	Real Number Array	Int	Cin	Cout	Marks[4]
Average	45.7	Real Numbers	Float	Cin	Cout	Avg
Count	3	Real Numbers	Int	Cin	Cout	Count
Next pointer	&#*^99317	Pointer	*	Cin	Cout	*next

3. Source Code

```
//this program is created to manage the student information using a linked list
//Student Name : 
//student ID : 
#include <iostream>
#include <fstream>
using namespace std;

//global varibles for node count
int count = 0 ;

//initializing the structs and references
struct person_tag{
string name;
string id;
};
struct course_tag{
string course_name;
int no_of_units;
int marks[4];
float avg;
};
struct student_tag{
struct person_tag student_info;
struct course_tag course_info;
student_tag *next;
};

//initialzing the fuctions
void read_file(student_tag *&head, student_tag *&last);
bool isEmpty(student_tag *&head);
int menu();
void showList(student_tag *&current);
void linear_Search(student_tag *&current, string name);
void find_maximum(student_tag *&current, int num);
int quit();


//main method
int main(){
//creating the nodes
student_tag *head;
student_tag *last;

//calling readfile function
read_file(head, last);

//calling the menu function
```

```
int x = menu();

//determinig the response
switch (x) {
case 1:{
showList(head);
break;
}
case 2:{
cout<<"Insert the name you are looking for : ";
string name;
cin>>name;
linear_Search(head, name);
break;
}
case 3:{
find_maximum(head, count);
break;
}
case 4:{
quit();
break;
}
default:
cout<<"Invalid Input. Please try again\n\n";
break;
}
return 0;
}

//read files details into the struct array
void read_file(student_tag *&head, student_tag *&last){
//creating a temparary struct and count varible
student_tag temp;
int total = 0;

//opening and checking for errors
fstream sample("Students.txt", ios::in | ios::out | ios::app);
if(sample.fail()){
cerr<<"error while opening the file"<<endl;
exit(1);
}
while (sample.eof()) {
sample>>temp.student_info.name>>temp.student_info.id>>temp.course_info.course_na
me>>temp.course_info.no_of_units>>temp.course_info.marks[0]>>temp.course_info.mar
ks[1]>>temp.course_info.marks[2]>>temp.course_info.marks[3];
for (int i = 0; i<4; i++) {
total = total + temp.course_info.marks[i];
}
temp.course_info.avg = total/4;
if (isEmpty(head)){
//insertFirstElement(head, last, temp);
student_tag *temp1 = new student_tag;
temp1 = &temp;
temp1->next = NULL;
head = temp1;
last = temp1;
}
else {
student_tag *temp1 = new student_tag;
for (int k=1; k<10; k++) {
temp1 = &temp;
temp1->next = NULL;
last->next = temp1;
last = temp1;
}
}

count++;
}

sample.close();
}

//empty function
bool isEmpty(student_tag *&head){
if (head == NULL) {
return true;
}
else {
return false;
}
}

//menu function for the responses
int menu(){
int response=0;

//displaying the options
cout<<"1 --> Display students' details"<<endl;
cout<<"2 --> Search for a student's mark"<<endl;
cout<<"3 --> Find the details of student with the largest average"<<endl;
cout<<"4 --> Quit program"<<endl;

cin>>response;

//pass the user response
return response;
}

//quit the program
```

```
int quit(){
cout<<"Thank you for being with us\n\t\t\tGOOD BYE!\n";
return 0;
}

//show the the details
void showList(student_tag *&current){
if (isEmpty(current)){
cout<<"The List is Empty"<<endl;
}
else {
cout<<"The list Contains\n";
while (current != NULL) {
cout<<current->student_info.name<<"\t";
cout<<current->student_info.id<<"\t";
cout<<current->course_info.course_name<<"\t";
cout<<current->course_info.no_of_units<<"\t";
cout<<current->course_info.marks[0]<<"\t";
cout<<current->course_info.marks[1]<<"\t";
cout<<current->course_info.marks[2]<<"\t";
cout<<current->course_info.marks[3]<<"\t";
cout<<current->course_info.avg<<"\t";

current = current->next;
}
}

//calling the menu function
int x = menu();

//determinig the response
switch (x) {
case 1:{
showList(current);
break;
}
case 2:{
cout<<"Insert the name you are looking for : ";
string name;
cin>>name;
linear_Search(current, name);
break;
}
case 3:{
find_maximum(current, count);
break;
}
case 4:{
quit();
break;
}
default:
cout<<"Invalid Input. Please try again\n\n";
break;
}
}

//void search the students marks
void linear_Search(student_tag *&current, string name){
while (!isEmpty(current)){
if (current->student_info.name == name){
cout<<"The marks are ..\n";
for (int i = 0; i<4; i++){
cout<<current->course_info.marks[i]<<"\t";
}
}
else {
cout << "Cannot find the name" << endl;
}
}

//calling the menu function
int x = menu();

//determinig the response
switch (x) {
case 1:{
showList(current);
break;
}
case 2:{
cout<<"Insert the name you are looking for : ";
string name;
cin>>name;
linear_Search(current, name);
break;
}
case 3:{
find_maximum(current, count);
break;
}
case 4:{
quit();
break;
}
default:
cout<<"Invalid Input. Please try again\n\n";
break;
}
}

//searching for the the maximum average value
```

```
void find_maximum(student_tag *&current, int num){
int i,j,temp, pass = 0;
//creating a temparary node for the check
student_tag *nextVal;
for(i = 0; i<num; i++) {
current->next = nextVal;
if(current->course_info.avg < nextVal->course_info.avg) {
temp = current->course_info.avg;
current->course_info.avg = nextVal->course_info.avg;
nextVal->course_info.avg = temp;
}
pass++;
}

cout<<"The highest scoring student's details are...\n";
cout<<nextVal->student_info.name<<endl;
cout<<nextVal->student_info.id<<endl;
cout<<nextVal->course_info.avg<<endl;

//calling the menu function
int x = menu();

//determinig the response
switch (x) {
case 1:{
showList(current);
break;
}
case 2:{
cout<<"Insert the name you are looking for : ";
string name;
cin>>name;
linear_Search(current, name);
break;
}
case 3:{
find_maximum(current, count);
break;
}
case 4:{
quit();
break;
}
default:
cout<<"Invalid Input. Please try again\n\n";
break;
}

}
```

4. Screenshots showing the working program

Task 8-5

1. Program Description

This program implements arranging three integers according to the ascending order. Linear comparison method has been used in this scenario and user can input any three numbers less than 50.

2. Source Code

```
//this program implements the sorting of three integers which is inserted by the user 
according to the accending order
#include <iostream>
using namespace std;

//initialize functions
void reorder(int *a, int *b, int *c);
int main(){
//initialize variables
int aa,bb,cc;

//get the inputs from the user
cout<<"Insert three integers"<<endl;
cin>>aa;
cin>>bb;
cin>>cc;

//assigning the pointers
int * a = & aa;
int * b = & bb;
int * c = & cc;

//call the re-order function
reorder(a, b, c);
return 0;
}

void reorder(int *a, int *b, int *c){
//compare the numbers according to the order
if(*a>*b){
if(*b>*c){
cout<<"The Order is "<<*c<<","<<*b<<","<<*a<<endl;
}
else{
if(*c>*a){
cout<<"The Order is "<<*b<<","<<*a<<","<<*c<<endl;
}
else{
cout<<"The Order is "<<*b<<","<<*c<<","<<*a<<endl;
}
}
}
else{
if(*b>*c){
if(*a>*c){
cout<<"The Order is "<<*c<<","<<*a<<","<<*b<<endl;
}
else{
cout<<"The Order is "<<*a<<","<<*b<<","<<*c<<endl;
}
}
else{
cout<<"The Order is "<<*a<<","<<*c<<","<<*b<<endl;
}
}
}
```

3. Screenshots showing the working program

Task 8-6

1. Program Description

This program implements a program that calculates and displays voltage amounts according to the resistance and current. All the values are inserted in the arrays and two different functions are used to do the operations.

2. Source Code

```
//This program implements a program that calculates and displays voltage amounts 
according to the resistance and current.
#include <iostream>
using namespace std;
//initializing the functions
void calcVolts(int current[10], int voltage[10], int resistance[10]);
void dispVolts(int arr1[10], int arr2[10], int arr3[10]);

int main(){
//initalizing the arrays
int current[10] = {2,4,6,8,2,3,5,12,3,4};
int voltage[10];
int resistance[10] = {12,34,54,23,43,76,23,11,24,54};

//calling the calcVolts function and passing the arrays
calcVolts(current, voltage, resistance);
//display voltage values
dispVolts(current, resistance, voltage);
return 0;
}

void calcVolts(int current[10], int voltage[10], int resistance[10]){
for (int i=0; i<10; i++){
voltage[i] = current[i] * resistance[i];
}
}

void dispVolts(int arr1[10], int arr2[10], int arr3[10]){
cout<<"Current \t Resistance \t Voltage"<<endl;
for (int j=0; j<10; j++){
cout<<arr1[j]<<" \t"<<arr2[j]<<" \t"<<arr3[j]<<endl;
}
}
```

3. Screenshots showing the working program

Task 9-2

1. Program Description

This program implements a employee management system which is used to read and store data from the user and store them in a nested struct according to the relevance. Array of struct has been used in this program and user can insert 5 employees to the system.

2. Source Code

```
//This program implements a employee management system
#include <iostream>
#include <iostream>
using namespace std;

//initializing the functions
struct Emp get_data(Emp [], int count);
void print_data(Emp[], int count);
double get_average(Emp [], int count, string companyName);
double get_salary(Emp[], string name);

//initializing the structs
struct company_detail{
string company_id;
string company_name;
};
```

```
struct Emp{
string emp_name;
string emp_id;
double salary;
company_detail cmp_detail;
};

int main(){

//declaring the arrays
Emp employee[5];

int count = 5;
//calling the functions
get_data(employee, 5);
print_data(employee, 5);

//insert the company name
cout<<"Insert the company name you need : ";
string company;
cin>>company;

//call the average function
get_average(employee, count, company);

//insert the the employee name for the salary calculations
cout<<"Insert the name of the employee :";
string name;
cin>>name;

//call the salary function
get_salary(employee, name);
return 0;
}

//read data for the array of structure
Emp get_data(Emp w[5], int count){

//insert details to the array
while (count!=0){
cout<<"Insert the name of the employee : ";
cin>>w[count].emp_name;

cout<<"Insert the Employee ID : ";
cin>>w[count].emp_id;

cout<<"Insert the employee salary : ";
cin>>w[count].salary;

cout<<"Insert Company Name : ";
cin>>w[count].cmp_detail.company_name;
```

```cpp
cout<<"Insert company ID : ";
cin>>w[count].cmp_detail.company_id;

cout<<endl<<endl;

count--;
}


return w[count];
}

//print data form the struct
void print_data(Emp w[5], int count){
while (count!=0) {
cout<<"Employee Name : "<<w[count].emp_name<<endl;
cout<<"Employee ID : "<<w[count].emp_id<<endl;
cout<<"Employee Salary : "<<w[count].salary<<endl;
cout<<"Company Name : "<<w[count].cmp_detail.company_name<<endl;
cout<<"Company ID : "<<w[count].cmp_detail.company_id<<endl;

cout<<endl<<endl;
count--;
}
}

//calculate the average salary for the each company
double get_average(Emp w[5], int count, string companyName){
double avg;
int num = 0;
double salary = 0.00;
//counting the company apearances
while (count!=0) {
if(w[count].cmp_detail.company_name==companyName){
num++;
salary = salary + w[count].salary;
}
else{

}
}
avg = salary / (double)num;
cout<<"The average salary of "<<companyName<<" company is "<<avg<<endl;

return avg;
}

//getting the salary of an employee
double get_salary(Emp w[5], string name){
double salary;
```

```
int count = 5;

//finding the salary value of the employee
while (count!=0){
if (w[count].emp_name == name){
cout<<"Salary of the "<<name<<" employee is "<<w[count].salary;
salary = w[count].salary;
}
else{

}
}

return salary;
}
```

3. Screenshots showing the working program

Task 9-3

1. Program Description

This program implements a media playlist. Users can insert data into the system and they also can access different software from the local computer.

2. Source Code

```
//This program implements a media playlist
#include <iostream>
#include <string>
#include <windows.h>
using namespace std;

//initializin the enum for genre
enum genre{
pop,
Jazz,
Classic
};

//initializing the struct
struct album {
string album_name;
genre kind;
int trach_number;
string tracks[5];
string tracklocation;
};
```

```
//initializing the functions
void add_album(album);
int initiateFunction();
void print_all_album(album);
void select_track_to_play(album);
int exit();

//main method
int main(){
//struct instance
album w;

//initializing the program
cout<<"Enter the Option"<<endl;
cout<<"\t 1 to add an album"<<endl;
cout<<"\t 2 to print an album"<<endl;
cout<<"\t 3 to play an album"<<endl;
cout<<"\t 4 Exit"<<endl;

int x;
cin>>x;

//determinging the functions and activities
if(x==1){
add_album(w);
}
else {
if (x==2){
print_all_album(w);
}
else {
if (x==3) {
select_track_to_play(w);
}
else {
if (x==4){
exit();
}
else {
cout<<"The inserted respose is not calid. Please run the program again"<<endl;
}
}
}
}

return 0;
}

//initializing function
```

```
int initiateFunction(){
int n=0;

cout<<"Enter the Option"<<endl;
cout<<"\t 1 to add an album"<<endl;
cout<<"\t 2 to print an album"<<endl;
cout<<"\t 3 to play an album"<<endl;
cout<<"\t 4 Exit"<<endl;

cin>>n;

return n;
}


//add album function
void add_album(album w){
int x,y;

cout<<"Enter Album Name : ";
cin>>w.album_name;
cout<<"Enter genre\n \t0-->pop \t1-->Jazz \t2-->Classic"<<endl;
cin>>x;

w.kind = static_cast<genre>(x);

cout<<"enter the number of tracks in the album : ";
cin>>y;

w.trach_number=y;

cout<<"Enter the track names"<<endl;
while (y!=0) {
cin>>w.tracks[y];
y--;
}

cout<<"Enter the file location for the tracks : ";
cin>>w.tracklocation;
cout<<endl<<endl;

int val = initiateFunction();

//determinging the functions and activities
if(val==1){
add_album(w);
}
else {
if (val==2){
print_all_album(w);
}
else {
if (val==3) {
select_track_to_play(w);
}
else {
if (val==4){
exit();
}
else {
cout<<"The inserted respose is not calid. Please run the program again"<<endl;
}
}
}
}
}

void print_all_album(album w){
//displaying the values
cout<<"The Album Name is : "<<w.album_name<<endl;
cout<<"The Genre of the album : "<<w.kind<<endl;
cout<<"No of tracks : "<<w.trach_number;
cout<<"The tracks are : "<<endl;
for (int i=0; i<w.trach_number; i++) {
cout<<w.tracks[i]<<endl;;
}
cout<<"Tracks are located at "<<w.tracklocation<<endl;

int val = initiateFunction();

//determinging the functions and activities
if(val==1){
add_album(w);
}
else {
if (val==2){
print_all_album(w);
}
else {
if (val==3) {
select_track_to_play(w);
}
else {
if (val==4){
exit();
}
else {
cout<<"The inserted respose is not calid. Please run the program again"<<endl;
}
}
}
}
```

```
}

void select_track_to_play(album w) {
//getting the trak name
cout<<"Select a track to play :";
string name;
cin>>name;

//play the sound track
bool played = PlaySound(name, NULL, SND_SYNC);

int val = initiateFunction();

//determinging the functions and activities
if(val==1){
add_album(w);
}
else {
if (val==2){
print_all_album(w);
}
else {
if (val==3) {
select_track_to_play(w);
}
else {
if (val==4){
exit();
}
else {
cout<<"The inserted respose is not calid. Please run the program again"<<endl;
}
}
}
}

}

int exit() {
return 0;
}
```

3. Screenshots showing the working program

Task 10-2

1. Program Description

This program implements a simple linked list that is going to contain five letters from my last name and display them accordingly in the terminal.

2. Source Code

```
//this program implents a linked list that contains 5 charactors.
#include <iostream>
using namespace std;

//initializing the struct
struct studentname {
char letter;
studentname *next;
};

//defining the functions
bool isEmpty(studentname *head);
void insertFirstElement(studentname *&head, studentname *&last, char letter);
void insert(studentname *&head, studentname *&last, char letter);
void showList(studentname *current);

int main() {
studentname *head = NULL;
studentname *last = NULL;

//calling functions ffor the insertions
insertFirstElement(head, last, 'D');
insert(head, last, 'h');
insert(head, last, 'a');
insert(head, last, 'r');
insert(head, last, 'm');

//diplaying the linked list
showList(head);

return 0;
}

//empty function
bool isEmpty(studentname *head){
if (head == NULL) {
return true;
}
else {
return false;
}
}
```

```
void insertFirstElement(studentname *&head, studentname *&last, char letter) {
studentname *temp = new studentname;
temp->letter = letter;
temp->next = NULL;
head = temp;
last = temp;
}

void insert(studentname *&head, studentname *&last, char letter) {
if (isEmpty(head)){
insertFirstElement(head, last, letter);
}
else {
studentname *temp = new studentname;
temp->letter = letter;
temp->next = NULL;
last->next = temp;
last = temp;
}
}

void showList(studentname *current){
if (isEmpty(current)){
cout<<"The List is Empty"<<endl;
}
else {
cout<<"The list Contains\n";
while (current != NULL) {
cout<<current->letter<<endl;
current = current->next;
}
}
}
```

3. Screenshots showing the working program

Task 10-3

1. Program Description

This program contains a simple linkned list that contains 10 integers hat user inserted according to the ascending order.

2. Source Code

```
//this program implements a simple linked list
#include <iostream>
using namespace std;

//initializing the self reference struct
struct node {
int numbers;
struct node *next;
};

//initializing functions
void get_numbers(int arr[], node *&head, node *&last);
void sort(int arr[], node *&head, node *&last);
bool isEmpty(node *head);
void insertFirstElement(node *&head, node *&last, int number);
void insert(node *&head, node *&last, int arr[]);
void showList(node *current);

//main method
int main () {
//creating the instances
node *head = NULL;
node *last = NULL;
int arr[10];

get_numbers(arr, head, last);
return 0;
}
//get the inputs from the users
void get_numbers(int arr[10], node *&head, node *&last){
cout<<"Insert 10 integers you like"<<endl;
for (int i=0; i<10; i++) {
cin>>arr[i];
}

//sort the numbers
sort(arr, head, last);
}

//sort the numbers
void sort(int arr[10], node *&head, node *&last){
int temp=0, pass=0;
for(int i = 0; i<10; i++) {
for(int j = i+1; j<10; j++){
if(arr[j] < arr[i]) {
temp = arr[i];
arr[i] = arr[j];
arr[j] = temp;
}
}
pass++;
}

//insert the elements
insert(head, last, arr);

//display numbers
showList(head);

}

//empty function
bool isEmpty(node *head){
if (head == NULL) {
return true;
}
else {
return false;
}
}

//insert the first element
void insertFirstElement(node *&head, node *&last, int number){
node *temp = new node;
temp->numbers = number;
temp->next = NULL;
head = temp;
last = temp;
}

//insert into the nodes
void insert(node *&head, node *&last, int arr[10]){
if (isEmpty(head)){
insertFirstElement(head, last, arr[0]);
}
else {
node *temp = new node;
for (int k=1; k<10; k++) {
temp->numbers = arr[k];
temp->next = NULL;
last->next = temp;
last = temp;
}
}
}

//show the list
void showList(node *current){
```

```
if (isEmpty(current)){
cout<<"The List is Empty"<<endl;
}
else {
cout<<"The list Contains\n";
while (current != NULL) {
cout<<current->numbers<<endl;
current = current->next;
}
}
}
```

3. Screenshots showing the working program