Name: Daanish Ahmad Mufti Reg: 2023171

Faculty: FCSE (CS) Section:H

# Lab Report

# Task 1: Dynamic Kitchen Management System

#include <iostream>

#include <cstdlib>

using namespace::std;

struct place {

    int fries = 0;

    int pizza = 0;

    int burger = 0;

    int pasta = 0;

    int coke = 0;

    int water = 0;

    bool status;

    double total;

    int order;

};

void bill(place\* temp, int ordernum)

{

    if(temp[ordernum].fries > 0)

    {

        cout << "< Fries > " << endl;

        cout << "Amount: " << temp[ordernum].fries << endl << endl;

    }

    if(temp[ordernum].burger > 0)

    {

        cout << "< Burger > " << endl;

        cout << "Amount: " << temp[ordernum].burger << endl << endl;

    }

    if(temp[ordernum].pizza > 0)

    {

        cout << "< Pizza > " << endl;

        cout << "Amount: " << temp[ordernum].pizza << endl << endl;

    }

    if(temp[ordernum].pasta > 0)

    {

        cout << "< Pasta > " << endl;

        cout << "Amount: " << temp[ordernum].pasta << endl << endl;

    }

    if(temp[ordernum].coke > 0)

    {

        cout << "< Coke > " << endl;

        cout << "Amount: " << temp[ordernum].coke << endl << endl;

    }

    if(temp[ordernum].water > 0)

    {

        cout << "< water > " << endl;

        cout << "Amount: " << temp[ordernum].water << endl << endl;

    }

    cout << "The total bill is: " << temp[ordernum].total << endl;

}

void pending(place \*temp,int size)

{

    for (int i= 0 ; i < size ; i++)

    {

            if(temp[i].status == true)

            {

                cout << "Order " << i+1 << " Completed" << endl;

            }

            else

            {

                cout << "Order " << i+1 << " Pending" << endl;

        }

    }

}

void complete(place \*temp,int ordernum,int size)

{

    for(int i = 0 ; i < size ; i++)

    {

        if(ordernum == temp[i].order)

        {

            temp[i].status = true;

        }

    }

}

void showordernum(place \*temp, int ordernum)

{

    cout << "----------Order-Number-"<< ordernum + 1 << "----------" << endl;

    if(temp[ordernum].fries > 0)

    {

        cout << "< Fries > " << endl;

        cout << "Amount: " << temp[ordernum].fries << endl << endl;

    }

    if(temp[ordernum].burger > 0)

    {

        cout << "< Burger > " << endl;

        cout << "Amount: " << temp[ordernum].burger << endl << endl;

    }

    if(temp[ordernum].pizza > 0)

    {

        cout << "< Pizza > " << endl;

        cout << "Amount: " << temp[ordernum].pizza << endl << endl;

    }

    if(temp[ordernum].pasta > 0)

    {

        cout << "< Pasta > " << endl;

        cout << "Amount: " << temp[ordernum].pasta << endl << endl;

    }

    if(temp[ordernum].coke > 0)

    {

        cout << "< Coke > " << endl;

        cout << "Amount: " << temp[ordernum].coke << endl << endl;

    }

    if(temp[ordernum].water > 0)

    {

        cout << "< water > " << endl;

        cout << "Amount: " << temp[ordernum].water << endl << endl;

    }

    if(temp[ordernum].status == true)

    {

        cout << "Status: <Completed>" << endl << endl;

    }

    else

    {

        cout << "status: <Pending>" << endl << endl;

    }

}

void addorder(place \*&temp, int &size, int &od) {

    place \*tempo = new place[size+1];

    for(int i = 0 ; i < size ; i++) {

        tempo[i] = temp[i];

    }

    temp = tempo;

    od++;

    temp[size].order = od;

    temp[size].status = false;

    int input, q;

    while(q != 1)

    {

    cout << "Enter the type of order (1-6): ";

    cin >> input;

    switch (input) {

        case 1: {

            cout << "Enter the amount of fries you want: ";

            cin >> temp[size].fries;

            temp[size].total += 20 \* temp[size].fries;

            break;

        }

        case 2: {

            cout << "Enter the amount of burgers you want: ";

            cin >> temp[size].burger;

            temp[size].total += 50 \* temp[size].burger;

            break;

        }

        case 3: {

            cout << "Enter the amount of pizza you want: ";

            cin >> temp[size].pizza;

            temp[size].total += 80 \* temp[size].pizza;

            break;

        }

        case 4: {

            cout << "Enter the amount of pasta you want: ";

            cin >> temp[size].pasta;

            temp[size].total += 60 \* temp[size].pasta;

            break;

        }

        case 5: {

            cout << "Enter the amount of coke you want: ";

            cin >> temp[size].coke;

            temp[size].total += 15 \* temp[size].coke;

            break;

        }

        case 6: {

            cout << "Enter the amount of water you want: ";

            cin >> temp[size].water;

            temp[size].total += 5 \* temp[size].water;

            break;

        }

    }

    cout << endl << "Enter 1 to stop ordering or any other number to continue: ";

    cin >> q;

 }

    size++;

}

void menudisp() {

    cout << "Enter 1 for fries <20$> per: " << endl;

    cout << "Enter 2 for burger <50$> per: " << endl;

    cout << "Enter 3 for pizza: <80$> per" << endl;

    cout << "Enter 4 for pasta: <60$> per" << endl;

    cout << "Enter 5 for coke: <15$> per" << endl;

    cout << "Enter 6 for water: <5$> per" << endl;

}

int main()

{

    int size = 0;

    int od = 0;

    int q;

    place \*temp = new place[size];

    while(q != 1)

    {

        int in;

        cout << "Enter 1 for add new order: " << endl;

        cout << "Enter 2 to print bill: " << endl;

        cout << "Enter 3 to complete order: " << endl;

        cout << "Enter 4 to check for pending orders: " << endl;

        cout << "Enter 5 to display the order by number: " << endl;

        cin >> in;

        system("CLS");

        switch (in) {

            case 1: {

                menudisp();

                addorder(temp, size, od);

                break;

            }

            case 2: {

                int num;

                cout << "Enter the order num to print bill: ";

                cin >> num;

                void (\*ptr)(place \*, int) = &bill;

                bill(temp, num - 1);

                break;

            }

            case 3: {

                cout << "Enter order number to mark complete: ";

                int num;

                cin >> num;

                void (\*ptr)(place \*, int, int) = &complete;

                complete(temp, num, size);

                break;

            }

            case 4: {

                void (\*ptr)(place \*, int) = &pending;

                pending(temp, size);

                break;

            }

            case 5: {

                int num;

                cout << "Enter the order num to show the order: ";

                cin >> num;

                void (\*ptr)(place \*, int) = &showordernum;

                showordernum(temp, num - 1);

                break;

                break;

            }

        }

        cout << "Enter 1 to quit or any other number to continue: ";

        cin >> q;

        system("CLS");

    }

    return 0;

}

Output:

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

See more functions yourself.

Uses a structure called place with variables ,string food, int amount, bool status, double total, int order. Has a void function called bill that shows the name of food item, the amount , the total price, its status (pending or completed). Has another void function called pending that performs a check on all orders and displaying theirs status. Void function complete is used by passing the order number by user to mark the order as completed. Void function addorder uses creates a new array of structure place objects with a size of size+1. The memory for this new array is allocated on the heap and then loop copies the elements from old array to new array and then assigning address of new array to old one. Next inside of this function we take user prompts and store information in index number stored in size ,after that incrementing size for next heap .Another Void Function showordernum displays the information and the status of the order by taking the order number as user input . Void Function menudisp is used to show the menu to user .In the main function I initialize some variables including the size and q later used in while loop to ask for user input to quit or continue, along with a dynamic array of structure place. Used a switch statement to select and call the defined functions discussed previously. I also function pointer for better memory handling.

# Task 2:

#include <iostream>

#include <cstdlib>

using namespace::std;

union Employeedetails{

    float salary;

};

struct Employee{

    string name;

    int id;

    string department;

    Employeedetails sal;

};

void display(Employee temp, int size)

{

        cout << "Details of Employee " << endl;

        cout << "Name: " << temp.name << endl;

        cout << "id: " << temp.id << endl;

        cout << "Department: " << temp.department << endl;

}

void search\_id(Employee \*temp, int size, int iid)

{

    for(int i = 0 ; i < size ; i++)

    {

        if(iid == temp[i].id)

        {

            cout << "Id found: " << temp[i].id << endl;

            display(temp[i], size);

            break;

        }

    }

}

void totalsal(Employee \*temp, int size)

{

    float total;

    for(int i = 0 ; i < size ; i++)

    {

        total += temp[i].sal.salary;

    }

    cout <<"The total salary is: ";

    cout << total;

}

void add(Employee \*temp, int &size) {

    Employee \*tempo = new Employee[size+1];

    for(int i = 0 ; i < size ; i++) {

        tempo[i] = temp[i];

    }

    temp = tempo;

    cout << "Data for Employee " << endl;

        cout << "Enter name: ";

        cin >> temp[size].name;

        cout << "Enter id: ";

        cin >> temp[size].id;

        cout << "Enter department: ";

        cin >> temp[size].department;

        cout << "Enter salary: ";

        cin >> temp[size].sal.salary;

        system("CLS");

    size++;

}

int main()

{

    int size;

    Employee\* all = new Employee[size];

    int quit = 0;

    while(quit != 1)

    {

        int sel;

        cout << "Enter 1 to add new Employee record: " << endl;

        cout << "Enter 2 to display information according to id: " << endl;

        cout << "Enter 3 to display total salary of employees: " << endl;

        cin >> sel;

        switch(sel)

        {

            case(1):

                {

                    system("CLS");

                    add(all, size);

                    break;

                }

            case(2):

                {

                    system("CLS");

                    int id;

                    cin >> id;

                    system("CLS");

                    cout << "Enter the id you want to search for: ";

                    search\_id(all, size + 1 , id);

                    break;

                }

            case(3):

                {

                    system("CLS");

                    void (\*ptr)(Employee \*, int) = &totalsal;

                    totalsal(all, size + 1);

                    break;

                }

         }

         cout << endl << "Enter 1 to quit or enter any other key to continue: ";

         cin >> quit;

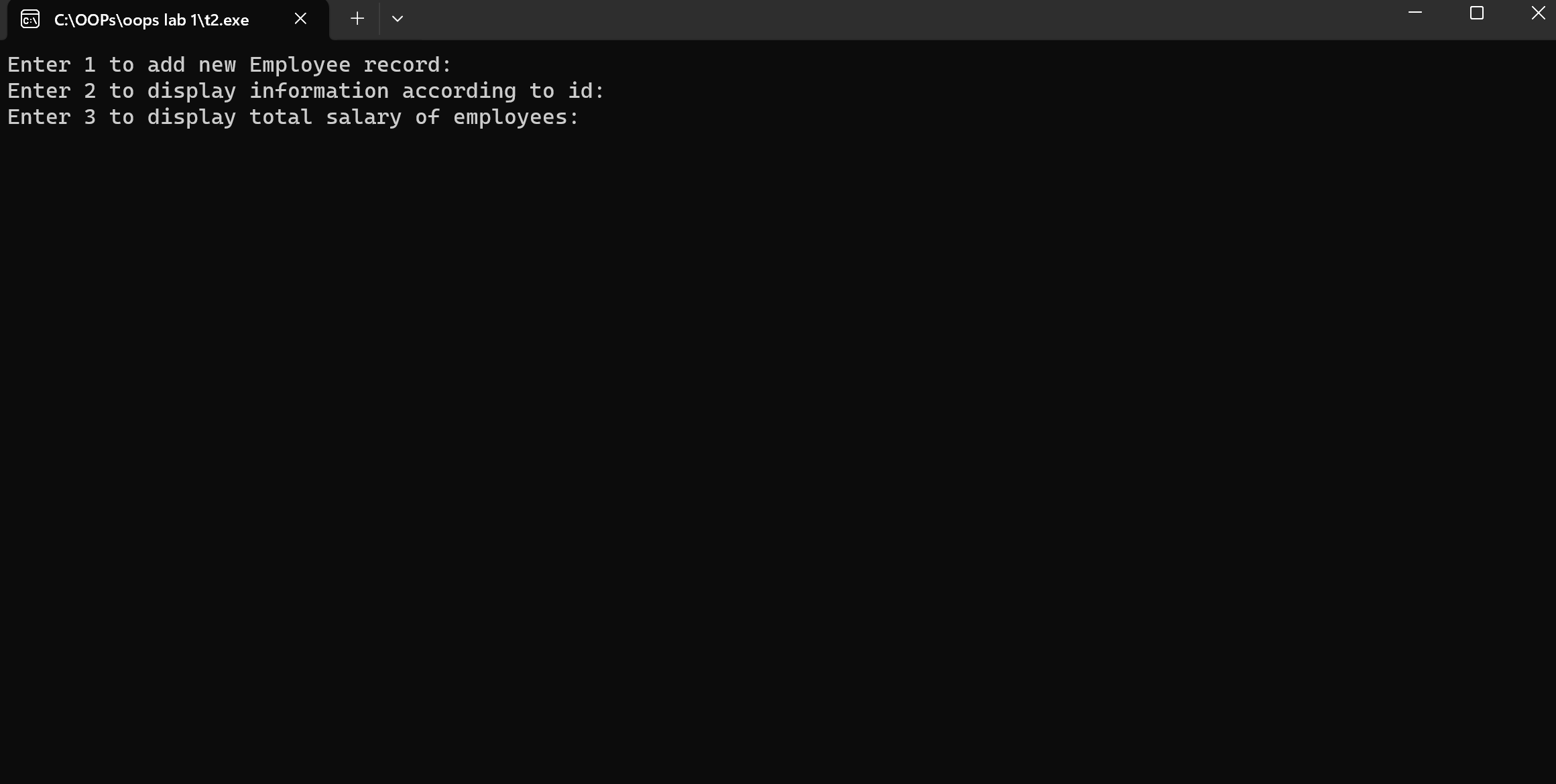
         system("CLS");

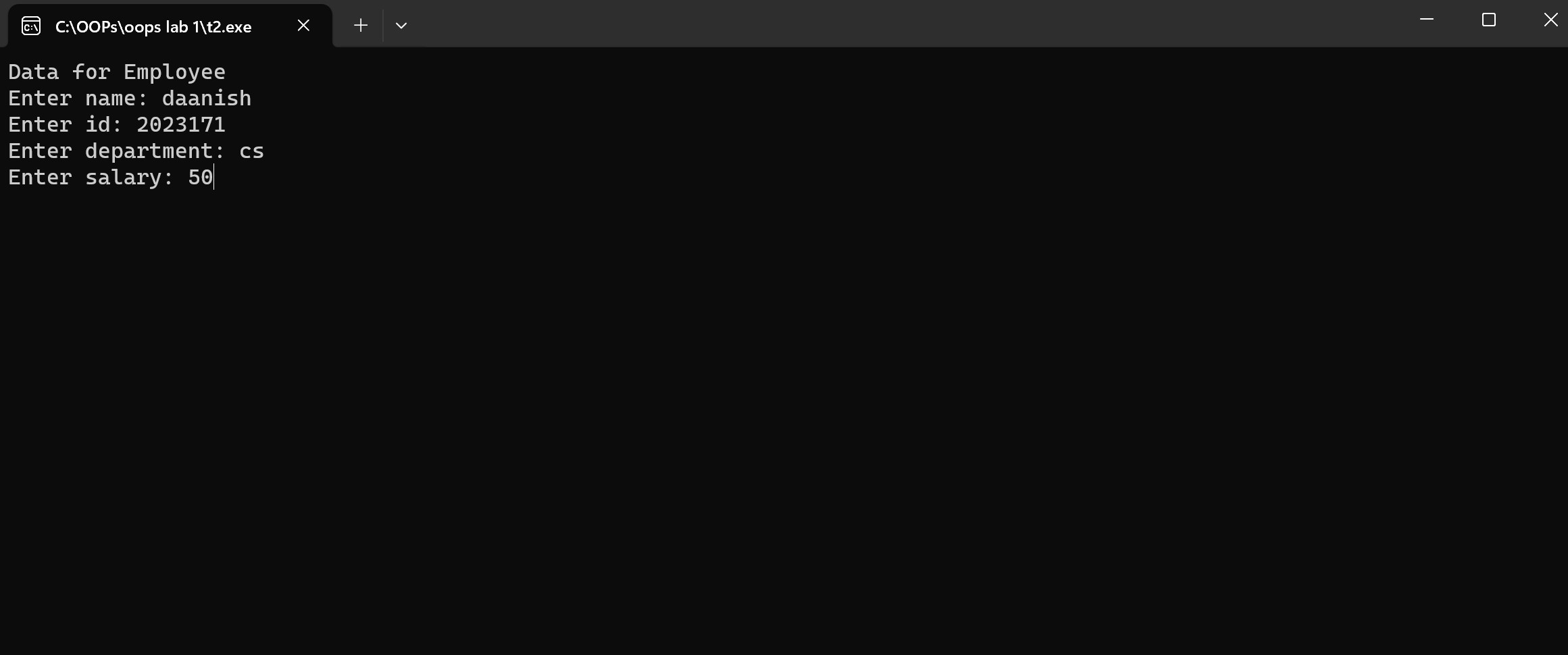
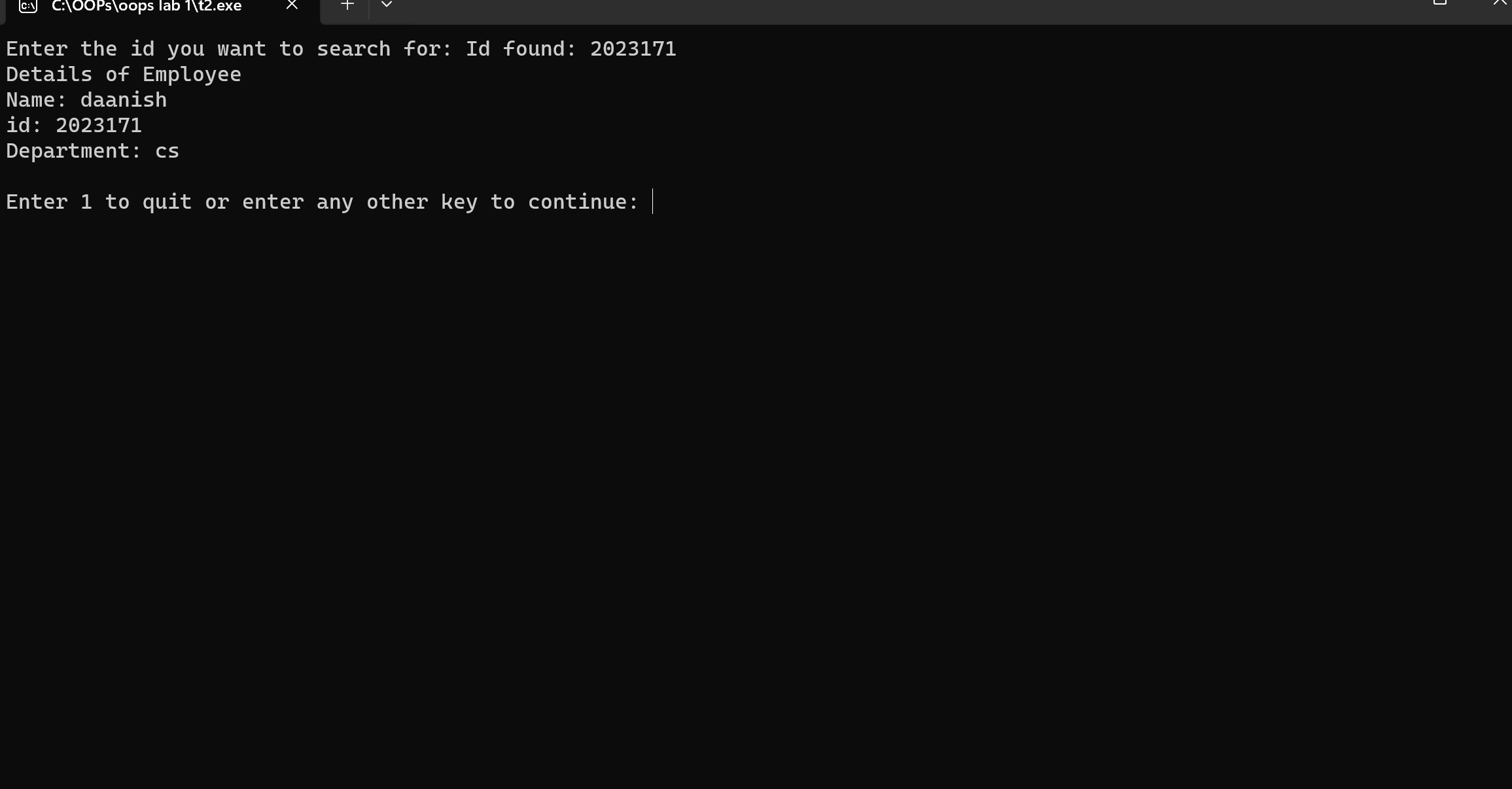
    }

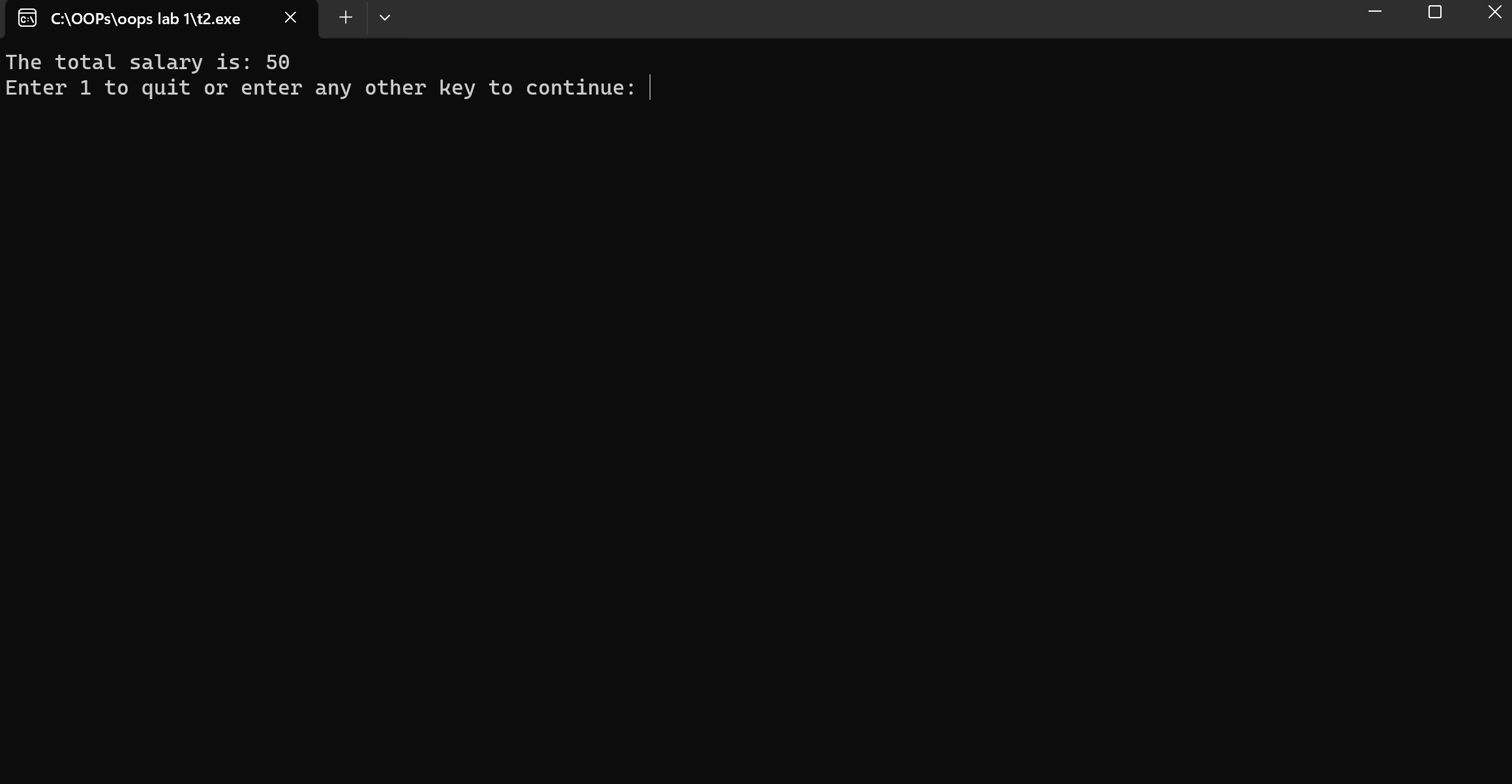
    return 0;

}

Output:





By adding more employees the total salary increases.

Using a union Employee details we have one variable in it float salary which has been called as an instance as sal in a structure called Employee. The structure Employee has following variables: string name, int id, string department. The void function display has been used to display information of specified employee by , giving their name, id and department. The void function search\_id utilizes a for loop and an if statement to search for the given id passed by the user, when the id matches it calls the previously discussed display function. Void function total sal utilizes a for loop the calculate the total of all the employees. Void function add creates a new array of structure place objects with a size of size+1. The memory for this new array is allocated on the heap and then loop copies the elements from old array to new array and then assigning address of new array to old one. Next inside of this function we take user prompts and store information in index number stored in size ,after that incrementing size for next heap. The main function first of all has all variables initialized ,along with creating a dynamic array structure called all. The variable q is used to take user input after each cycle to continue or quit. Through a switch statement we are able to call functions previously discussed.

# Task 3:

#include <iostream>

#include <cstdlib>

using namespace::std;

struct data{

    int roll\_num;

    string name;

    int age;

    int address;

};

void display(data\* temp, int size)

{

    for(int i = 0 ; i < size ; i++)

    {

        if(temp[i].age == 14)

        {

            cout << temp[i].name << ", ";

        }

    }

}

void even(data\* temp, int size)

{

    for(int i = 0 ; i < size ; i++)

    {

        if((temp[i].roll\_num % 2) == 0 )

        {

            cout << temp[i].name << ", ";

        }

    }

}

void search\_roll(data\* temp, int rolln, int size)

{

    bool found;

    string tempd;

    for(int i = 0 ; i < size ; i++)

    {

        if(rolln == temp[i].roll\_num)

        {

            found = true;

            tempd = temp[i].name;

            break;

        }

    }

    if(found == true)

    {

       cout << "Roll number found: " << endl;

       cout << tempd;

    }

    else

    {

        cout << "Roll number not found !";

    }

}

int main()

{

    int size;

    cout << "Enter number of students: ";

    cin >> size;

    data\* student = new data [size];

    int quit = 0;

    for(int i = 0 ; i < size ; i++)

    {

        cout << "Entery <" <<i + 1 <<">" << endl << endl;

        cout << "Enter roll number: ";

        cin >> student[i].roll\_num;

        cout << "Enter name: ";

        cin >> student[i].name;

        cout << "Enter age: ";

        cin >> student[i].age;

        cout << "Enter address: ";

        cin >> student[i].address;

        system("CLS");

    }

    while(quit != 1)

    {

    cout << "Enter 1 to print all names of students with age 14" << endl;

    cout << "Enter 2 to print all names of students with even roll numebr" << endl;

    cout << "Enter 3 to search for student by passing roll number" << endl;

    int sel;

    cin >> sel;

    switch(sel)

    {

        case(1):

        {

            system("CLS");

            display(student, size);

            break;

        }

        case(2):

        {

            system("CLS");

            even(student, size);

            break;

        }

       case(3):

       {

        system("CLS");

        int sea;

        cout << "Enter the roll num: ";

        cin >> sea;

        search\_roll(student, sea, size);

        break;

       }

}

cout << endl;

cout << "Enter 1 to exit or any other to continue: ";

cin >> quit;

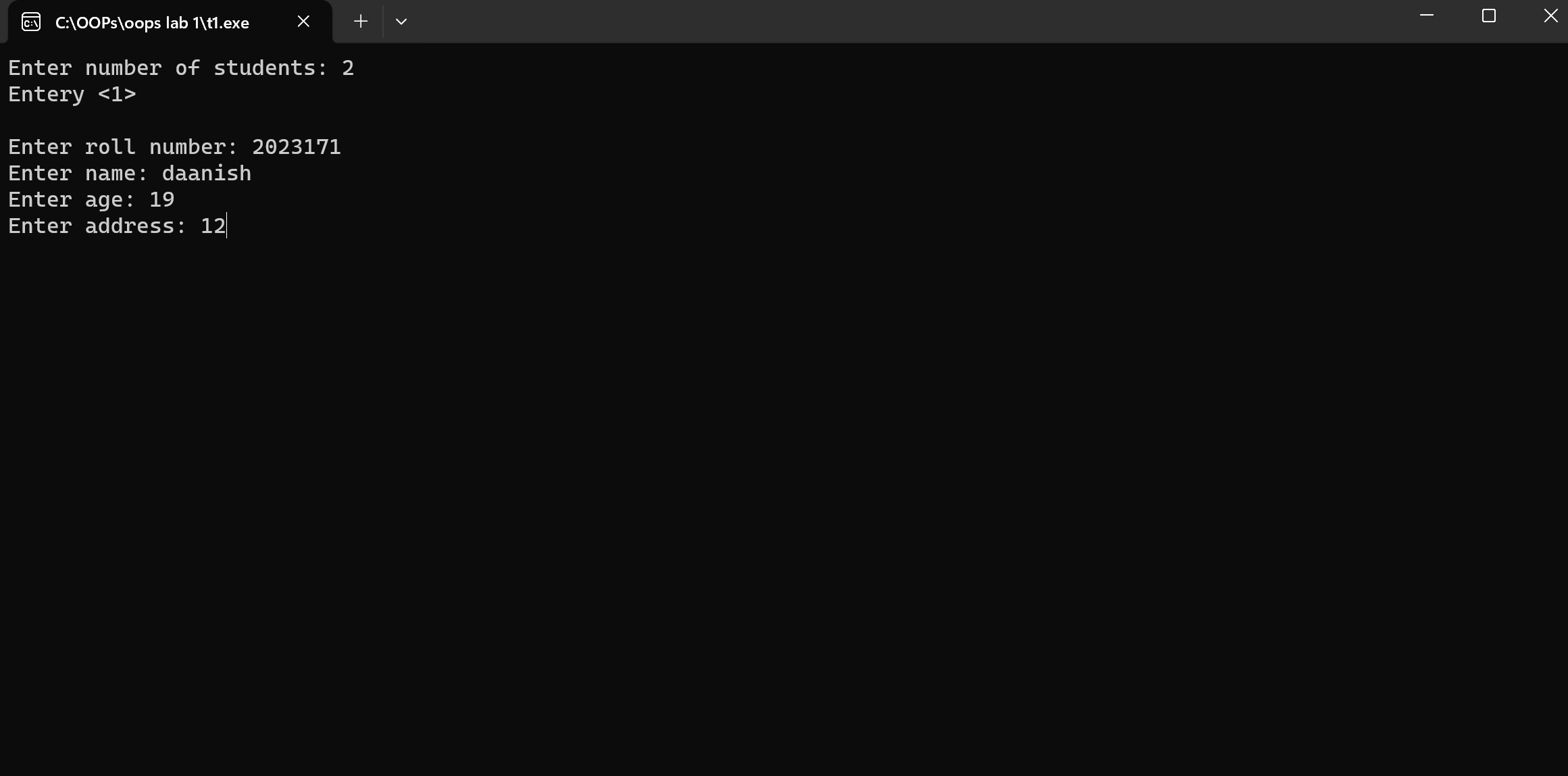
system("CLS");

}

    return 0;

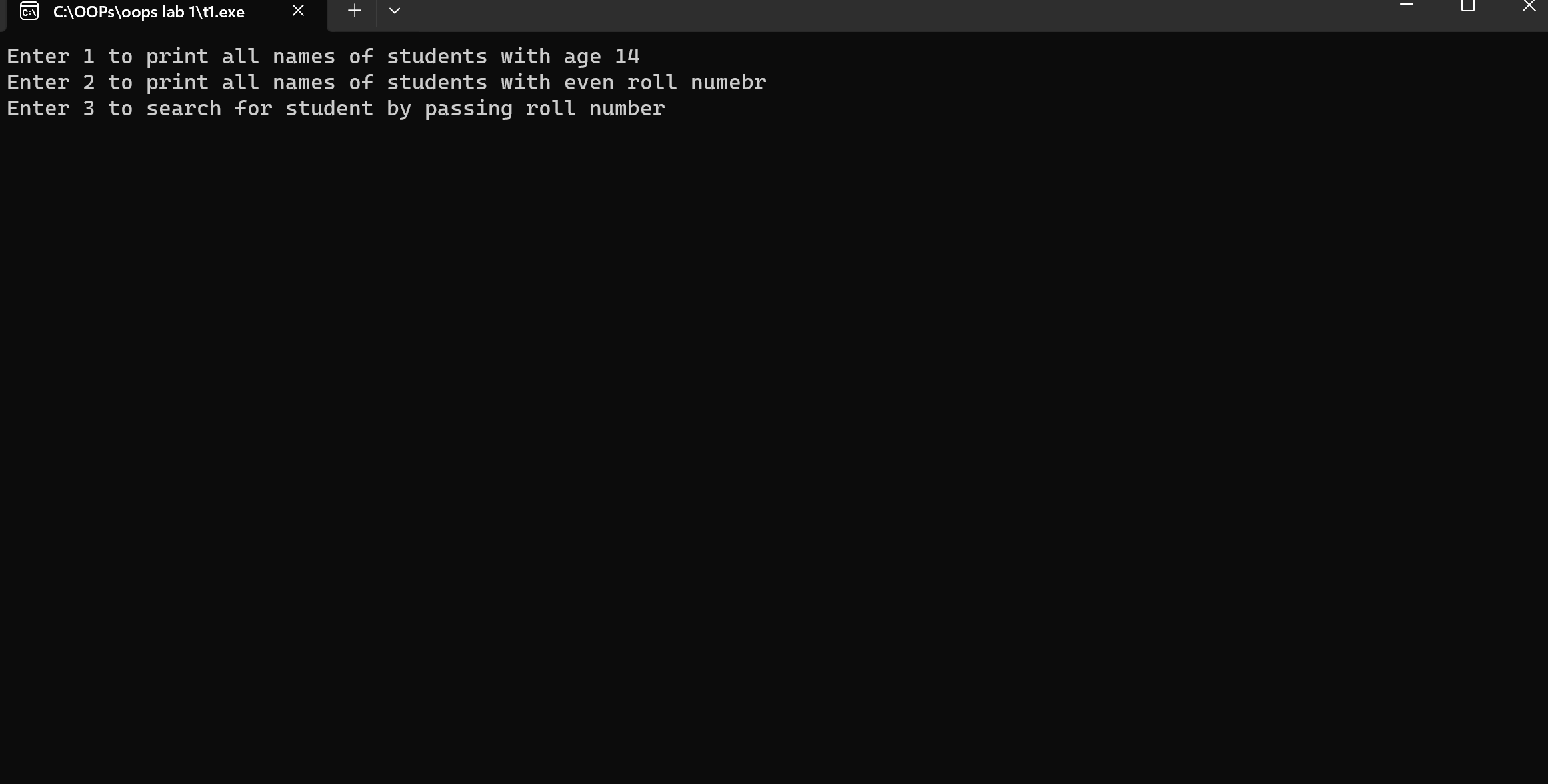
}

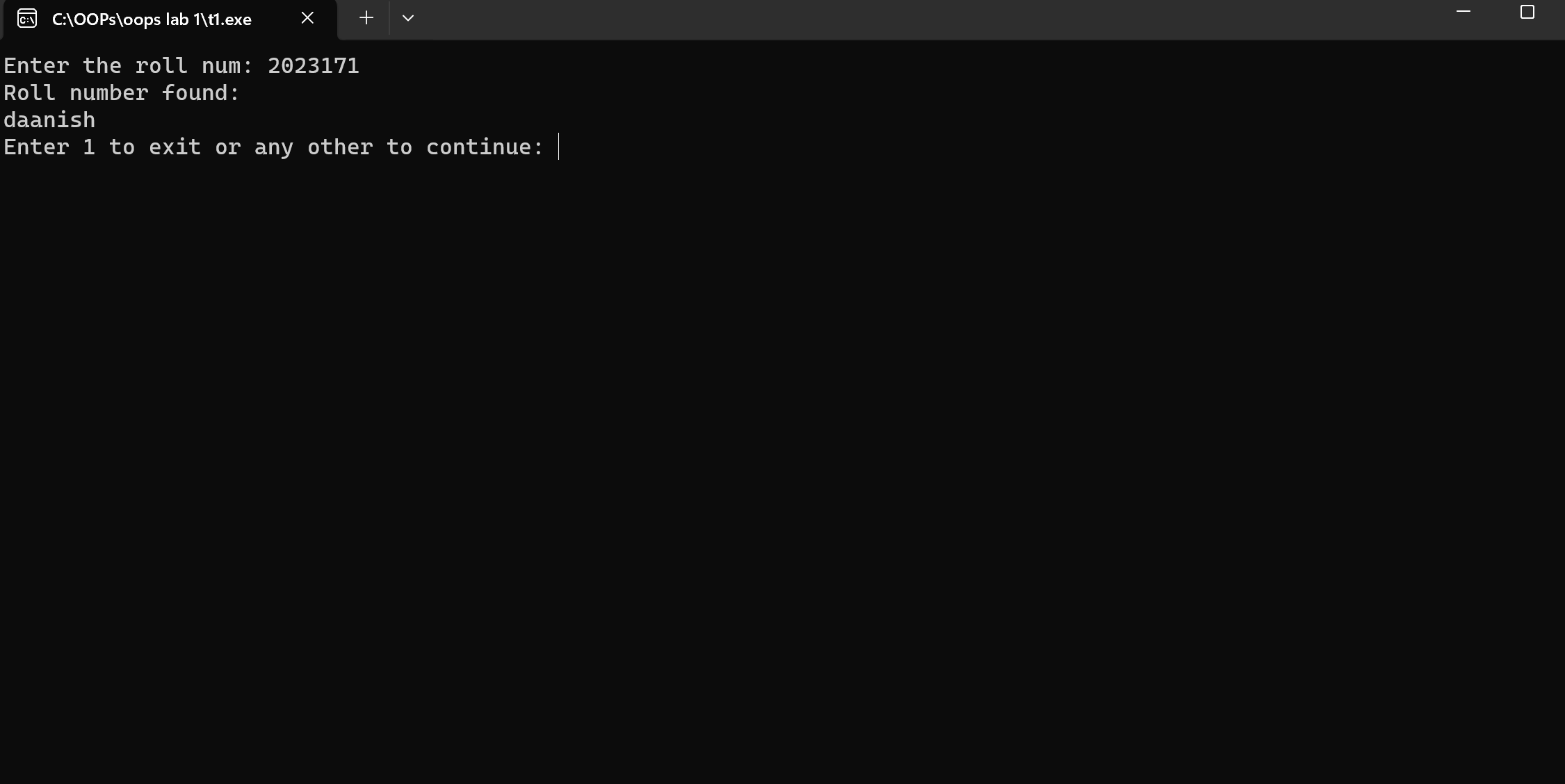
Output:



A screenshot of a computer

Description automatically generated





A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Does not display the students ages of 14 because the two students I input are above 14.

By implementing structure called data I am able to store following variables: int roll\_num, string name, int age and int address. A void function display is used to iterate through the ages of the students and display the names of students with age 14, check implemented by for loop and if statement. A function called void even is used to display the names of students with even number roll numbers, again with use of for loop, if statement with condition of using a mod operator and checking if mod of number by 2 is equal to 0 or not ? Void function search\_roll is used to perform a check through the student instances and find the matching roll number defined by the user. In the main function we initialize a few variables and create a dynamic array structure called student. The variable q is used to take user input after each cycle to continue or quit. Firstly prompts user how many students records to be recorded and stored in variable size. Then using for loop stores information for all the students. Switch statement used to call previously discussed functions.

# Task 4:

#include <iostream>

#include <cstdlib>

using namespace::std;

struct Employee

{

    string name;

    float salary;

    int hours;

};

void store(Employee \*temp, int size)

{

    for(int i = 0 ; i < size ; i++)

    {

        cout << "For Employee " << i+1 << endl;

        cout << "Enter Name: ";

        cin >> temp[i].name;

        cout << "Enter salary: ";

        cin >> temp[i].salary;

        cout << "Enter hours: ";

        cin >> temp[i].hours;

        system("CLS");

    }

}

void increase(Employee \*temp, int size)

{

    for(int i = 0 ; i < size ; i++)

    {

        if(temp[i].hours < 8)

        {

            temp[i].salary += 50;

        }

        else if(temp[i].hours >= 8 && temp[i].hours < 10)

        {

            temp[i].salary += 100;

        }

        else if(temp[i].hours >= 12)

        {

            temp[i].salary += 150;

        }

        else

        {

            cout << "Invalid !";

        }

    }

}

void display(Employee \*temp, int size)

{

    for(int i = 0 ; i < size ; i++)

    {

        cout << "For Employee " << i+1 << endl;

        cout << "Name: " << temp[i].name << endl;

        cout << "salary: " << temp[i].salary << endl << endl;

    }

}

int main()

{

    int size;

    cout << "Enter number of Employees: ";

    cin >> size;

    system("CLS");

    Employee \*all = new Employee[size];

    store(all,size);

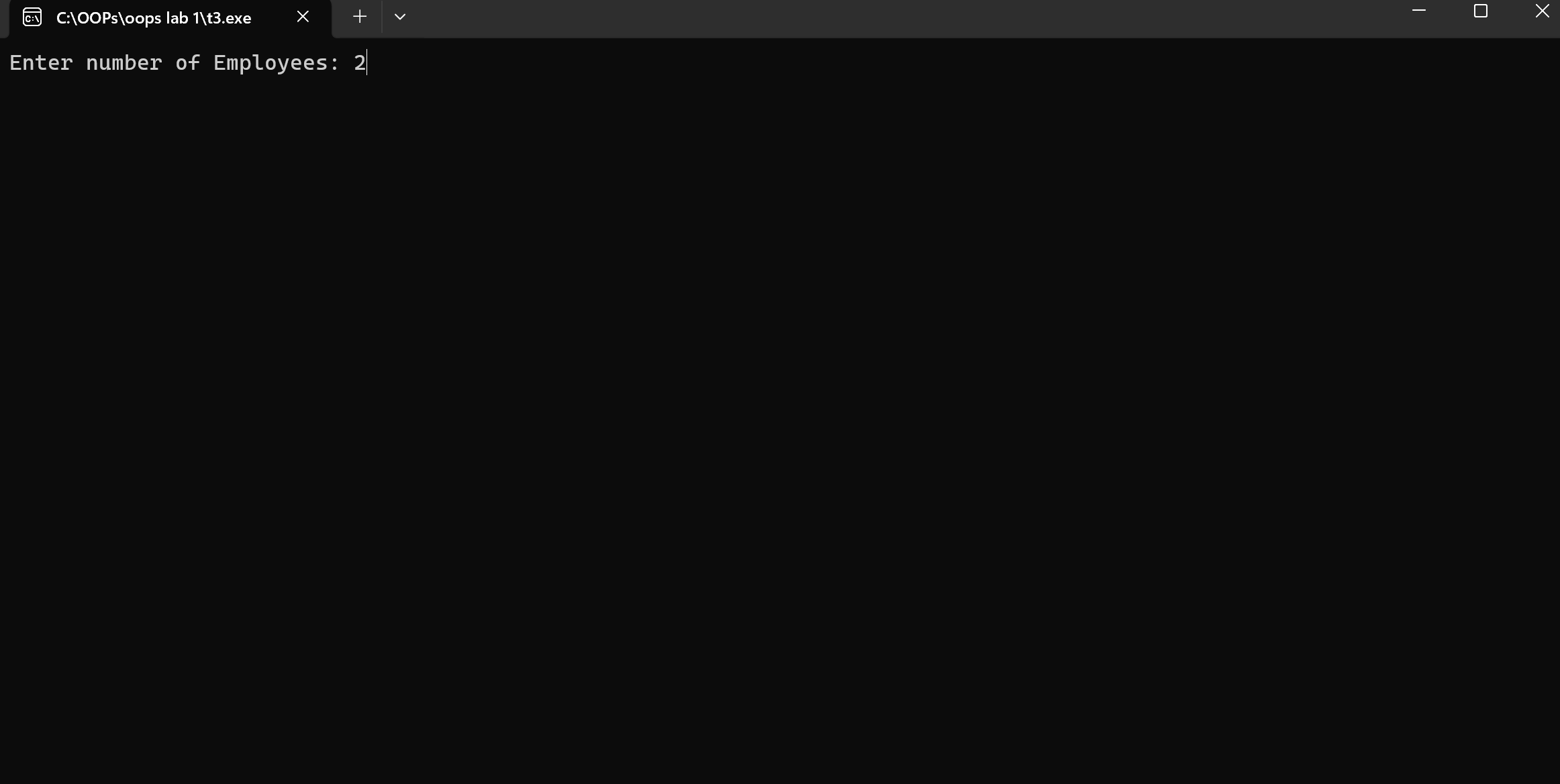
    increase(all, size);

    display(all, size);

    return 0;

}

Output:

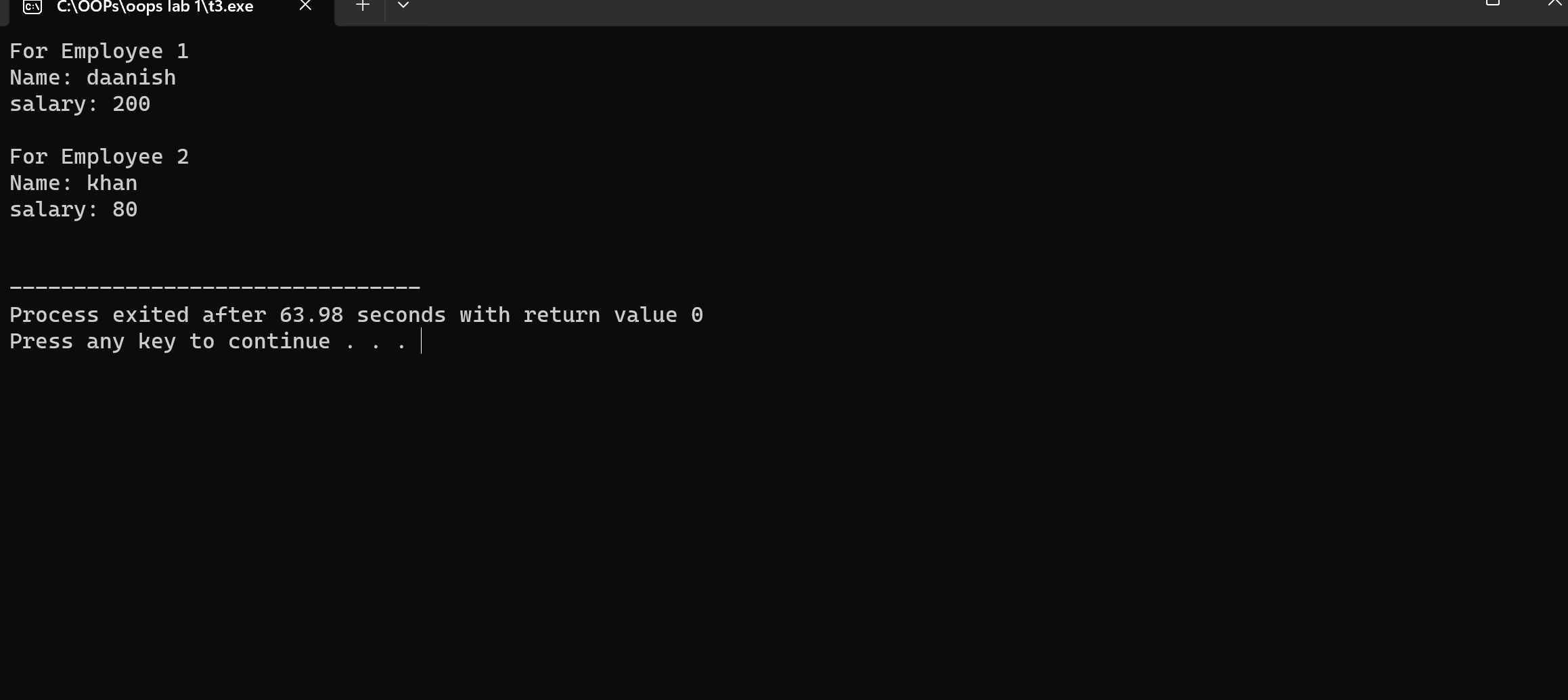


A black screen with a black border

Description automatically generated

A black screen with a black border

Description automatically generated



For above 10 or 15 employees you can enter number of Employees yourself.

Using a structure Employee I create variables string name, float salary, int hours. A void function store is used to iterate through the defined number of employees and take user input for the details. A Void function called increase is used to perform check on employees number of hours and increments their salary on the bases of it. For this I have used a for loop for employee iteration and if statements with comparison operators to check for the qualification of hours. Void function display is used to display the Name and the final salary of the Employees. The main function prompts user to enter the number of employees, create a dynamic array structure called all and call the previously discussed functions.

# Task 5:

#include <iostream>

#include <cstdlib>

using namespace::std;

struct books{

    int accession;

    string author;

    string title;

    bool issue;

};

void display(books \*&temp, int size, int access) {

    for(int i = 0 ; i < size ; i++) {

        if(access == temp[i].accession) {

            cout << "Info for book number " << i+1 << endl;

            cout << "Name of Author: " << temp[i].author << endl;

            if(temp[i].issue == true) {

                cout << "Issued" << endl;

            } else {

                cout << "Not issued" << endl;

            }

            cout << "Tittle: " << temp[i].title << endl;

        }

    }

    cout << endl;

}

void add(books \*&temp, int &size) {

    books \*tempo = new books[size+1];

    for(int i = 0 ; i < size ; i++) {

        tempo[i] = temp[i];

    }

    cout << "Enter access num: ";

    cin >> tempo[size].accession;

    cout << "Enter author name: ";

    cin >> tempo[size].author;

    cout << "Enter book tittle: ";

    cin >> tempo[size].title;

    tempo[size].issue = false;

    temp = tempo;

    size++;

}

int authdisp(books \*temp, string aut, int size)

{

    int counter;

    for(int i = 0  ; i < size ; i++)

    {

        if(aut == temp[i].author)

        {

            counter++;

        }

    }

    return counter;

}

int tittledisp(books \*temp , string titl, int size)

{

    int counter;

    for(int i = 0 ; i < size ; i++)

    {

        if(titl == temp[i].title)

        {

            counter++;

        }

    }

    return counter;

}

int totaldisp(books \*temp , int size, int total)

{

    int counter = 0;

    for(int i = 0 ; i < size ; i++)

    {

       if(temp[i].issue == false)

{

++total;

++counter;

}

}

    return counter;

}

void issue(books \*&temp , int size, int access, int &total)

{

    for (int i = 0 ; i < size ; i++)

    {

        if(access == temp[i].accession)

        {

            temp[i].issue = true;

            total--;

        }

    }

}

int main() {

    int size = 0;

    int total = 0;

    int q = 0;

    books \*all = new books[size];

    while(q != 1) {

        cout << "Enter 1 to display book info: " << endl;

        cout << "Enter 2 to add new book: " << endl;

        cout << "Enter 3 to display book of particular author: " << endl;

        cout << "Enter 4 to display number of books of particular tittle: " << endl;

        cout << "Enter 5 to display total number of books in library: " << endl;

        cout << "Enter 6 to issue books: " << endl;

        int sel;

        cin >> sel;

        system("CLS");

        cout << endl;

        switch(sel) {

            case(1): {

                cout << "Enter access number: ";

                int access;

                cin >> access;

                display(all, size, access);

                break;

            }

            case(2): {

                add(all, size);

                break;

            }

            case(3): {

                cout << "Enter name of author: ";

                string auth;

                cin >> auth;

                cout << authdisp(all, auth, size) << endl;

                break;

            }

            case(4):{

                string tittl;

                cout << "Enter Tittle: ";

                cin >> tittl;

                cout << tittledisp(all, tittl, size) << endl;

                break;

            }

            case(5):{

                cout << totaldisp(all, size , total) << endl;

                break;

            }

            case(6):{

                cout << "Enter access number of the book to issue: ";

                int access;

                cin >> access;

                issue(all, size, access, total);

                break;

            }

        }

    cout << "Enter 1 to quit or enter any other number to contine: ";

    cin >> q;

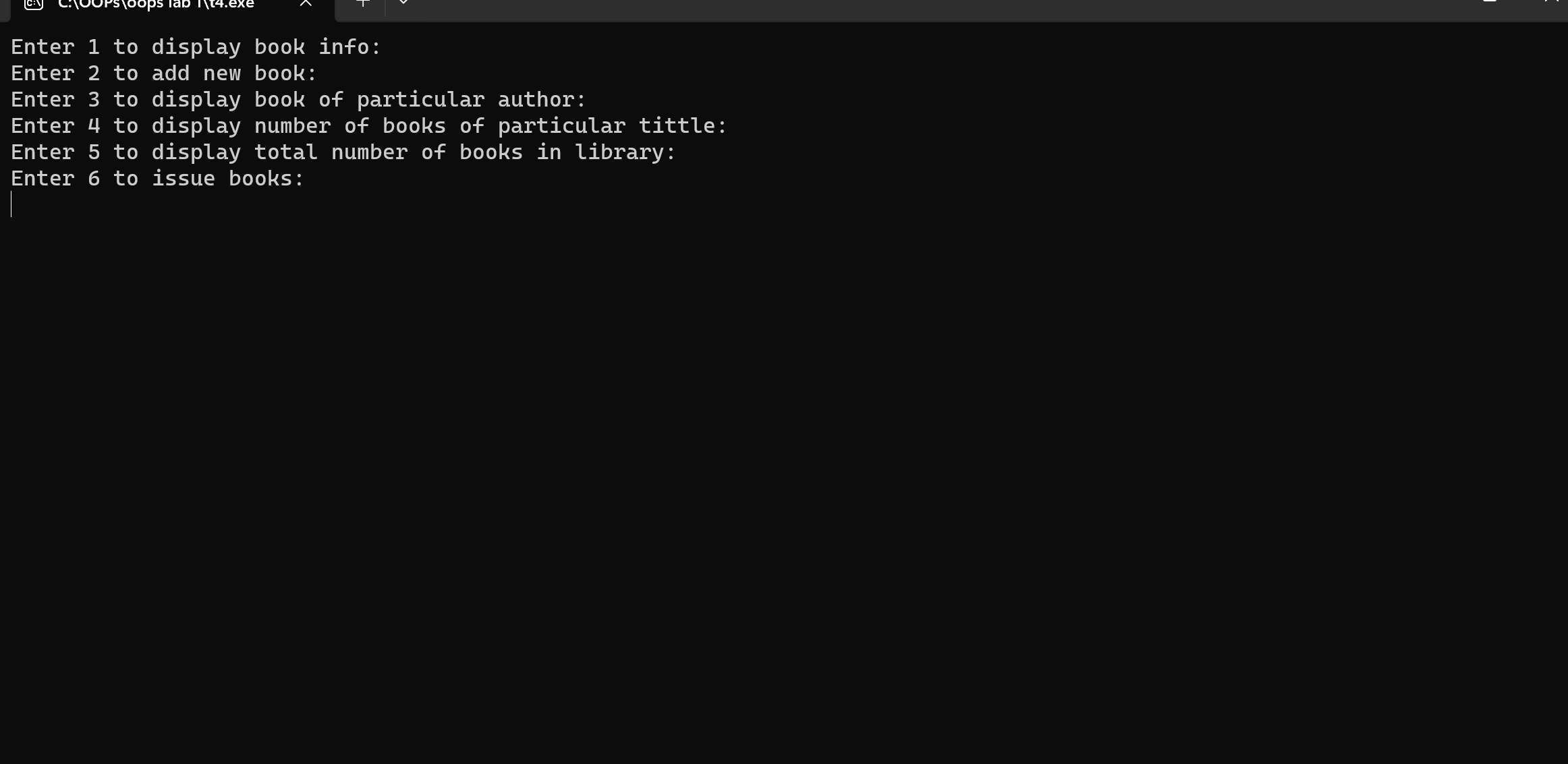
    system("CLS");

}

    return 0;

}

Output:



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

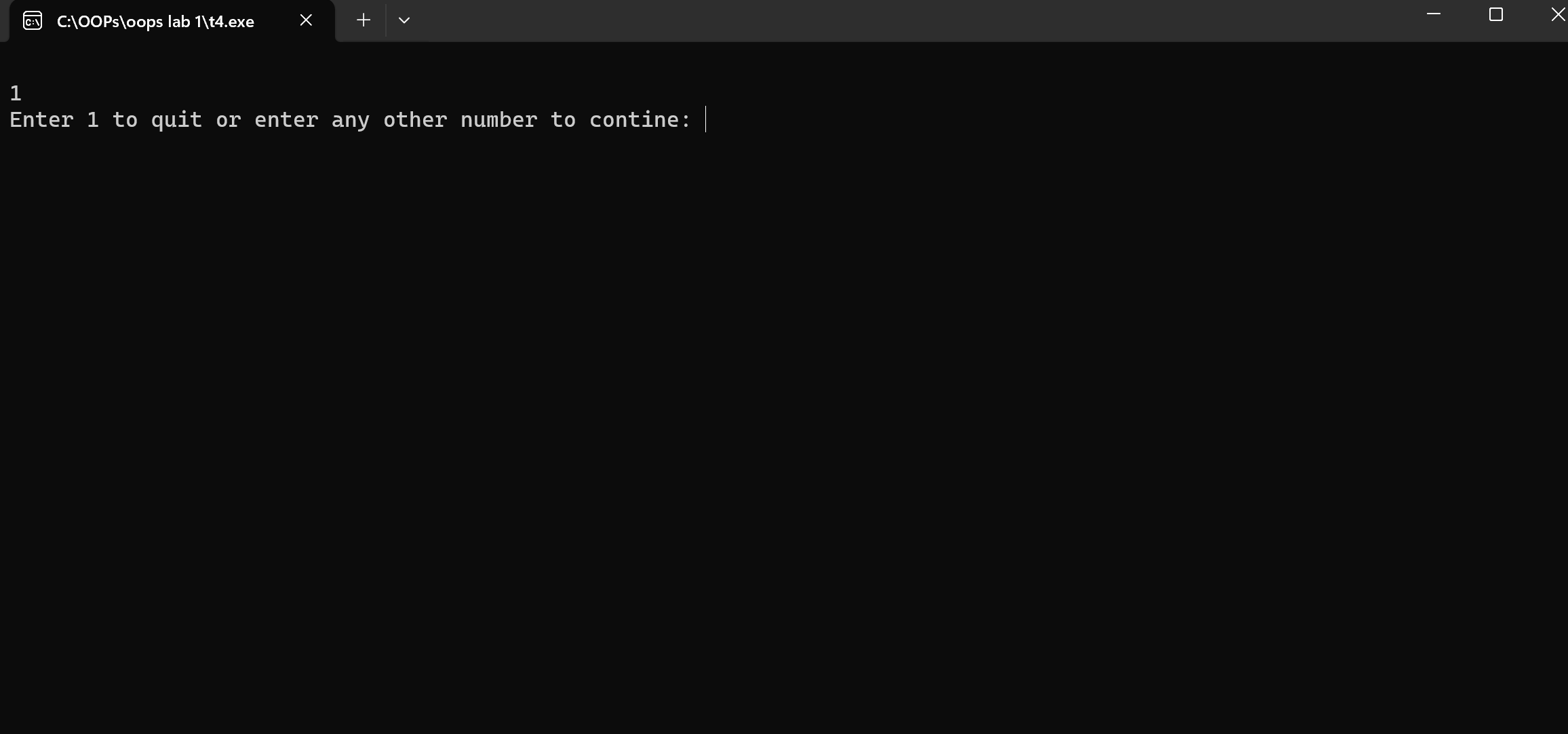
Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

****

**This ^ one shows the total number of books in library which is 1 because when it is issued in decerements, in case you missed it.**

Created a structure called books having variables int accession, string author, string tittle and bool issue. The void function display searches for accession number provided by user and displays the details of the book based on the accession number, it displays the author’s name and state( issued or not). Another void function called add creates a new array of structure books  with a size of size+1. The memory for this new array is allocated on the heap and then loop copies the elements from old array to new array and then assigning address of new array to old one. Next inside of this function we take user prompts and store information of books in index number stored in size ,after that incrementing size for next heap. An integer function authdisp features the number of books written by the author and gives a return value of the counter incremented in a for loop with if statement to check for matching authors. An integer function tittledisp features the number of books with specified tittle and gives a return value of the counter incremented in a for loop with if statement to check for matching tittles. An integer function totaldisp returns the total number of books by incrementing a counter in a for loop with condition of the issue to be false for the number of books which are not issued. Void function issue is uses a Boolean variable to assign true to the books with the user specifying the accession numbers and decrementing the total number of books. In the mainb function some variables are initialized. Variable q is used in a while loop and prompts user to either continue through loop or exit. A dynamic array structure called all is created. With the help of switch statements we call the previously defined functions.