|  |  |
| --- | --- |
| Data Structures & Algorithms Diploma in IT, CSF  Year 2 (2024/25) Semester 4 | Week 2 |
| 2 Hours |
| **Practical 2 – Lists** | |

1. A list is required to store the names and phone numbers of the contacts in a mobile phone.

A sample of the main menu is shown below.

|  |
| --- |
| ---------------- Main Menu -------------------  [1] List the phone numbers  [2] Add a new contact  [3] Remove a contact  [4] Search for a phone number  [0] Exit  ----------------------------------------------  Enter your option : \_\_ |

1. Identify and list the operations required for the application

i) Display all contacts

Operation: Get all items in the list (retrieve), foreach item in list

ii) Add a new contact

Operation: Add an item at the end of the list (appen)

iii) Remove a contact

- By phone number

Operation: Remove an item at given position identified by phone number (remove element from list)

iv) Search for a phone number

Operation: Get the item at a given position in the list (find element in list, probably by number)

1. Specify the operations of the List ADT.
2. getAllPhoneNumbers()
3. add(item: phoneNumber)
4. remove(item: phoneNumber)
5. get(item: phoneNumber)

(c) Implement the operations of the List ADT.

2. Write the application program to use the List ADT to store the names and phone numbers and allow the user to perform the following operations:

(a) List all the names and the phone numbers

// Function to display all contacts

void getAllPhoneNumbers() {

    if (contacts.empty()) {

        cout << "No contacts available.\n";

    } else {

        cout << "\nContacts:\n";

        for (const auto& contact : contacts) {

            cout << "Name: " << contact.getName()

                 << ", Phone: " << contact.getTelNo() << "\n";

        }

    }

}

(b) Add a new contact

// Function to add a new contact

void add(const string& name, const string& telNo) {

    contacts.emplace\_back(name, telNo);

    cout << name << " has been added.\n";

}

(c) Remove a contact

// Function to remove a contact by phone number

void remove(const string& telNo) {

    for (auto it = contacts.begin(); it != contacts.end(); ++it) {

        if (it->getTelNo() == telNo) {

            cout << it->getName() << " has been removed.\n";

            contacts.erase(it);

            return;

        }

    }

    cout << "Contact not found.\n";

}

(d) Search for a phone number

// Function to search for a contact by phone number

void get(const string& telNo) {

    for (const auto& contact : contacts) {

        if (contact.getTelNo() == telNo) {

            cout << "Found: " << contact.getName()

                 << " - " << contact.getTelNo() << "\n";

            return;

        }

    }

    cout << "Contact not found.\n";

}

1. In the same application, create a List ADT to store Module objects. The application adds 3 Module objects to the list and calculates the total for all the objects.

class Module

{

private:

    string name;

    double ca;

    double ct;

    double asg;

public:

    Module(const string name, double ca, double ct, double asg)

    {

        this->name = name;

        this->ca = ca;

        this->ct = ct;

        this->asg = asg;

    }

    // Setters

    void setName(string name) { this->name = name; }

    void setCa(double ca) { this->ca = ca; }

    void setCt(double ct) { this->ct = ct; }

    void setAsg(double asg) { this->asg = asg; }

    // Getters

    string getName() const { return name; }

    double getCa() const { return ca; }

    double getCt() const { return ct; }

    double getAsg() const { return asg; }

    // Calculate total score with weights: ca: 30%, ct: 30%, asg: 40%

    double calculateTotal() const

    {

        return (ca \* 0.3) + (ct \* 0.3) + (asg \* 0.4);

    }

};

void calculateModules() {

    // Create a list of modules

    vector<Module> modules;

    // Add modules to the list

    modules.emplace\_back("Programming", 80, 70, 90);

    modules.emplace\_back("Mathematics", 70, 60, 80);

    modules.emplace\_back("Database", 85, 75, 95);

    // Calculate and display total scores

    for (const auto &module : modules)

    {

        cout << "Module: " << module.getName()

             << ", Total: " << module.calculateTotal() << "\n";

    }

}

Appendix A – *Class Element Diagrams*

|  |
| --- |
| Person |
| -name:String  -telNo:String |
| +Person(String, String)  +setName(String):void  +getName():String  +setTelNo(String):void  +getTelNo():String |

|  |
| --- |
| Module |
| -name:String  -ca:double  -ct:double  -asg:double |
| +Module(String, double, double, double)  +setName(String):void  +getName():String  +setCa(double):void  +getCa():double  +setCt(double):void  +getCt():double  +setAsg(double):void  +getAsg():double  +CalculateTotal(): double // ca:30% ct:30% asg:40% |