Driver.cpp

#include<iostream>

#include<vector>

#include<list>

#include<algorithm>

#include<fstream>

#include<iomanip>

#include "Person.h"

#include "Task.h"

using namespace std;

template< class Container1, class Container2 >

void rotateTasks(Container1 team, Container2 tasks);

//funtion to print menu

void printMenu()

{

    cout << "Please choose an action: \n";

    cout << "1. Search by Name(First and Second) \n";

    cout << "2. Add a new Person\n";

    cout << "3. Delete a Person by entering Phone Number \n";

    cout << "4. Print all persons in the team in current order\n";

    cout << "5. Print all persons in the team in reverse order\n";

    cout << "6. Assign Tasks\n";  //Bonus

    cout << "7. Exit \n";

}

int main()

{

    bool found;

    list<Person> team; //or vector<Person> team;

    ifstream myCin("persons.txt");

    if (myCin.fail())

    {

        cout << "File Not Found \n";

        exit(1);

    }

    string fname, lname;

    int phNum;

    //Reading contacts from file

    while (myCin >> fname >> lname >> phNum)

    {

        Person td(phNum, fname, lname);

        team.push\_back(td);

    }

    myCin.close();

    //This is for bonus

    //Reading tasks from a file

    list<Task> tasks; //or vector<Task> tasks;

    myCin.open("tasks.txt");

    if (myCin.fail())

    {

        cout << "File Not Found \n";

        exit(1);

    }

    int taskID, taskDuration;

    string taskName;

    while (myCin >> taskID >> taskName >> taskDuration)

    {

        Task td(taskID, taskName, taskDuration);

        tasks.push\_back(td);

    }

    myCin.close();

    //Ex1..contd..

    int action = 0;

    printMenu();

    cin >> action;

    while (action != 7)

    {  found = false;

        switch (action)

        {

        case 1: // Search by Name(First and Second)

            cout << "Please enter a First Name and Second Name to search for: ";

            cin >> fname;

            cin >> lname;

            //your code goes here

            for(auto it = team.begin(); it != team.end() || !found; ++it)

            {

                if(it->getFirstName() == fname && it->getLastName() == lname)

                {

                    cout << "Phone Number: " << it->getPhNum() << endl;

                    found = true;

                }

            }

            if(!found)

                cout << "Person not found.\n";

            break;

        case 2: //Add a new Person

            cout << "Enter First Name: ";

            cin >> fname;

            cout << endl;

            cout << "Enter Last Name: ";

            cin >> lname;

            cout << endl;

            cout << "Enter Phone Num: ";

            cin >> phNum;

            cout << endl;

            //your code goes here

            team.push\_back(Person(phNum, fname, lname));

            cout << "New user added.\n";

            break;

        case 3:  //Delete a Person by entering Phone Number

            cout << "Please enter a Phone Number to delete the contact: ";

            cin >> phNum;

            //your code goes here

            for(auto it = team.begin(); it != team.end() || !found; ++it)

            {

                if(it->getPhNum() == phNum)

                {

                    it = team.erase(it);

                    cout << "Person Deleted\n";

                    found = true;

                }

            }

            if(!found)

                cout << "Person not found.\n";

            break;

        case 4: //Print team in current order

            cout << "All persons in the team are: " << endl;

            //your code goes here

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

                cout << it->getFirstName() << "\t" << it->getLastName() << "\t" << it->getPhNum() << endl;

            break;

        case 5: //Print team in reverse order

            cout << "All persons in the team in reverse order are: " << endl;

            //your code goes here

            for(auto it = team.rbegin(); it != team.rend(); ++it)

                cout << it->getFirstName() << "\t" << it->getLastName() << "\t" << it->getPhNum() << endl;

            break;

            //Bonus

        case 6: //Bonus part: Assign Tasks

            rotateTasks(team,tasks);

            //your code goes here

        }

        cout << endl;

        printMenu();

        cin >> action;

    }

    //storing updated contacts back to the file

    ofstream myCout("persons.txt");

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

    {

        myCout << it->getFirstName() << "  " << it->getLastName() << "  " << it->getPhNum() << endl;

    }

    myCin.close();

    myCout.close();

}

template< class Container1, class Container2 >

void rotateTasks(Container1 team, Container2 tasks) {

    cout << "Enter number of days: ";

    int numdays;

    cin >> numdays;

    cout << "Available team members: \n";

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

        cout << it->getFirstName() << " ";

    }

    cout << "\n\t";

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

        cout << it->getTasktName() << "\t";

    }

    cout << "Enjoying No Duty..........\n\n";

    for (int i = 1; i <= numdays; i++) {

        cout << "day" << i << "\t";

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

            cout << it->getFirstName() << "\t";

        }

        std::rotate(team.rbegin(), std::next(team.rbegin()),team.rend());

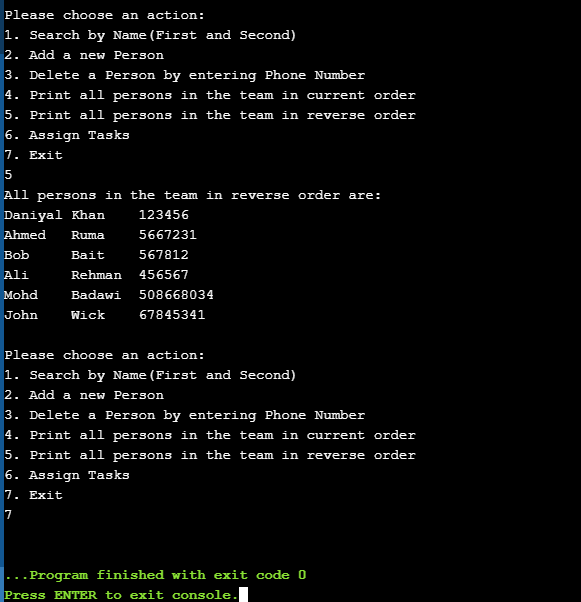
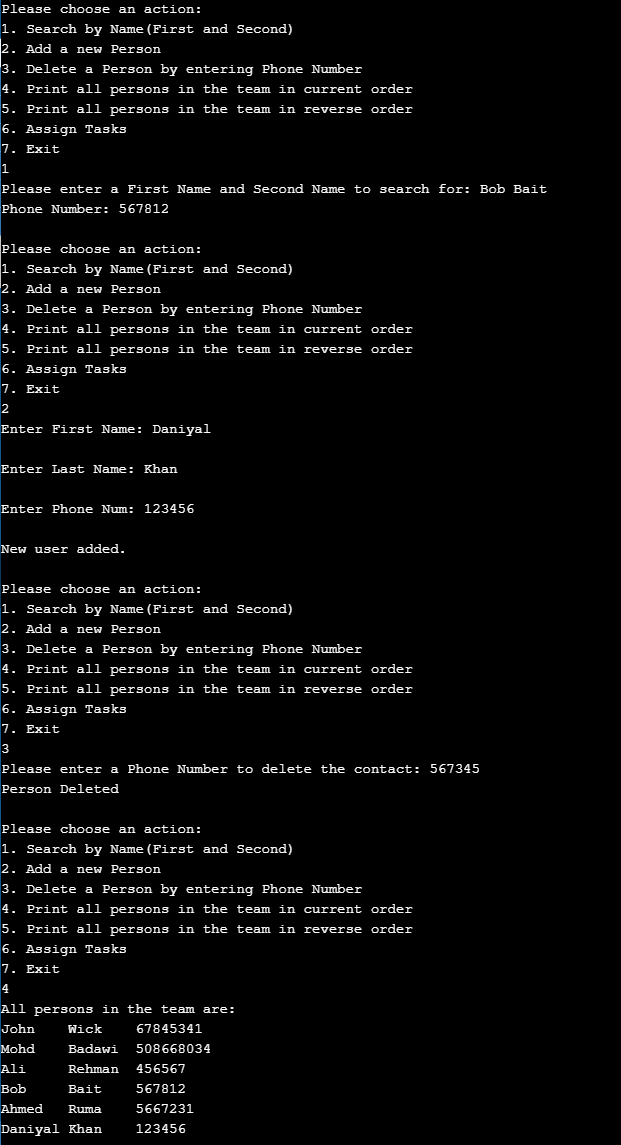
        cout << endl << endl;

    }

}

Screenshots:

Exercise 1:



Exercise 2:

