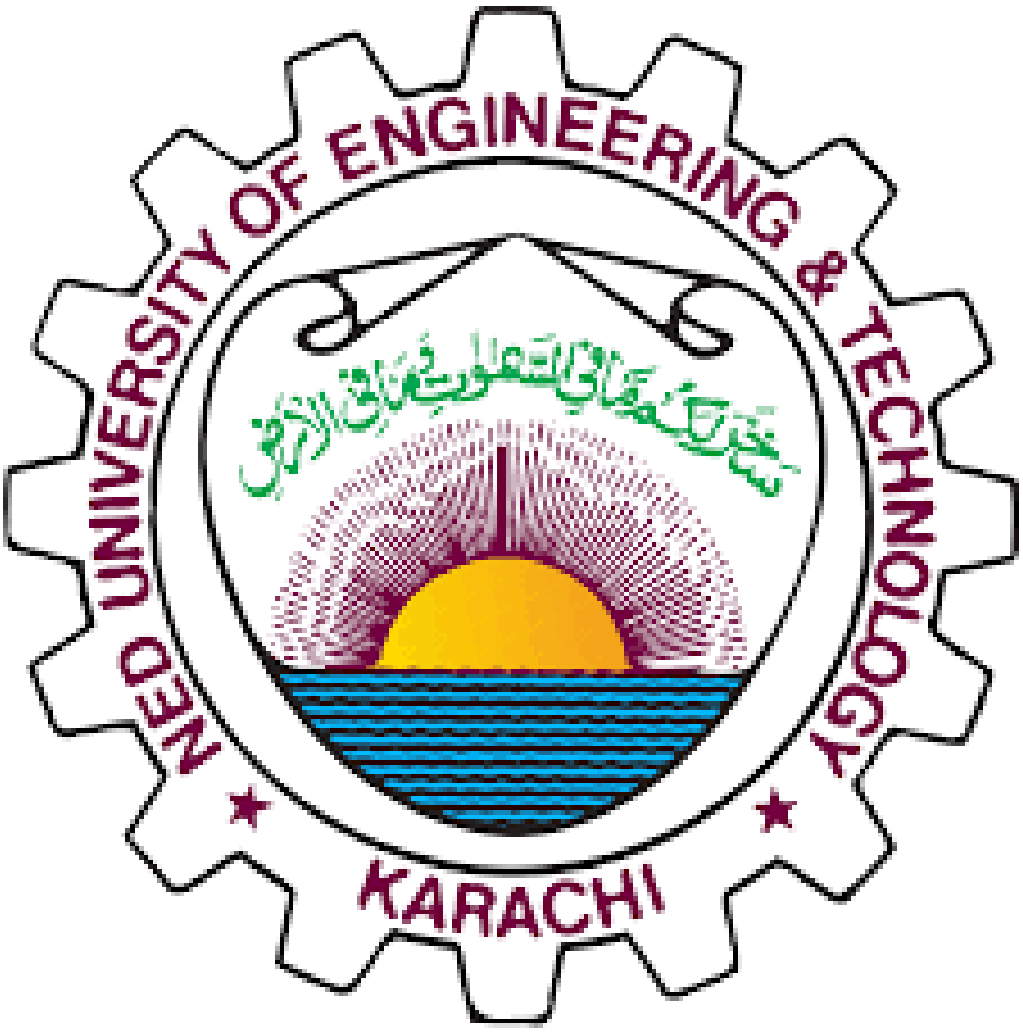

OBJECT ORIENTED PROGRAMMING (CT-260)

GROUP MEMBERS

NAME: RANIA IMRAN ROLL NUMBER: CT-23005

NAME: FATIMA KHAN ROLL NUMBER: CT-23024

NAME: NEHA ROLL NUMBER: CT-23022

NAME: ALIZA BAIG ROLL NUMBER: CT-23021

Exercise

1. Create a C++ Program to implement login. It should accept user name and password and throw a custom exception if the password has less than 6 characters or does not contain a digit.

SOURCE CODE:

```
#include<iostream>
#include<stdexcept>
using namespace std;
class passwordError{
    string message;
    public:
    passwordError(const string &msg):message(msg){}
   string what() const{
        return message;
    }
};
bool isDigit(char c){
    return c>= '0' && c<='9';
}
void validatePassword(const string &password){
    if(password.length()<6){
        throw passwordError("Password must be atleast 6 characters long");
    }
    bool hasDigit = false;
    for(int i = 0; i < password.length(); i++){
        if(isDigit(password[i])){
            hasDigit = true;
            break;
        }
    }
    if(!hasDigit){
        throw passwordError("Password must have atleast 1 digit");
    }
}
void login(){
    string username, password;

    bool validpassword = false;
    while(!validpassword){
    cout<<"Enter username: ";
    getline(cin, username);

    cout<<"Enter password: ";
    getline(cin, password);
```

```
    try{
        validatePassword(password);
         cout<<"Login successful"<<endl;

    }

    catch (const passwordError& e){
        cout<<"Login failed"<<e.what()<<endl;
    }
    }
}
int main(){
    login();
    return 0;
}
```

OUTPUT:

```
PS C:\Users\DC\Desktop\vs code> cd "c:\Users\DC\Desktop"
$?) { .\tempCodeRunnerFile }
Enter username: RANIA
Enter password: rani
Login failedPassword must be atleast 6 characters long
Enter username: Rania
Enter password: rania123
Login successful
```

2. Create a class for dynamic stack using vectors. Implement the push( ), pop( ) and peek() functions. The object of stack class will be used to take any sentence as input and it should have a method reverse( ) which will reverse each word in the string.

SOURCE CODE:

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
class Stack {
private:
    vector<char> stack;
public:
    void push(char c) {
        stack.push_back(c);
    }
    void pop() {
        if (!stack.empty()) {
```

```
            stack.pop_back();
        } else {
            cerr << "Stack is empty. Cannot pop." << endl;
        }
    }
    char peek() const {
        if (!stack.empty()) {
            return stack.back();
        } else {
            cerr << "Stack is empty. Cannot peek." << endl;
            return '\0';
        }
    }

    bool isEmpty() const {
        return stack.empty();
    }

    string reverse(const string& sentence) {
        string result;
        string word;

        for (char c : sentence) {
            if (c == ' ') {
                while (!isEmpty()) {
                    result += peek();
                    pop();
                }
                result += ' ';
            } else {
                push(c);
            }
        }
        while (!isEmpty()) {
            result += peek();
            pop();
        }

        return result;
    }
};

int main() {
    Stack stack;
    string sentence;
```

```
    cout << "Enter a sentence: ";
    getline(cin, sentence);

    string reversedSentence = stack.reverse(sentence);
    cout << "Reversed sentence: " << reversedSentence << endl;

    return 0;
}
```

OUTPUT:

```
PS C:\Users\DC\Desktop\vs code> cd "c:\Users\
P_LAB_11_02 }
Enter a sentence: blah blah blah
Reversed sentence: halb halb halb
PS C:\Users\DC\Desktop\vs code\C++> cd "c:\Us
.\OOP_LAB_11_02 }
Enter a sentence: hello
Reversed sentence: olleh
PS C:\Users\DC\Desktop\vs code\C++>
```

3. You are given N integers. Store N integers in a vector and write a function Sort for sorting the N integers and print the sorted order. Now use the sorting algorithms provided in STL for sorting the vector. Measure the time taken by the two methods for sorting the vector and print the results. [Hint: You can use built-in function time( ) or anyother built-in method for measuring the processing time of sorting operation.]

SOURCE CODE:

```
#include<iostream>
#include<vector>
#include<string>
#include<ctime>
#include<bits/stdc++.h>
#include<chrono>
using namespace std;
using namespace std::chrono;
void sort(vector<int> vec){
    auto start = high_resolution_clock::now();
    for(int i = 0; i < vec.size(); i++ ){
        for(int j = 0; j < vec.size() - i - 1; j++){
            if(vec[j] > vec[j+1]){
                int temp = vec[j];
                vec[j] = vec[j+1];
                vec[j+1] = temp;
            }
```

```
        }
    }
    cout<<"Sorted \n";
    for(auto x : vec){
        cout<<x<<" ";
    }
    cout<<endl;
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stop-start);
    cout<<"Time taken:"<<endl;
    cout<<duration.count()<<" microseconds"<<endl;
}
int main(){
    vector<int> v { 4, 6, 3, 10, 1, 7, 4, 2, 25, 27, 14, 28, 21, 75, 21, 77, 90};

    auto start = high_resolution_clock::now();
    sort(v.begin(), v.end());
    auto stop = high_resolution_clock::now();
    cout<<"Sorted \n";
    for(auto x : v){
        cout<<x<<" ";
    }
    cout<<endl;

    auto duration = duration_cast<microseconds>(stop-start).count();
    cout<<"Time taken:"<<endl;
    cout<<duration<<" microseconds"<<endl;

    sort(v);

    return 0;
}
```

OUTPUT:

```
Sorted
1 2 3 4 4 6 7 10 14 21 21 25 27 28 75 77 90
Time taken:
0 microseconds
Sorted
1 2 3 4 4 6 7 10 14 21 21 25 27 28 75 77 90
Time taken:
1891 microseconds
PS C:\Users\DC\Desktop\vs code\C++>
```

4. You are a teacher and want to keep track of your students' grades. You decide to use the map container in C++ to store the student names as keys and their corresponding grades as values. Design and implement a program in C++ to fulfill the following requirements:

- Input: Prompt the user to enter the names and grades of multiple students. Allow the user to continue entering data until they choose to stop.

- Storage: Store the student names as keys and their grades as values in a map container.

- Retrieval: Implement a function to retrieve the grade associated with a given student's name. If the student is not found in the map, display an appropriate message.

- Update: Implement a function to update the grade of a specific student. Prompt the user to enter the new grade and update the corresponding value in the map.

- Deletion: Implement a function to remove a student and their grade from the map based on a given student's name.

- Display: Display all the student names and their corresponding grades stored in the map.

Testing: Write a program to test the functionality of your map implementation. Add multiple students with their grades, retrieve grades for specific students, update grades, delete students, and display the final list of students and their grades. As you implement the program, consider using appropriate data types, input validation, error handling, and clear user prompts. Remember to thoroughly test your implementation to ensure correctness and accuracy in storing and retrieving the student grades.

SOURCE CODE:

```
#include <iostream>
#include <string>
#include <map>

using namespace std;

void add_student(map<string, int> &students){
    string name;
    int grade;
    cout << "Enter student name: ";
    cin >> name;
    cout << "Enter student grade: ";
    while(true){
        cin >> grade;
        if(cin.fail() || grade < 0 || grade > 100){
            cin.clear();   //clear the error flags
            cin.ignore(1000, '\n');   //discard invalid input
            cout << "Invalid input. Enter a number between 0 and 100: ";
        }
        else{
            cin.ignore(1000, '\n');  // discard any remaining input
```

```
            break;
        }
    }
    students[name] = grade;
    cout << "Student added successfully.\n";
}
void retrieve_grade(const map<string, int> &students){
    string name;
    cout << "Enter student name to retrieve grade: ";
    cin >> name;
    auto it = students.find(name);
    if (it != students.end()){
        cout << "Grade of " << name << ": " << it->second << "\n";
    }
    else{
        cout << "Student not found.\n";
    }
}
void update_grade(map<string, int> &students){
    string name;
    int new_grade;
    cout << "Enter student name to update grade: ";
    cin >> name;
    auto it = students.find(name);
    if(it != students.end()){
        cout << "Enter new grade for " << name << ": ";
        while(true){
            cin >> new_grade;
            if(cin.fail() || new_grade < 0 || new_grade > 100){
                cin.clear();   //clear the error flags
                cin.ignore(1000, '\n');   //discard invalid input
                cout << "Invalid grade. Enter a number between 0 and 100: ";
            }
            else{
                cin.ignore(1000, '\n');  // discard any remaining input
                break;
            }
        }
        it->second = new_grade;
        cout << "Grade updated successfully.\n";
    }
    else{
        cout << "Student not found.\n";
    }
}
```

```
void delete_student(map<string, int> &students){
    string name;
    cout << "Enter student name to delete: ";
    cin >> name;
    auto it = students.find(name);
    if(it != students.end()){
        students.erase(it);
        cout << "Student " << name << " deleted.\n";
    }
    else{
        cout << "Student not found.\n";;
    }
}
void display(const map<string, int> &students){
    if(students.empty()){
        cout << "No students in the list.\n";
    }
    else{
        cout << "Student grades:\n";
        for(const auto &entry : students){
            cout << "Name: " << entry.first << ", Grade: " << entry.second <<
"\n";
        }
    }
}
int main(){
    map<string, int> students;
    int choice;

    do {
        cout << "\nMenu:\n";
        cout << "1. Add student\n";
        cout << "2. Retrieve grade\n";
        cout << "3. Update grade\n";
        cout << "4. Delete student\n";
        cout << "5. Display all students\n";
        cout << "6. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                add_student(students);
                break;
            case 2:
```

```
                retrieve_grade(students);
                break;
            case 3:
                update_grade(students);
                break;
            case 4:
                delete_student(students);
                break;
            case 5:
                display(students);
                break;
            case 6:
                cout << "Exiting the program.\n";
                break;
            default:
                cout << "Invalid choice. Please try again.\n";
                cin.clear();
                cin.ignore(1000, '\n');
        }
    }
    while (choice != 6);

    return 0;
}
```

OUTPUT:

```
Menu:
1. Add student
2. Retrieve grade
3. Update grade
4. Delete student
5. Display all students
6. Exit
Enter your choice: 1
Enter student name: Fatima
Enter student grade: A
Invalid input. Enter a number between 0 and 100: 96
Student added successfully.

Menu:
1. Add student
2. Retrieve grade
3. Update grade
4. Delete student
5. Display all students
6. Exit
Enter your choice: 3
Enter student name to update grade: Fatima
Enter new grade for Fatima: 99
Grade updated successfully.

Menu:
1. Add student
2. Retrieve grade
3. Update grade
4. Delete student
5. Display all students
6. Exit
Enter your choice:
1
Enter student name: Neha
Enter student grade: 100
Student added successfully.
```

```
Menu:
1. Add student
2. Retrieve grade
3. Update grade
4. Delete student
5. Display all students
6. Exit
Enter your choice: 3
Enter student name to update grade: Neha
Enter new grade for Neha: 96
Grade updated successfully.

Menu:
1. Add student
2. Retrieve grade
3. Update grade
4. Delete student
5. Display all students
6. Exit
Enter your choice: 2
Enter student name to retrieve grade: Fatima
Grade of Fatima: 99

Menu:
1. Add student
2. Retrieve grade
3. Update grade
4. Delete student
5. Display all students
6. Exit
Enter your choice: 5
Student grades:
Name: Fatima, Grade: 99
Name: Neha, Grade: 96
```

```
Menu:
1. Add student
2. Retrieve grade
3. Update grade
4. Delete student
5. Display all students
6. Exit
Enter your choice: 4
Enter student name to delete: Neha
Student Neha deleted.

Menu:
1. Add student
2. Retrieve grade
3. Update grade
4. Delete student
5. Display all students
6. Exit
Enter your choice: 5
Student grades:
Name: Fatima, Grade: 99

Menu:
1. Add student
2. Retrieve grade
3. Update grade
4. Delete student
5. Display all students
6. Exit
Enter your choice: 6
Exiting the program.
PS C:\Users\DC\Desktop\vs code\C++>
```

5. You are hosting a party and want to keep track of the unique guests attending. To achieve this, you decide to use the set container in C++ to store the names of the guests.

Design and implement a program in C++ to fulfill the following requirements:

- Input: Prompt the user to enter the names of multiple guests attending the party. Allow the user to continue entering names until they choose to stop.

- Storage: Store the guest names in a set container, which will automatically enforce uniqueness by discarding duplicate names.

- Display: Display all the unique guest names stored in the set in alphabetical order.

- Count: Display the total number of unique guests attending the party.

Testing: Write a program to test the functionality of your set implementation. Add multiple guest names, including duplicates, and ensure that only unique names are stored in the set. Display the final list of unique guest names and the total count of guests attending the party. As you implement the program, consider using appropriate data types, input validation, error handling, and clear user prompts.

SOURCE CODE:

```
#include <iostream>

#include <set>
#include <string>
#include <algorithm>

using namespace std;

void displayMenu() {
    cout << "Guest List Management System\n";
    cout << "1. Add a guest\n";
    cout << "2. Display guest list\n";
    cout << "3. Display total number of unique guests\n";
    cout << "4. Exit\n";
    cout << "Enter your choice: ";
}

void addGuest(set<string>& guests) {
    string name;
    cout << "Enter guest name: ";
    getline(cin, name);
    guests.insert(name);   //sorting names in alphabatical order
}

void displayGuests(const set<string>& guests) {
    if (guests.empty()) {
        cout << "No guests have been added yet.\n";
```

```
    } else {
        cout << "Unique guest names in alphabetical order:\n";
        for (const auto& name : guests) {
            cout << name << "\n";
        }
    }
}

void displayGuestCount(const set<string>& guests) {
    cout << "Total number of unique guests: " << guests.size() << "\n";
}

int main() {
    set<string> guests;
    int choice;

    do {
        displayMenu();
        cin >> choice;
        cin.get();

        switch (choice) {
            case 1:
                addGuest(guests);
                break;
            case 2:
                displayGuests(guests);
                break;
            case 3:
                displayGuestCount(guests);
                break;
            case 4:
                cout << "Exiting the program.\n";
                break;
            default:
                cout << "Invalid choice. Please try again.\n";
                break;
        }
    } while (choice != 4);

    return 0;
}
```

OUTPUT:

```
1. Add a guest
2. Display guest list
3. Display total number of unique guests
4. Exit
Enter your choice: 1
Enter guest name: Aliza
Guest List Management System
1. Add a guest
2. Display guest list
3. Display total number of unique guests
4. Exit
Enter your choice: 1
Enter guest name: Rania
Guest List Management System
1. Add a guest
2. Display guest list
3. Display total number of unique guests
4. Exit
Enter your choice: 2
Unique guest names in alphabetical order:
Aliza
Rania
Guest List Management System
1. Add a guest
2. Display guest list
3. Display total number of unique guests
4. Exit
Enter your choice: 3
Total number of unique guests: 2
Guest List Management System
1. Add a guest
2. Display guest list
3. Display total number of unique guests
4. Exit
Enter your choice: 4
Exiting the program.
PS C:\Users\DC\Desktop\vs code\C++>
```