

**Name:Barira khalid**

**Batch: 10 june-10 july**

**linkedIn:**

[www.linkedin.com/in/barira-khalid-77a908276](https://www.linkedin.com/in/barira-khalid-77a908276)

**Github :**

I don't have github id

## **TASK 1: CGPA Calculator (C++)**

**CODE:**

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
    cout << "Enter number of courses: ";
    cin >> n;

    vector<float> grades(n);
    vector<int> credits(n);
    float totalPoints = 0;
    int totalCredits = 0;

    for (int i = 0; i < n; i++) {
        cout << "Enter grade points (e.g. 4.0, 3.7) for course " << i + 1 << ": ";
        cin >> grades[i];
        cout << "Enter credit hours for course " << i + 1 << ": ";
        cin >> credits[i];
```

```
        totalPoints += grades[i] * credits[i];
        totalCredits += credits[i];
    }

    float gpa = totalPoints / totalCredits;
    cout << "\nSemester GPA: " << gpa << endl;

    // Extend with multiple semesters for CGPA
    return 0;
}
```

## OUTPUT:

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     int n;
7     cout << "Enter number of courses: ";
8     cin >> n;
9
10    vector<float> grades(n);
11    vector<int> credits(n);
12    float totalPoints = 0;
13    int totalCredits = 0;
14
15    for (int i = 0; i < n; i++) {
16        cout << "Enter grade points (e.g. 4.0, 3.7) for course " << i + 1 << ": ";
17        cin >> grades[i];
18        cout << "Enter credit hours for course " << i + 1 << ": ";
19        cin >> credits[i];
20        totalPoints += grades[i] * credits[i];
21        totalCredits += credits[i];
22    }
23
24    float gpa = totalPoints / totalCredits;
25    cout << "\nSemester GPA: " << gpa << endl;
26
27    // Extend with multiple semesters for CGPA
28    return 0;
29 }
```

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6      int n;
7      cout << "Enter number of courses: ";
8      cin >> n;
9
10     vector<float> grades(n);
11     vector<int> credits(n);

```

```

Enter number of courses: 8
Enter grade points (e.g. 4.0, 3.7) for course 1: 4
Enter credit hours for course 1: 3
Enter grade points (e.g. 4.0, 3.7) for course 2:
3.77
Enter credit hours for course 2: 3
Enter grade points (e.g. 4.0, 3.7) for course 3: 2.777
Enter credit hours for course 3: 3
Enter grade points (e.g. 4.0, 3.7) for course 4: 2.33
Enter credit hours for course 4: 3
Enter grade points (e.g. 4.0, 3.7) for course 5: 2
Enter credit hours for course 5: 3
Enter grade points (e.g. 4.0, 3.7) for course 6: 4
Enter credit hours for course 6: 1
Enter grade points (e.g. 4.0, 3.7) for course 7: 3.77
Enter credit hours for course 7: 1
Enter grade points (e.g. 4.0, 3.7) for course 8: 2.77
Enter credit hours for course 8: 1

Semester GPA: 3.06506

```

## TASK 2: Login and Registration System

### CODE:

```

#include <iostream>
#include <fstream>
#include <string>
using namespace std;

const string USER_FILE = "users.txt";

```

```

bool isUsernameTaken(const string& username) {
    ifstream file(USER_FILE);
    string user, pass;
    while (file >> user >> pass) {
        if (user == username) {
            return true;
        }
    }
    return false;
}

void registerUser() {
    string username, password;

    cout << "\n[Register]\nEnter username: ";
    cin.ignore(); // CLEAR input buffer before getline
    getline(cin, username);

    if (isUsernameTaken(username)) {
        cout << "Username already exists. Try a different one.\n";
        return;
    }

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

    ofstream file(USER_FILE, ios::app);
    file << username << " " << password << endl;
    file.close();

    cout << "Registration successful!\n";
}

void loginUser() {
    string username, password, user, pass;

    cout << "\n[Login]\nEnter username: ";
    cin.ignore(); // CLEAR input buffer before getline
    getline(cin, username);

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

```

```
ifstream file(USER_FILE);
bool found = false;

while (file >> user >> pass) {
    if (user == username && pass == password) {
        found = true;
        break;
    }
}

if (found)
    cout << " Login successful!\n";
else
    cout << " Invalid username or password.\n";
}

int main() {
    int choice;
    do {
        cout << "\n=== MENU ===\n";
        cout << "1. Register\n";
        cout << "2. Login\n";
        cout << "3. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1: registerUser(); break;
            case 2: loginUser(); break;
            case 3: cout << "Exiting...\n"; break;
            default: cout << "Invalid option. Try again.\n";
        }
    } while (choice != 3);

    return 0;
}
```

**OUTPUT:**

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 using namespace std;
5
6 const string USER_FILE = "users.txt";
7
8 bool isUsernameTaken(const string& username) {
9     ifstream file(USER_FILE);
10    string user, pass;
11    while (file >> user >> pass) {
12        if (user == username) {
13            return true;
14        }
15    }
16    return false;
17 }
18
19 void registerUser() {
20    string username, password;
21
22    cout << "\n[Register]\nEnter username: ";
23    cin.ignore(); // CLEAR input buffer before getline
24    getline(cin, username);
25
26    if (isUsernameTaken(username)) {
27        cout << "Username already exists. Try a different one.\n";
28        return;
29    }
30    cout << "Enter password: ";
31    string pass;
32    while (pass.empty()) {
33        pass = getpass("Password: ");
34        if (pass.empty()) {
35            cout << "Password cannot be empty.\n";
36            continue;
37        }
38        if (pass.length() < 6) {
39            cout << "Password must be at least 6 characters long.\n";
40            continue;
41        }
42        break;
43    }
44    cout << "Confirm password: ";
45    string confirm;
46    while (confirm.empty()) {
47        confirm = getpass("Confirm Password: ");
48        if (confirm.empty()) {
49            cout << "Confirm password cannot be empty.\n";
50            continue;
51        }
49        if (confirm.length() < 6) {
52            cout << "Confirm password must be at least 6 characters long.\n";
53            continue;
54        }
55        if (confirm != pass) {
56            cout << "Passwords do not match. Try again.\n";
57            continue;
58        }
59        break;
60    }
61    file.open(USER_FILE, ios::app);
62    file << user << "\n" << pass << "\n";
63    file.close();
64    cout << "User registered successfully.\n";
65 }
```

```
2. Login
3. Exit
Enter choice: 3
Exiting...
```

input

1 2 3 4 5 6 7 8 9 10 11 12



```
[Register]
Enter username: barira
Enter password: 123
✓Registration successful!
```

```
=== MENU ===
1. Register
2. Login
3. Exit
Enter choice: 1
```

```
[Register]
Enter username: ullu
Enter password: 345
✓Registration successful!
```

```
=== MENU ===
1. Register
2. Login
3. Exit
Enter choice: 2
```

```
[Login]
Enter username: ullu
Enter password: 345
✓Login successful!
```

```
=== MENU ===
1. Register
2. Login
3. Exit
Enter choice: 3
Exiting...
```

```
...Program finished with exit code 0
Press ENTER to exit console.□
```

main.cpp    uzma.txt    ⋮    barira.txt    ⋮    users.txt    ⋮

```
1  barira 123
2  ullu 345
3
```

## **TASK 3: Sudoku Solver**

## CODE:

```
#include <iostream>
using namespace std;

const int N = 9;

bool isSafe(int grid[N][N], int row, int col, int num) {
    for (int x = 0; x < 9; x++)
        if (grid[row][x] == num || grid[x][col] == num)
            return false;

    int startRow = row - row % 3, startCol = col - col % 3;
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            if (grid[i + startRow][j + startCol] == num)
                return false;

    return true;
}

bool solveSudoku(int grid[N][N], int row, int col) {
    if (row == N - 1 && col == N)
        return true;
    if (col == N) {
        row++;
        col = 0;
    }
    if (grid[row][col] != 0)
        return solveSudoku(grid, row, col + 1);

    for (int num = 1; num <= 9; num++) {
        if (isSafe(grid, row, col, num)) {
            grid[row][col] = num;
            if (solveSudoku(grid, row, col + 1))
                return true;
            grid[row][col] = 0;
        }
    }
    return false;
}

void printGrid(int grid[N][N]) {
    cout << "\nSolved Sudoku:\n";
```

```

    for (int r = 0; r < N; r++) {
        for (int d = 0; d < N; d++) {
            cout << grid[r][d] << " ";
        }
        cout << endl;
    }
}

int main() {
    int grid[N][N];

    cout << "Enter the Sudoku puzzle (use 0 for empty cells):\n";
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            cin >> grid[i][j];
            if (grid[i][j] < 0 || grid[i][j] > 9) {
                cout << "Invalid input. Please enter numbers between 0 and 9 only.\n";
                return 1;
            }
        }
    }

    if (solveSudoku(grid, 0, 0))
        printGrid(grid);
    else
        cout << "No solution exists for the given puzzle.\n";

    return 0;
}

```

**OUTPUT:**

```

1 #include <iostream>
2 using namespace std;
3
4 const int N = 9;
5
6 bool isSafe(int grid[N][N], int row, int col, int num) {
7     for (int x = 0; x < 9; x++)
8         if (grid[row][x] == num || grid[x][col] == num)
9             return false;
10
11     int startRow = row - row % 3, startCol = col - col % 3;
12     for (int i = 0; i < 3; i++)
13         for (int j = 0; j < 3; j++)
14             if (grid[i + startRow][j + startCol] == num)
15                 return false;
16
17     return true;
18 }
19
20 bool solveSudoku(int grid[N][N], int row, int col) {
21     if (row == N - 1 && col == N)
22         return true;
23     if (col == N) {
24         row++;
25         col = 0;
26     }
27     if (grid[row][col] != 0)
28         return solveSudoku(grid, row, col + 1);
29
30     for (int num = 1; num <= 9; num++)
31         if (isSafe(grid, row, col, num)) {
32             grid[row][col] = num;
33             if (solveSudoku(grid, row, col + 1))
34                 return true;
35             grid[row][col] = 0;
36         }
37     return false;
38 }

```



input

```

8 1 4 2 5 3 7 6 9
5 9 5 4 1 7 3 8 2

```

Enter the Sudoku puzzle (use 0 for empty cells):

```
0 0 3 0 2 0 6 0 0
9 0 0 3 0 5 0 0 1
0 0 1 8 0 6 4 0 0
0 0 8 1 0 2 9 0 0
7 0 0 0 0 0 0 0 8
0 0 6 7 0 8 2 0 0
0 0 2 6 0 9 5 0 0
8 0 0 2 0 3 0 0 9
0 0 5 0 1 0 3 0 0
```

Solved Sudoku:

```
4 8 3 9 2 1 6 5 7
9 6 7 3 4 5 8 2 1
2 5 1 8 7 6 4 9 3
5 4 8 1 3 2 9 7 6
7 2 9 5 6 4 1 3 8
1 3 6 7 9 8 2 4 5
3 7 2 6 8 9 5 1 4
8 1 4 2 5 3 7 6 9
6 9 5 4 1 7 3 8 2
```

...Program finished with exit code 0

Press ENTER to exit console.

## **TASK 4: Banking System**



## CODE:

```
#include <iostream>
#include <vector>
using namespace std;

class Account {
    string name;
    int accountNo;
    double balance;

public:
    Account(string n, int acc, double bal) {
        name = n;
        accountNo = acc;
        balance = bal;
    }

    void deposit(double amount) {
        balance += amount;
        cout << "Deposited: " << amount << endl;
    }

    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            cout << "Withdrawn: " << amount << endl;
        } else {
            cout << "Insufficient balance.\n";
        }
    }

    void showDetails() {
        cout << "Name: " << name << ", Acc#: " << accountNo << ", Balance: " << balance << endl;
    }
};

int main() {
    Account acc("Barira Khan", 101, 1000);

    int choice;
    double amount;
    do {
        cout << "\n1. Deposit\n2. Withdraw\n3. Show Account\n4. Exit\nChoice: ";
```

```
cin >> choice;
switch (choice) {
    case 1:
        cout << "Amount to deposit: ";
        cin >> amount;
        acc.deposit(amount);
        break;
    case 2:
        cout << "Amount to withdraw: ";
        cin >> amount;
        acc.withdraw(amount);
        break;
    case 3:
        acc.showDetails();
        break;
}
} while (choice != 4);

return 0;
}
```

## OUTPUT:

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 class Account {
6     string name;
7     int accountNo;
8     double balance;
9
10 public:
11     Account(string n, int acc, double bal) {
12         name = n;
13         accountNo = acc;
14         balance = bal;
15     }
16
17     void deposit(double amount) {
18         balance += amount;
19         cout << "Deposited: " << amount << endl;
20     }
21
22     void withdraw(double amount) {
23         if (amount <= balance) {
24             balance -= amount;
25             cout << "Withdrawn: " << amount << endl;
26         } else {
27             cout << "Insufficient balance.\n";
```

1. Deposit
2. Withdraw
3. Show Account
4. Exit

Choice: 1

Amount to deposit: 500

Deposited: 500

1. Deposit
2. Withdraw
3. Show Account
4. Exit

Choice: 3

Name: Barira Khan, Acc#: 101, Balance: 1500

1. Deposit
2. Withdraw
3. Show Account
4. Exit

Choice: 1

Amount to deposit: 2000

Deposited: 2000

1. Deposit
2. Withdraw
3. Show Account
4. Exit

Choice: 3

Name: Barira Khan, Acc#: 101, Balance: 3500

1. Deposit
2. Withdraw
3. Show Account
4. Exit

Choice: 2

Amount to withdraw: 5000

Insufficient balance.

Choice: 3  
Name: Barira Khan, Acc#: 101, Balance: 3500

1. Deposit
2. Withdraw
3. Show Account
4. Exit

Choice: 2  
Amount to withdraw: 5000  
Insufficient balance.

1. Deposit
2. Withdraw
3. Show Account
4. Exit

Choice: 2  
Amount to withdraw: 700  
Withdrawn: 700

1. Deposit
2. Withdraw
3. Show Account
4. Exit

Choice: 3  
Name: Barira Khan, Acc#: 101, Balance: 2800

1. Deposit
2. Withdraw
3. Show Account
4. Exit

Choice: 4

...Program finished with exit code 0  
Press ENTER to exit console.