PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

Class : SY BTech **Acad. Yr. 2025-26** **Semester : I**
Name of the student: Hariom Shrikrishna Gundale **PRN : 124B1B036**
Department: Computer Enginnering **Division : A**
Course Name : Data Structures Laboratory **Course Code: BCE23PC02**
Completion Date : 28/07/2025

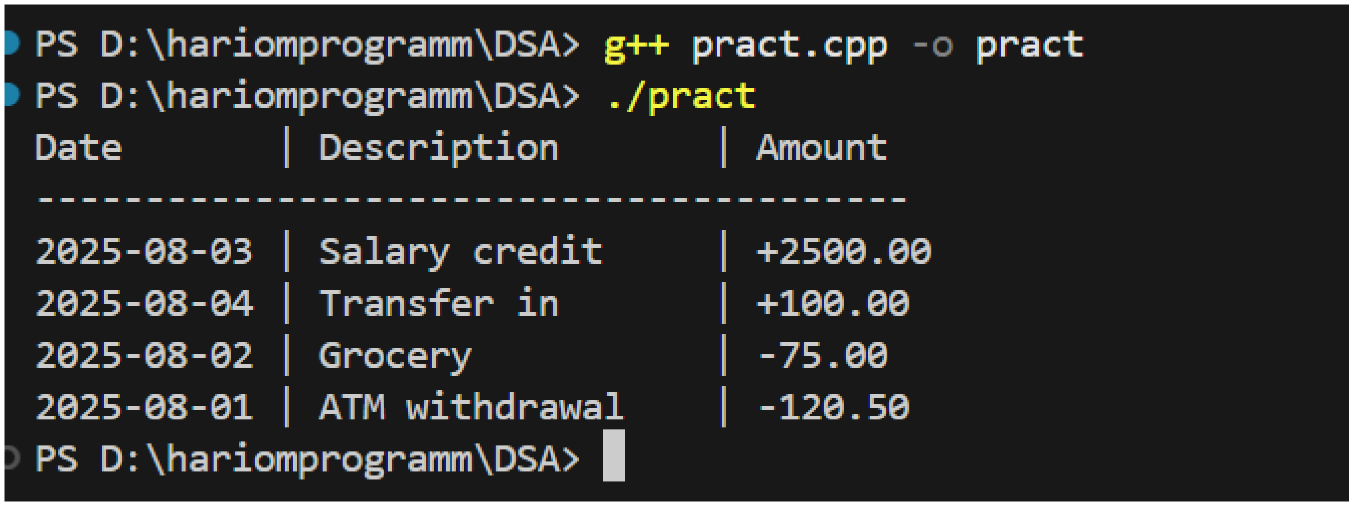
Assignment No. 3

Problem Statement: A banking app needs to display a user's transaction history sorted by transaction amount to quickly identify large deposits or withdrawals. Write a program for above scenario.

Source Code :

```
#include <iostream>
#include <vector>
#include <iomanip>
#include <string>

using namespace std;

class Transaction {
public:
    Transaction(string d, double a, string desc)
        : date(move(d)), amount(a), description(move(desc)) {}

    double getAmount() const { return amount; }
    const string& getDate() const { return date; }
    const string& getDescription() const { return description; }

private:
    string date;
    double amount;
    string description;
};

class TransactionSorter {
public:
    void sortTransactions(vector<Transaction>& txs) {
        if (!txs.empty())
            mergeSort(txs, 0, static_cast<int>(txs.size()) - 1);
    }

private:
    void mergeSort(vector<Transaction>& txs, int left, int right) {
        if (left >= right) return;
        int mid = left + (right - left) / 2;
        mergeSort(txs, left, mid);
        mergeSort(txs, mid + 1, right);
        merge(txs, left, mid, right);
    }

    void merge(vector<Transaction>& txs, int left, int mid, int right) {
        int n1 = mid - left + 1, n2 = right - mid;
        vector<Transaction> L, R;
        L.reserve(n1);
        R.reserve(n2);
        for (int i = 0; i < n1; ++i) L.push_back(txs[left + i]);
        for (int j = 0; j < n2; ++j) R.push_back(txs[mid + 1 + j]);

        int i = 0, j = 0, k = left;
        while (i < n1 && j < n2) {
            if (L[i].getAmount() >= R[j].getAmount())
                txs[k++] = L[i++];
            else
                txs[k++] = R[j++];
        }
        while (i < n1) txs[k++] = L[i++];
        while (j < n2) txs[k++] = R[j++];
    }
};

int main() {
    vector<Transaction> txs = {
        {"2025-08-01", -120.50, "ATM withdrawal"},
        {"2025-08-03", 2500.00, "Salary credit"},
        {"2025-08-02", -75.00, "Grocery"},
        {"2025-08-04", 100.00, "Transfer in"}
    };

    TransactionSorter sorter;
    sorter.sortTransactions(txs);

    cout << fixed << setprecision(2);
    cout << "Date       | Description       | Amount\n";
    cout << "----------------------------------------\n";
    for (const auto& tx : txs) {
        cout << tx.getDate() << " | "
             << left << setw(17) << tx.getDescription() << " | "
             << (tx.getAmount() >= 0 ? "+" : "") << tx.getAmount() << "\n";
    }
    return 0;
}
```

Screen Shot of Output :



```
PS D:\hariomprogramm\DSA> g++ pract.cpp -o pract
PS D:\hariomprogramm\DSA> ./pract
Date       | Description          | Amount
----------------------------------------
2025-08-03 | Salary credit        | +2500.00
2025-08-04 | Transfer in          | +100.00
2025-08-02 | Grocery              | -75.00
2025-08-01 | ATM withdrawal       | -120.50
PS D:\hariomprogramm\DSA> 
```

Conclusion:

Thus, we have successfully implemented the C++ program to manage the banking transaction issue in bank using the merge sort we have sorted traction in decreasing order of amount.Merge sort efficiently sort the transaction.