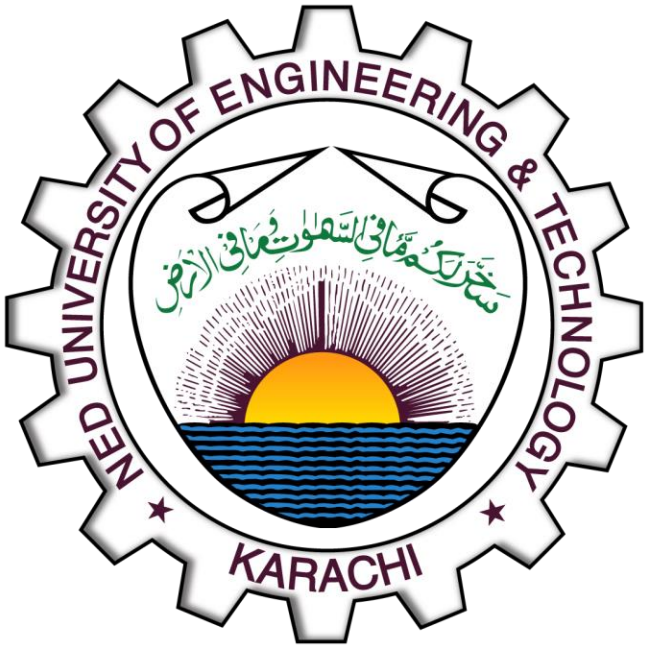
OBJECT ORIENTED PROGRAMMING

[CT-260]



NAME: TAHA AHMED MALLICK

ROLL NO: CT-25183

DEPARTMENT: BCIT **BATCH:** 2025

YEAR & SECTION: FSCS-D

LAB : 4

QUESTION#1:

CODE:

```
#include <iostream>
#include <string.h>
using namespace std;

class Employee
{
    char *EmployeeName;
    const int EmployeeId;

public:
    Employee(int id, const char *name = "N/A") : EmployeeId(id)
    {
        EmployeeName = new char[strlen(name) + 1]();
        strcpy(EmployeeName, name);
    }
    char* getName() { return EmployeeName; }
    void changeName(const char *name)
    {
        if (EmployeeName == name)
            return;
        delete[] EmployeeName;
        EmployeeName = new char[strlen(name) + 1]();
        strcpy(EmployeeName, name);
    }
    int getID() const { return EmployeeId; }
    ~Employee() { delete[] EmployeeName; }
};

int main()
{
    Employee Employee1(1001, "Jhon"), Employee2(1002, "Jhonathan"),
Employee3(1003, "Jhonny");
    cout << "IDs\tNames" << endl;
    cout << Employee1.getID() << "\t" << Employee1.getName() << endl;
    cout << Employee2.getID() << "\t" << Employee2.getName() << endl;
    cout << Employee3.getID() << "\t" << Employee3.getName() << endl;

    Employee2.changeName("Taha");

    cout << "\033[1;33m====After changing Employee2 name====\033[0m" << endl;
```

```
    cout << "IDs\tNames" << endl;
    cout << Employee1.getID() << "\t" << Employee1.getName() << endl;
    cout << Employee2.getID() << "\t" << Employee2.getName() << endl;
    cout << Employee3.getID() << "\t" << Employee3.getName() << endl;
    return 0;
}
```

OUTPUT:

```
IDs     Names
1001    Jhon
1002    Jhonathan
1003    Jhonny
====After changing Employee2 name====
IDs     Names
1001    Jhon
1002    Taha
1003    Jhonny
```

QUESTION#2:

CODE:

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

class DynamicArray
{
    int *arr, arraySize;

public:
    DynamicArray(int size) : arraySize(size) { this->arr = new int[size](); }

    ~DynamicArray() { delete[] arr; }

    void push(int val)
    {
        int *temp = new int[arraySize + 1];
        for (int i = 0; i < arraySize; i++)
            temp[i] = arr[i];
        temp[arraySize++] = val;
        delete[] arr;
        arr = temp;
    }

    void printArray() const
    {
        cout << "Your array: ";
        for (int i = 0; i < arraySize; i++)
            cout << arr[i] << (i == arraySize - 1 ? "" : ", ");
        cout << endl
             << "Total elements: " << arraySize << endl;
    }

    int size() const { return arraySize; }
};

int main()
{
    DynamicArray myArray(3);
    cout << "Initial array: ";
    myArray.printArray();
```

```
    myArray.push(5);
    myArray.push(10);

    cout << "After push operations: ";
    myArray.printArray();

    cout << "Size of array: " << myArray.size() << endl;
    return 0;
}
```

OUTPUT:

```
Initial array: Your array: 0, 0, 0
Total elements: 3
After push operations: Your array: 0, 0, 0, 5, 10
Total elements: 5
Size of array: 5
```

QUESTION#3:

CODE:

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

class Account
{
    int acc_no = 0, acc_bal = 0, security_code = 0;
    static int acc_count;

public:
    Account() { acc_count++; }

    void setAccNo(int acc_no) { this->acc_no = acc_no; }
    void setAccBal(int acc_bal) { this->acc_bal = acc_bal; }
    void setSecurityCode(int security_code) { this->security_code =
security_code; }

    int getAccNo() const { return acc_no; }
    int getAccBal() const { return acc_bal; }
    int getSecurityCode() const { return security_code; }

    static int getAccCount() { return acc_count; }

    ~Account() { acc_count--; }
};

int Account::acc_count = 0;

int main()
{

    cout << "Initial Account Count: " << Account::getAccCount() << endl;


    Account acc1, acc2;
    acc1.setAccNo(1001);
    acc1.setAccBal(5000);
    acc1.setSecurityCode(1234);
    acc2.setAccNo(1002);
    acc2.setAccBal(7500);
```

```
    cout << "\n=== Static Feature ===" << endl;
    cout << "Total Accounts Created: " << Account::getAccCount() << endl;

    cout << "\n=== Encapsulation Feature ===" << endl;
    cout << "Acc 1 [No: " << acc1.getAccNo() << " | Bal: " <<
acc1.getAccBal() << "]" << endl;
    cout << "Acc 2 [No: " << acc2.getAccNo() << " | Bal: " <<
acc2.getAccBal() << "]" << endl;

    cout << "\nAcc 1-> Security Code (via getter): " <<
acc1.getSecurityCode() << endl;

    return 0;
}
```

OUTPUT:

```
Initial Account Count: 0

=== Static Feature ===
Total Accounts Created: 2

=== Encapsulation Feature ===
Acc 1 [No: 1001 | Bal: 5000]
Acc 2 [No: 1002 | Bal: 7500]

Acc 1-> Security Code (via getter): 1234
```

QUESTION#4:

CODE:

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

class SmartDevice
{
    const int id;
    int batteryLevel;

public:
    SmartDevice(int id, int batteryLevel) : id(id), batteryLevel(batteryLevel) {}
    void setBatteryLevel(int level)
    {
        if (level >= 0 && level <= 100)
            batteryLevel = level;
        else
            cout << "Battery level must be between 0 and 100." << endl;
    }
    // Const getter methods which do not modify the state of the object
    int getId() const { return id; }
    int getBatteryLevel() const { return batteryLevel; }
};

int main()
{
    SmartDevice device1(1, 75);
    cout << "Device ID: " << device1.getId() << endl;
    cout << "Battery Level: " << device1.getBatteryLevel() << "%" << endl;
    device1.setBatteryLevel(85);
    cout << "Updated Battery Level: " << device1.getBatteryLevel() << "%" <<
endl;
    return 0;
}
```

OUTPUT:

```
Device ID: 1
Battery Level: 75%
Updated Battery Level: 85%
```

QUESTION#5:

CODE:

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

class Room
{
    int days;
    string customerName;
    static const double dailyRent;

public:
    Room(int days, const string customerName) : days(days),
customerName(customerName) {}
    double calculateRent() const
    {
        if (days > 7)
            return dailyRent * (days - 1);
        else
            return dailyRent * days;
    }
    void displayBill() const
    {
        cout << "\n--- Hotel Mercato Receipt ---" << endl;
        cout << "Customer Name: " << customerName << endl;
        cout << "Days of Stay : " << days << endl;
        cout << "Total Rent   : Rs. " << calculateRent() << endl;
        if (days > 7)
        {
            cout << "Status       : Discount Applied (1 Day Free!)" << endl;
        }
    }
};

const double Room::dailyRent = 1000.85;

int main()
{
    Room room1(10, "John Doe");
    room1.displayBill();
    return 0;
}
```

OUTPUT:

```
--- Hotel Mercato Receipt ---
Customer Name: John Doe
Days of Stay : 10
Total Rent   : Rs. 9007.65
Status       : Discount Applied (1 Day Free!)
```