WINTER DOMAIN CAMP

NAME:Vikram kumar **UID:22BCS12220**

SECTION: KPIT-901-B **DATE: 19/12/2024**

Q1. Sum of Natural Numbers up to N

Sol.

```cpp
#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    int sum_natural_numbers = n * (n + 1) / 2;
    cout << sum_natural_numbers << endl;
    return 0;
}
```

OUTPUT:

```
100
5050
```

Q2. Count Digits in a Number.

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

int main() {
    int n;
    cin >> n;
    int count = 0;
    while (n > 0) {
        n /= 10;
        count++;
    }
    cout << count << endl;
    return 0;
}
```

OUTPUT:

```
900000
6
```

Q3. Function Overloading for Calculating Area.

Sol.

```cpp
#include <iostream>
using namespace std;
double area(double radius) {
    const double PI = 3.14159;
    return PI * radius * radius;
}
int area(int length, int breadth) {
    return length * breadth;
}
double area(double base, double height) {
    return 0.5 * base * height;
}
int main() {
    double radius;
    cout << "Enter radius of circle: ";
    cin >> radius;
    int length, breadth;
    cout << "Enter length and breadth of rectangle: ";
    cin >> length >> breadth;
    double base, height;
    cout << "Enter base and height of triangle: ";
    cin >> base >> height;
    cout << "Area of circle: " << area(radius) << endl;
    cout << "Area of rectangle: " << area(length, breadth) << endl;
    cout << "Area of triangle: " << area(base, height) << endl;
    return 0;
}
```

```
Enter radius of circle: 4
Enter length and breadth of rectangle: 3
4
Enter base and height of triangle: 6
8
Area of circle: 50.2654
Area of rectangle: 12
Area of triangle: 24


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

Q4. **Implement Polymorphism for Banking Transactions**

Sol.

```cpp
#include <iostream>
using namespace std;
class Account {
protected:
    double balance;
public:
    Account(double bal) : balance(bal) {}
    virtual void calculateInterest() = 0;
    virtual ~Account() {}
};
class SavingsAccount : public Account {
    double rate;
    int time;
public:
    SavingsAccount(double bal, double r, int t) : Account(bal), rate(r), time(t) {}
    void calculateInterest() override {
        double interest = (balance * rate * time) / 100;
        cout << "Savings Account Interest: " << interest << endl;
    }
};
class CurrentAccount : public Account {
    double maintenanceFee;
public:
    CurrentAccount(double bal, double fee) : Account(bal), maintenanceFee(fee) {}
    void calculateInterest() override {
        balance -= maintenanceFee;
        cout << "Balance after fee deduction: " << balance << endl;
    }
};

int main() {
    int accountType;
```

```
    cout << "Enter Account Type (1 for Savings, 2 for Current): ";
    cin >> accountType;

    if (accountType == 1) {
        // Savings Account
        double balance, rate;
        int time;
        cout << "Enter Balance: ";
        cin >> balance;
        cout << "Enter Interest Rate (in %): ";
        cin >> rate;
        cout << "Enter Time (in years): ";
        cin >> time;

        if (balance < 1000 || rate < 1 || rate > 15 || time < 1 || time > 10) {
            cout << "Invalid input values." << endl;
            return 1;
        }

        SavingsAccount sa(balance, rate, time);
        sa.calculateInterest();

    } else if (accountType == 2) {
        // Current Account
        double balance, fee;
        cout << "Enter Balance: ";
        cin >> balance;
        cout << "Enter Monthly Maintenance Fee: ";
        cin >> fee;

        if (balance < 1000 || fee < 50 || fee > 500) {
            cout << "Invalid input values." << endl;
```

```
            cout << "Invalid input values." << endl;
            return 1;
        }

        CurrentAccount ca(balance, fee);
        ca.calculateInterest();

    } else {
        cout << "Invalid account type." << endl;
    }

    return 0;
}
```

OUTPUT:

```
Enter Account Type (1 for Savings, 2 for Current): 1
Enter Balance: 10000
Enter Interest Rate (in %): 5
Enter Time (in years): 3
Savings Account Interest: 1500


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

Q5. Hierarchical Inheritance for Employee Management System.

Sol.

```cpp
#include <iostream>
#include <string>
using namespace std;
class Employee {
protected:
    string name;
    int id;
    int salary;
public:
    Employee(string n, int i, int s) : name(n), id(i), salary(s) {}
    virtual void displayEarnings() = 0;
    virtual ~Employee() {}
};
class Manager : public Employee {
    int rating;

public:
    Manager(string n, int i, int s, int r) : Employee(n, i, s), rating(r) {}
    void displayEarnings() override {
        int bonus = (rating * 10 * salary) / 100;
        cout << "Employee: " << name << " (ID: " << id << ")" << endl;
        cout << "Role: Manager" << endl;
        cout << "Base Salary: " << salary << endl;
        cout << "Bonus: " << bonus << endl;
        cout << "Total Earnings: " << (salary + bonus) << endl;
    }
};
class Developer : public Employee {
    int extraHours;

public:
    Developer(string n, int i, int s, int h) : Employee(n, i, s), extraHours(h) {}
```

```
    Developer(string n, int i, int s, int h) : Employee(n, i, s), extraHours(h) {}
    void displayEarnings() override {
        int overtimeCompensation = extraHours * 500;
        cout << "Employee: " << name << " (ID: " << id << ")" << endl;
        cout << "Role: Developer" << endl;
        cout << "Base Salary: " << salary << endl;
        cout << "Overtime Compensation: " << overtimeCompensation << endl;
        cout << "Total Earnings: " << (salary + overtimeCompensation) << endl;
    }
};
int main() {
    int empType;
    cout << "Enter Employee Type (1 for Manager, 2 for Developer): ";
    cin >> empType;
    if (empType == 1) {
        string name;
        int id, salary, rating;
        cout << "Enter Name: ";
        cin >> name;
        cout << "Enter ID: ";
        cin >> id;
        cout << "Enter Salary: ";
        cin >> salary;
        cout << "Enter Performance Rating (1-5): ";
        cin >> rating;

        if (salary < 10000 || salary > 1000000 || rating < 1 || rating > 5) {
            cout << "Invalid input values." << endl;
            return 1;
        }
        Manager manager(name, id, salary, rating);
        manager.displayEarnings();
```

```
            return 1;
        }
        Manager manager(name, id, salary, rating);
        manager.displayEarnings();
    } else if (empType == 2) {
        string name;
        int id, salary, extraHours;
        cout << "Enter Name: ";
        cin >> name;
        cout << "Enter ID: ";
        cin >> id;
        cout << "Enter Salary: ";
        cin >> salary;
        cout << "Enter Extra Hours Worked: ";
        cin >> extraHours;
        if (salary < 10000 || salary > 1000000 || extraHours < 0 || extraHours > 100)
            cout << "Invalid input values." << endl;
            return 1;}
        Developer developer(name, id, salary, extraHours);
        developer.displayEarnings();
    } else {
        cout << "Invalid employee type." << endl;
    }
    return 0;
}
```

OUTPUT:

```
Enter Employee Type (1 for Manager, 2 for Developer): 1
Enter Name: Alice
Enter ID: 101
Enter Salary: 50000
Enter Performance Rating (1-5): 4
Employee: Alice (ID: 101)
Role: Manager
Base Salary: 50000
Bonus: 20000
Total Earnings: 70000
```