

Name: Sahil

UID: 22BCS10928

SECTION: 901_KPIT(B)

1) Sum of Natural Numbers up to N

Calculate the sum of all natural numbers from 1 to n, where n is a positive integer. Use the formula:

$$\text{Sum} = n \times (n+1) / 2 .$$

Take n as input and output the sum of natural numbers from 1 to n .

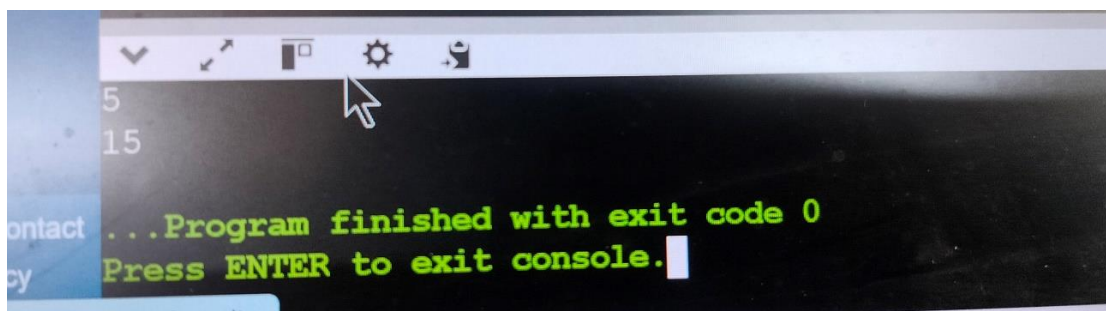
Answer:

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

```
int main() {
    int n;
    int sum = 0;
    cin >> n;
    if (n < 0) {
        cout << "is not a natural number";
    } else {
        sum = n * (n + 1) / 2;
        cout << sum;
    }

    return 0;
}
```

OUTpUT:



2) Count Digits in a Number

Objective

Count the total number of digits in a given number n . The number can be a positive integer. For example, for the number 12345, the count of digits is 5. For a number like 900000, the count of digits is 6.

Given an integer n , your task is to determine how many digits are present in n . This task will help you practice working with loops, number manipulation, and conditional logic.

Answer:

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

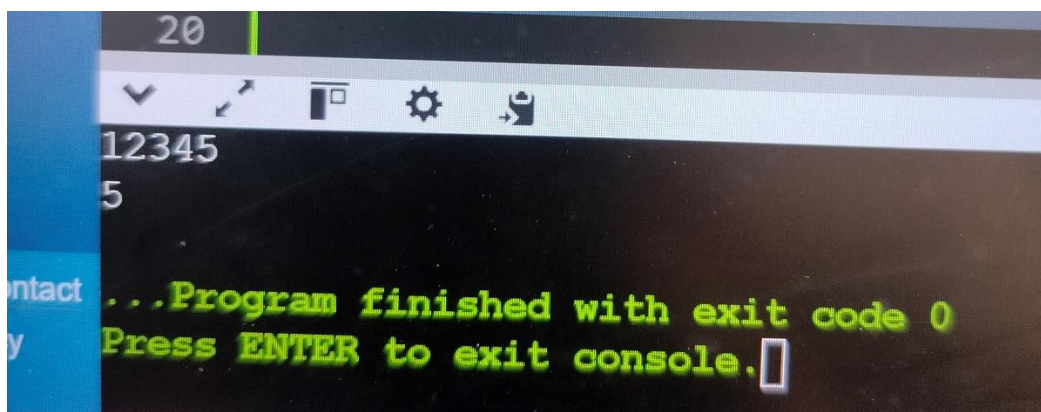
```
int main() {
    int n, count = 0;
    cin >> n;

    if (n < 0) {
        n = -n;
    }

    do {
        count++;
        n /= 10;
    } while (n > 0);

    cout << count;
    return 0;
}
```

OUTPUT:



3) Function Overloading for Calculating Area.

Objective

Write a program to calculate the area of different shapes using function overloading. Implement overloaded functions to compute the area of a circle, a rectangle, and a triangle.

Answer:

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

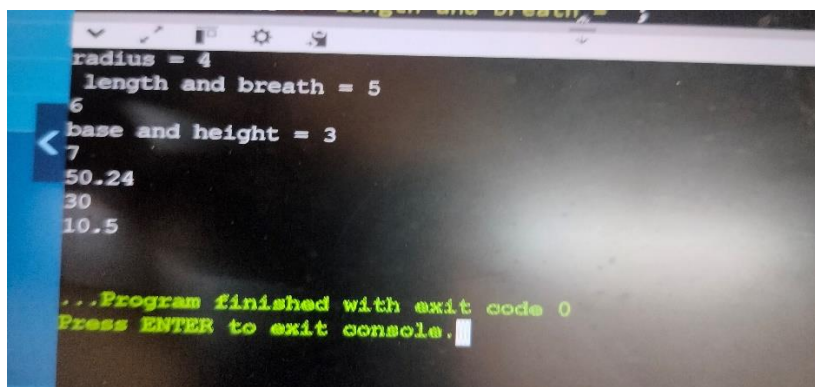
double area(double r) {
    return 3.14 * r * r;
}

int area(int l, int b) {
    return l * b;
}

double area(double base, double height) {
    return 0.5 * base * height;
}

int main() {
    double r, base, height;
    int l, b;
    cout<<"radius = ";
    cin >> r;
    cout<<" length and breath = ";
    cin >> l >> b;
    cout<<"base and height = ";
    cin >> base >> height;
    cout << area(r) << endl;
    cout << area(l, b) << endl;
    cout << area(base, height) << endl;

    return 0;
}
```



4) Hierarchical Inheritance for Employee Management System

Objective

Create a C++ program to simulate an employee management system using hierarchical inheritance. Design a base class Employee that stores basic details (name, ID, and salary).

Create two derived classes:

Manager: Add and calculate bonuses based on performance ratings.

Developer: Add and calculate overtime compensation based on extra hours worked.

The program should allow input for both types of employees and display their total earnings.

Answer:

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

class Employee {
protected:
    string name;
    int id;
    double salary;

public:
    void setDetails(string n, int i, double s) {
        name = n;
        id = i;
        salary = s;
    }

    virtual void displayDetails() {
        cout << "Name: " << name << endl;
        cout << "ID: " << id << endl;
        cout << "Base Salary: $" << salary << endl;
    }
};

class Manager : public Employee {
private:
    double bonus;

public:
    void setBonus(double performanceRating) {

        bonus = salary * (performanceRating / 100.0);
    }

    void displayDetails() override {
        Employee::displayDetails();
        cout << "Bonus: " << bonus << endl;
        cout << "Total Earnings: " << salary + bonus << endl;
    }
};
```

```

    }
};

class Developer : public Employee {
private:
    double overtimePay;
    double hourlyRate;

public:
    void setOvertime(int extraHours) {
        hourlyRate = salary / 160;
        overtimePay = hourlyRate * extraHours;
    }

    void displayDetails() override {
        Employee::displayDetails();
        cout << "Overtime Pay: " << overtimePay << endl;
        cout << "Total Earnings: " << salary + overtimePay << endl;
    }
};

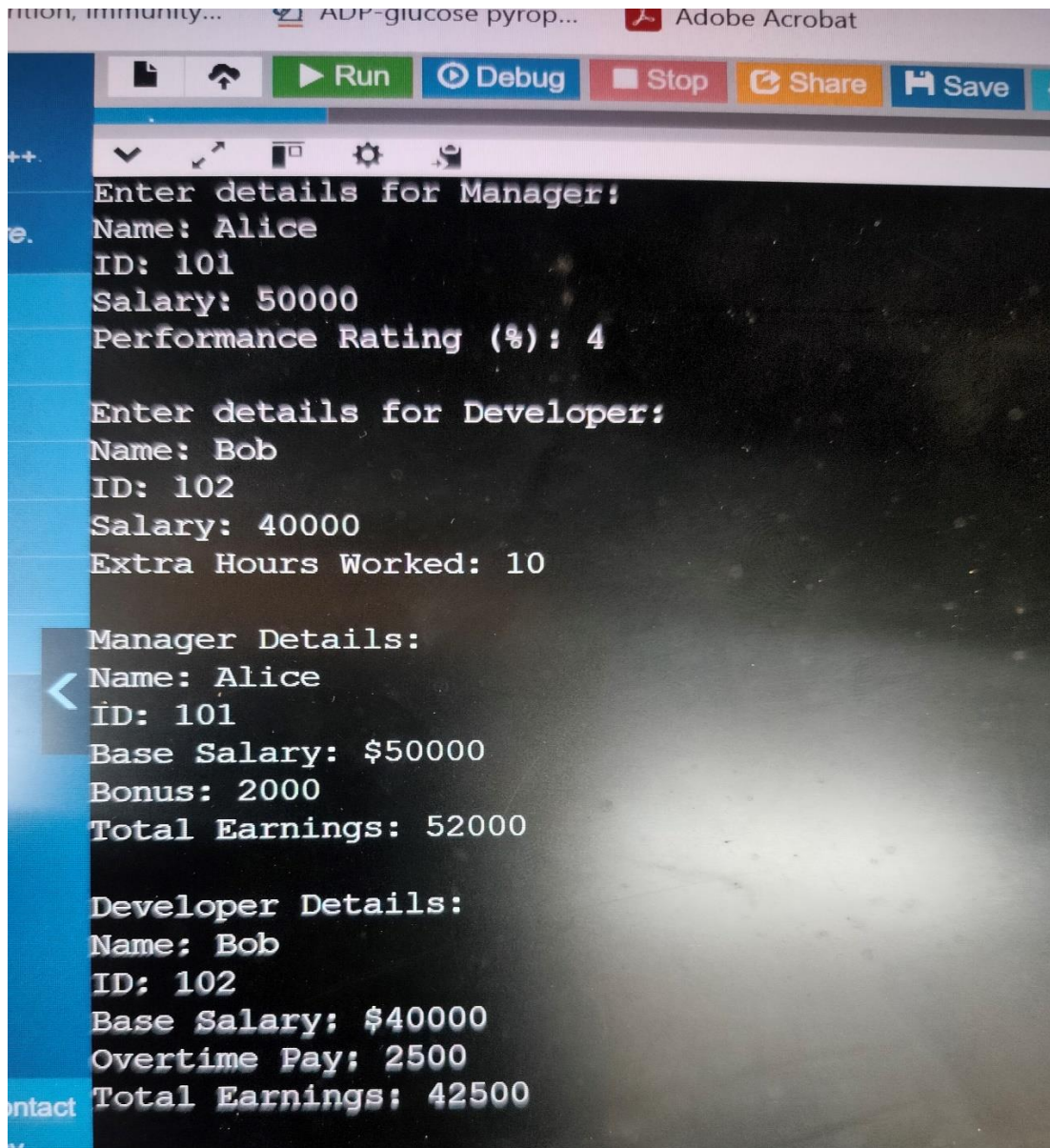
int main() {
    Manager m;
    Developer d;
    cout << "Enter details for Manager:\n";
    string name;
    int id;
    double salary, performanceRating;
    cout << "Name: ";
    cin >> name;
    cout << "ID: ";
    cin >> id;
    cout << "Salary: ";
    cin >> salary;
    m.setDetails(name, id, salary);
    cout << "Performance Rating (%): ";
    cin >> performanceRating;
    m.setBonus(performanceRating);
    cout << "\nEnter details for Developer:\n";
    int extraHours;
    cout << "Name: ";
    cin >> name;
    cout << "ID: ";
    cin >> id;
    cout << "Salary: ";
    cin >> salary;
    d.setDetails(name, id, salary);
    cout << "Extra Hours Worked: ";
    cin >> extraHours;
    d.setOvertime(extraHours);
    cout << "\nManager Details:\n";
    m.displayDetails();
}

```

```

cout << "\nDeveloper Details:\n";
d.displayDetails();
return 0;
}

```



5) Multi-Level Inheritance for Vehicle Simulation

Objective

Create a C++ program to simulate a vehicle hierarchy using multi-level inheritance. Design a base class Vehicle that stores basic details (brand, model, and mileage). Extend it into the Car class to add attributes like fuel efficiency and speed. Further extend it into ElectricCar to include battery capacity and charging time. Implement methods to calculate:

Fuel Efficiency: Miles per gallon (for Car).

Range: Total distance the electric car can travel with a full charge.

Answer:

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

class Vehicle {
protected:
    string brand, model;
    double mileage;

public:
    void setVehicleDetails(string b, string m, double mi) {
        brand = b;
        model = m;
        mileage = mi;
    }

    void displayVehicleDetails() {
        cout << "Vehicle: " << brand << " " << model << endl;
        cout << "Mileage: " << mileage << " miles" << endl;
    }
};

class Car : public Vehicle {
private:
    double fuelUsed, distance, fuelEfficiency;

public:
    void setCarDetails(double fuel, double dist) {
        fuelUsed = fuel;
        distance = dist;
        fuelEfficiency = distance / fuelUsed;
    }

    void displayCarDetails() {
        displayVehicleDetails();
        cout << "Fuel Efficiency: " << fuelEfficiency << " miles/gallon" << endl;
    }
};

int main() {
    int vehicleType;
    cin >> vehicleType;

    if (vehicleType == 1) {
        Car car;
        string brand, model;
        double mileage, fuel, distance;

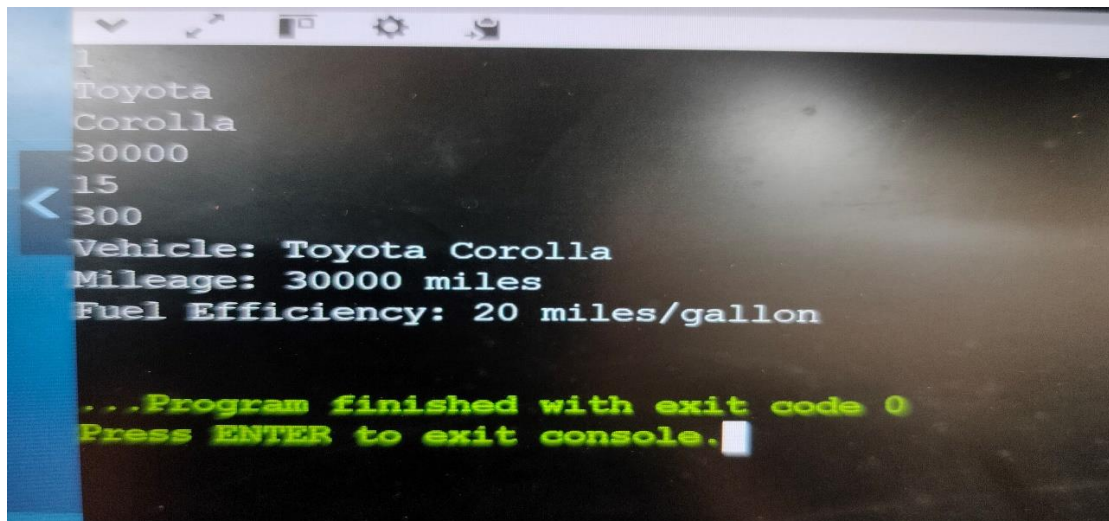
        cin >> brand >> model >> mileage;
        car.setVehicleDetails(brand, model, mileage);
    }
}
```



```
    cin >> fuel >> distance;
    car.setCarDetails(fuel, distance);

    car.displayCarDetails();
} else {
    cout << "Invalid choice\n";
}

return 0;
}
```

A screenshot of a Windows command prompt window showing the execution of a C++ program. The program prompts for car details: make, model, mileage, fuel efficiency, and distance. The user has entered 'Toyota', 'Corolla', '30000', '15', and '300'. The program then displays the car's details: 'Vehicle: Toyota Corolla', 'Mileage: 30000 miles', and 'Fuel Efficiency: 20 miles/gallon'. Finally, it shows a green message: '...Program finished with exit code 0' and 'Press ENTER to exit console.' with a cursor at the end.

```
1
Toyota
Corolla
30000
15
300
Vehicle: Toyota Corolla
Mileage: 30000 miles
Fuel Efficiency: 20 miles/gallon

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