DOMAIN WINTER WINNING CAMP

DAY: 1

Submitted by: Agrima Sharma

UID: 22BCS15314

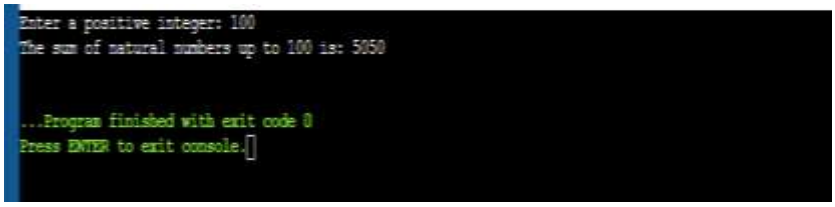
Section: 620/A

1) **Sum of Natural Numbers up to N**

Code:

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

int main() {
   int n;
   cout << "Enter a positive integer: ";
   cin >> n;

   int sum = (n * (n + 1)) / 2;

   cout << "The sum of natural numbers up to " << n << " is: " << sum << endl;

   return 0;
}
```

Output:



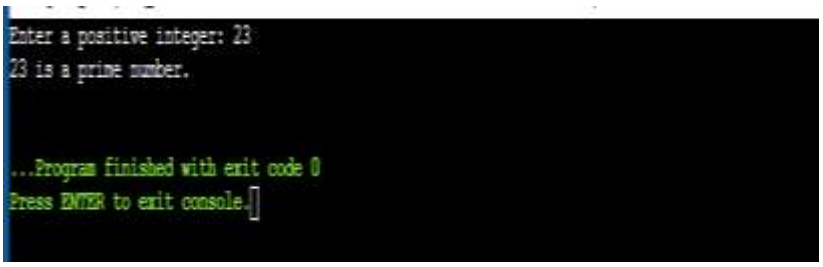
2) **Check if a Number is Prime**

Code:

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

bool isPrime(int n) {
   if (n <= 1) return false;
   for (int i = 2; i * i <= n; i++) {
```

```
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}

int main() {
    int num;
    cout << "Enter a positive integer: ";
    cin >> num;

    if (isPrime(num)) {
        cout << num << " is a prime number." << endl;
    } else {
        cout << num << " is not a prime number." << endl;
    }

    return 0;
}
```

Output:



3) Count Digits in a Number

Code:

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

int countDigits(int n) {
    int count = 0;
        while (n > 0) {
        n /= 10;
        count++;
```

```
    }
    return count;
}

int main() {
    int num;
    cout << "Enter an integer: ";
    cin >> num;

    int digitCount = countDigits(num);
    cout << "The number " << num << " has " << digitCount << " digit(s)." << endl;

    return 0;
}
```

Output:

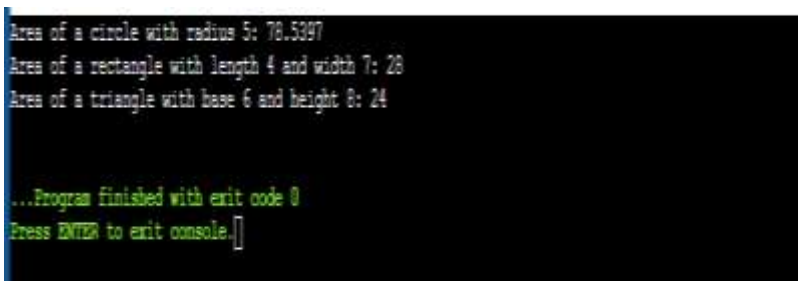
4) **Function Overloading for Calculating Area.**

Code:

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

// Function to calculate the area of a circle
double area(double radius) {
    return 3.14159 * radius * radius;
}

// Function to calculate the area of a rectangle
double area(double length, double width) {
    return length * width;
}
```

```
// Function to calculate the area of a triangle
double area(double base, double height, bool isTriangle) {
   return 0.5 * base * height;
}

int main() {
   cout << "Area of a circle with radius 5: " << area(5) << endl;
   cout << "Area of a rectangle with length 4 and width 7: " << area(4, 7) << endl;
   cout << "Area of a triangle with base 6 and height 8: " << area(6, 8, true) << endl;

   return 0;
}
```

Output:



5) **Matrix Multiplication Using Function Overloading**

Code:

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

// Function to multiply two square matrices (n x n)
void multiplyMatrix(int n, int A[][10], int B[][10], int result[][10]) {
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         result[i][j] = 0;
         for (int k = 0; k < n; k++) {
            result[i][j] += A[i][k] * B[k][j];
         }
      }
   }
}

// Function to multiply two rectangular matrices (rowsA x colsA and rowsB x colsB)
```

```
void multiplyMatrix(int rowsA, int colsA, int A[][10], int rowsB, int colsB, int B[][10], int result[][10]) {
    if (colsA != rowsB) {
        cout << "Matrix multiplication not possible. Column count of A must equal row count of B." << endl;
        return;
    }

    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsB; j++) {
            result[i][j] = 0;
            for (int k = 0; k < colsA; k++) {
                result[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

void displayMatrix(int rows, int cols, int matrix[][10]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}

int main() {
    int A[10][10], B[10][10], result[10][10];
    int rowsA, colsA, rowsB, colsB;

    cout << "Enter rows and columns for matrix A: ";
    cin >> rowsA >> colsA;
    cout << "Enter elements of matrix A:" << endl;
    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsA; j++) {
            cin >> A[i][j];
        }
```

```
    }

    cout << "Enter rows and columns for matrix B: ";
    cin >> rowsB >> colsB;
    cout << "Enter elements of matrix B:" << endl;
    for (int i = 0; i < rowsB; i++) {
        for (int j = 0; j < colsB; j++) {
            cin >> B[i][j];
        }
    }

    cout << "Result of matrix multiplication:" << endl;

    if (rowsA == colsA && rowsB == colsB && rowsA == rowsB) {
        multiplyMatrix(rowsA, A, B, result);
        displayMatrix(rowsA, colsA, result);
    } else {
        multiplyMatrix(rowsA, colsA, A, rowsB, colsB, B, result);
        displayMatrix(rowsA, colsB, result);
    }

    return 0;
}
```

Output:

```
Enter rows and columns for matrix A: 2
2
Enter elements of matrix A:
1
2
3
4
Enter rows and columns for matrix B: 2
2
Enter elements of matrix B:
5
6
7
8
Result of matrix multiplication:
19 22
43 50


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

6) Multi-Level Inheritance for Vehicle Simulation

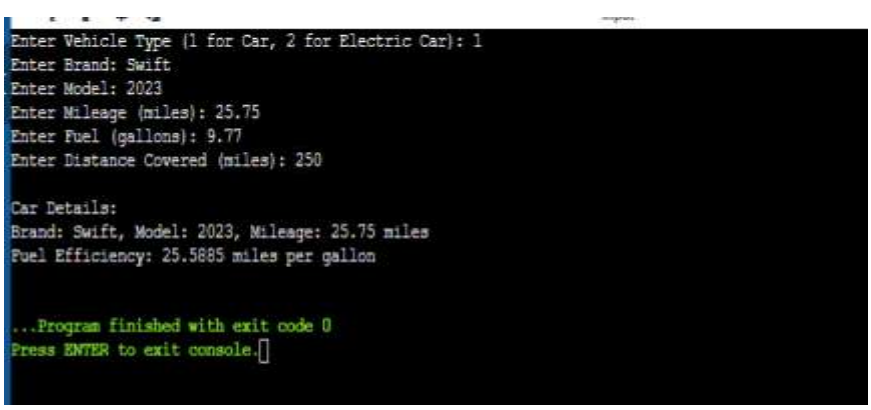
Code:

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

// Base class: Vehicle
class Vehicle {
protected:
   string brand;
   string model;
   double mileage;

public:
   Vehicle(const string &brand, const string &model, double mileage)
      : brand(brand), model(model), mileage(mileage) {}

   void displayDetails() const {
      cout << "Brand: " << brand << ", Model: " << model << ", Mileage: " << mileage << "
miles\n";
   }
};

// Derived class: Car
class Car : public Vehicle {
protected:
   double fuel; // in gallons
   double distanceCovered; // in miles

public:
   Car(const string &brand, const string &model, double mileage, double fuel, double
distanceCovered)
      : Vehicle(brand, model, mileage), fuel(fuel), distanceCovered(distanceCovered) {}

   double calculateFuelEfficiency() const {
      if (fuel == 0) return 0;
      return distanceCovered / fuel; // miles per gallon
   }
```

```
    void displayCarDetails() const {
        displayDetails();
        cout << "Fuel Efficiency: " << calculateFuelEfficiency() << " miles per gallon\n";
    }
};

// Further derived class: ElectricCar
class ElectricCar : public Vehicle {
protected:
    double batteryCapacity; // in kWh
    double efficiency;      // miles per kWh

public:
    ElectricCar(const string &brand, const string &model, double mileage, double
batteryCapacity, double efficiency)
        : Vehicle(brand, model, mileage), batteryCapacity(batteryCapacity),
efficiency(efficiency) {}

    double calculateRange() const {
        return batteryCapacity * efficiency; // total distance with a full charge
    }

    void displayElectricCarDetails() const {
        displayDetails();
        cout << "Range: " << calculateRange() << " miles with a full charge\n";
    }
};

// Main function
int main() {
    int vehicleType;
    cout << "Enter Vehicle Type (1 for Car, 2 for Electric Car): ";
    cin >> vehicleType;

    string brand, model;
    double mileage;
    cout << "Enter Brand: ";
    cin >> brand;
```

```
    cout << "Enter Model: ";
    cin >> model;
    cout << "Enter Mileage (miles): ";
    cin >> mileage;

    if (vehicleType == 1) {
        double fuel, distanceCovered;
        cout << "Enter Fuel (gallons): ";
        cin >> fuel;
        cout << "Enter Distance Covered (miles): ";
        cin >> distanceCovered;

        Car car(brand, model, mileage, fuel, distanceCovered);
        cout << "\nCar Details:\n";
        car.displayCarDetails();
    } else if (vehicleType == 2) {
        double batteryCapacity, efficiency;
        cout << "Enter Battery Capacity (kWh): ";
        cin >> batteryCapacity;
        cout << "Enter Efficiency (miles per kWh): ";
        cin >> efficiency;

        ElectricCar electricCar(brand, model, mileage, batteryCapacity, efficiency);
        cout << "\nElectric Car Details:\n";
        electricCar.displayElectricCarDetails();
    } else {
        cout << "Invalid vehicle type selected.\n";
    }

    return 0;
}
```

Output:



```
Enter Vehicle Type (1 for Car, 2 for Electric Car): 1
Enter Brand: Swift
Enter Model: 2023
Enter Mileage (miles): 25.75
Enter Fuel (gallons): 9.77
Enter Distance Covered (miles): 250

Car Details:
Brand: Swift, Model: 2023, Mileage: 25.75 miles
Fuel Efficiency: 25.5885 miles per gallon


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