Day 1

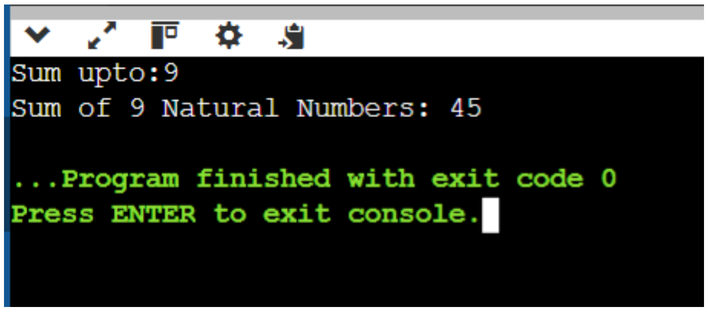
Student Name: Arju Kumari **UID:** 22BCS13364
Branch: BE-CSE **Section/Group:** 22BCS_IOT_603-A
Semester: 5th **Date of Performance:** 19th December, 2024

Problem 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:
Sum=n×(n+1)/2.
Take n as input and output the sum of natural numbers from 1 to n .

Solution :

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

int main()
{
    cout << "Sum upto:";
    int n;
    cin >> n;
    cout << "Sum of " << n << " Natural Numbers: ";
    int sum = n * (n + 1) / 2;
    cout << sum;
    return 0;
}
```

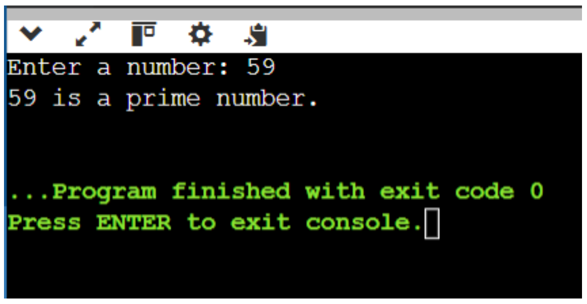
Output:



```
Sum upto:9
Sum of 9 Natural Numbers: 45

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

Problem 2: Check if a Number is Prime.
Check if a given number n is a prime number. A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.

Solution:
```
#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 n;
    cout << "Enter a number: ";
    cin >> n;
    if (isPrime(n)) {
        cout << n << " is a prime number." << endl;
    } else {
        cout << n << " is not a prime number." << endl;
    }
    return 0;
}
```

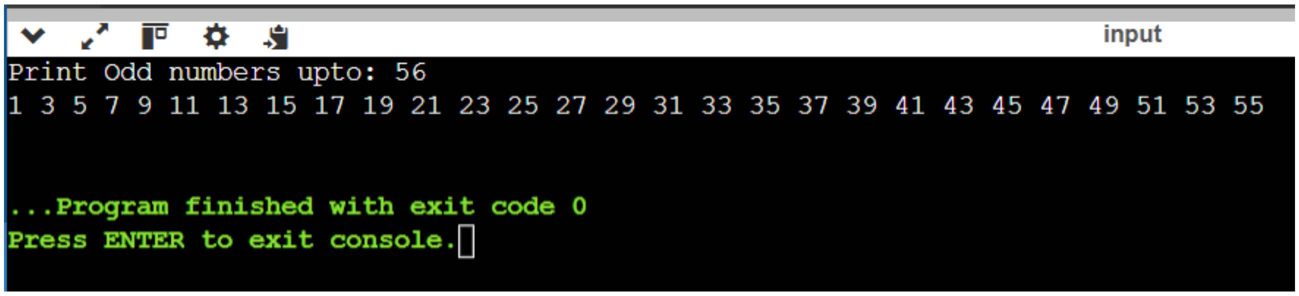
Output:

```
Enter a number: 59
59 is a prime number.


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

Problem 3: Print Odd Numbers up to N.
Print all odd numbers between 1 and n, inclusive. Odd numbers are integers that are not divisible by 2. These numbers should be printed in ascending order, separated by spaces.

Solution:

```
#include <iostream>
using namespace std;
void printOddNumbers(int n) {
    for (int i = 1; i <= n; i += 2) {
        cout << i << " ";
    }
    cout << endl;
}
int main() {
    int n;
    cout << "Print Odd numbers upto: ";
    cin >> n;
    if (n < 1) {
        cout << "Invalid input! n should be greater than or equal to 1." << endl;
        return 1;
    }
    printOddNumbers(n);
    return 0;
}
```

Output:



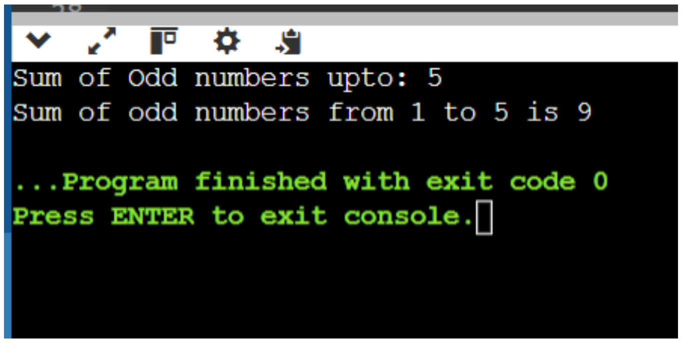
```
Print Odd numbers upto: 56
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55


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

Problem 4: Sum of Odd Numbers up to N**.**
Calculate the sum of all odd numbers from 1 to n. An odd number is an integer that is not divisible by 2. The sum of odd numbers, iterate through all the numbers from 1 to n, check if each number is odd, and accumulate the sum.

Solution:

```
#include <iostream>
using namespace std;
void SumOddNumbers(int n)
{
   int sum = 0;
   for (int i = 1; i <= n; i += 2)
   {
      sum =sum+i;
   }
   cout <<"Sum of odd numbers from 1 to "<<n<<" is "<<sum;
}
int main() {
   int n;
   cout << "Sum of Odd numbers upto: ";
   cin >> n;
   if (n < 1) {
      cout << "Invalid input! n should be greater than or equal to 1." << endl;
      return 1;
   }
   SumOddNumbers(n);
   return 0;
}
```

Output:



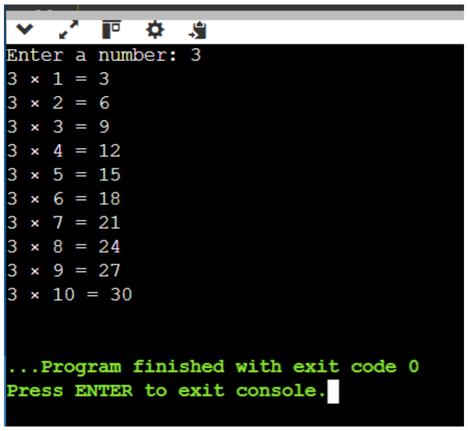
```
Sum of Odd numbers upto: 5
Sum of odd numbers from 1 to 5 is 9

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

Problem 5: Print Multiplication Table of a Number.
Print the multiplication table of a given number n. A multiplication table for a number n is a list of products of n with integers from 1 to 10. For example, the multiplication table for 3 is:
3×1=3,3×2=6,…,3×10=30

Solution:

```
#include <iostream>
using namespace std;
void multiplicationTable(int n) {
    for (int i = 1; i <= 10; i++) {
        cout << n << " × " << i << " = " << n * i << endl;
    }
}
int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    multiplicationTable(n);
    return 0;
}
```

Output:



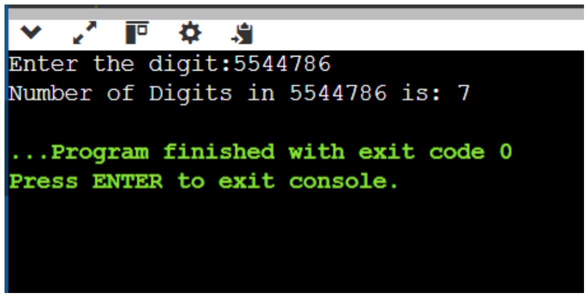
```
Enter a number: 3
3 × 1 = 3
3 × 2 = 6
3 × 3 = 9
3 × 4 = 12
3 × 5 = 15
3 × 6 = 18
3 × 7 = 21
3 × 8 = 24
3 × 9 = 27
3 × 10 = 30


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

Problem 6: Count Digits in a Number
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 determining how many digits are present in n. This task will help you practice working with loops, number manipulation, and conditional logic.

Solution:

```
#include<iostream>
using namespace std;
int Count(int n)
{
    if (n == 0) {
        return 1;
    }
    int count = 0;
    while (n != 0) {
        n /= 10;
        count++;
    }
    return count;
}
int main()
{
    cout << "Enter the digit:";
    int n;
    cin >> n;
    cout << "Number of Digits in " << n << " is: ";
    cout << Count(n);
    return 0;
}
```

Output:



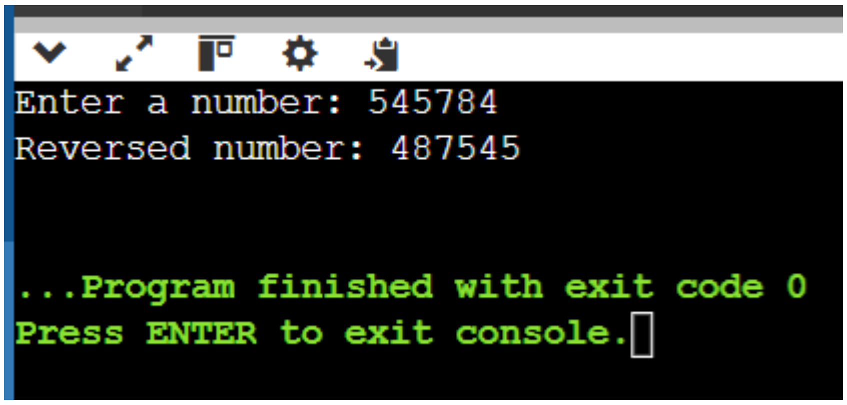
```
Enter the digit:5544786
Number of Digits in 5544786 is: 7

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

Problem 7: Reverse a Number.
Reverse the digits of a given number n. For example, if the input number is 12345, the output should be 54321. The task involves using loops and modulus operators to extract the digits and construct the reversed number.

Solution:

```
#include <iostream>
using namespace std;
int reverseNumber(int n) {
    int reversed = 0;
    while (n != 0) {
        int digit = n % 10;
        reversed = reversed * 10 + digit;
        n = n / 10;
    }
    return reversed;
}
int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    int reversedNumber = reverseNumber(n);
    cout << "Reversed number: " << reversedNumber << endl;
    return 0;
}
```

Output:



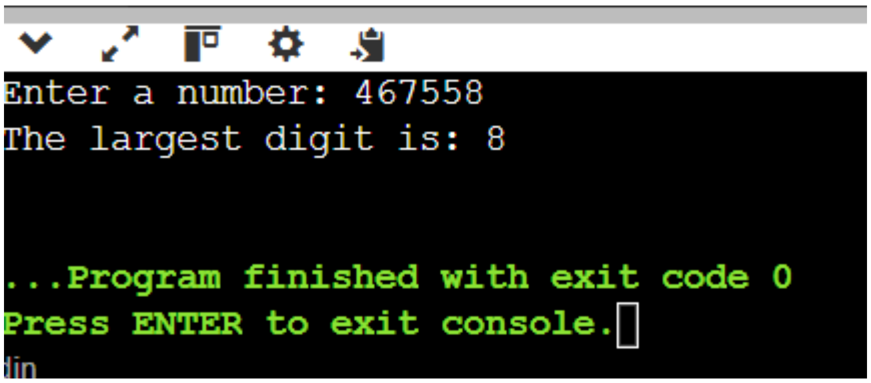
```
Enter a number: 545784
Reversed number: 487545


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

Problem 8: Find the Largest Digit in a Number
Find the largest digit in a given number n. For example, for the number 2734, the largest digit is 7. You need to extract each digit from the number and determine the largest one. The task will involve using loops and modulus operations to isolate the digits.

Solution:

```
#include <iostream>
using namespace std;
int largestDigit(int n) {
    int largest = 0;
    while (n != 0) {
        int digit = n % 10;
        if (digit > largest) {
            largest = digit;
        }
        n = n / 10;
    }
    return largest;
}
int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    int largest = largestDigit(n);
    cout << "The largest digit is: " << largest << endl;
    return 0;
}
```

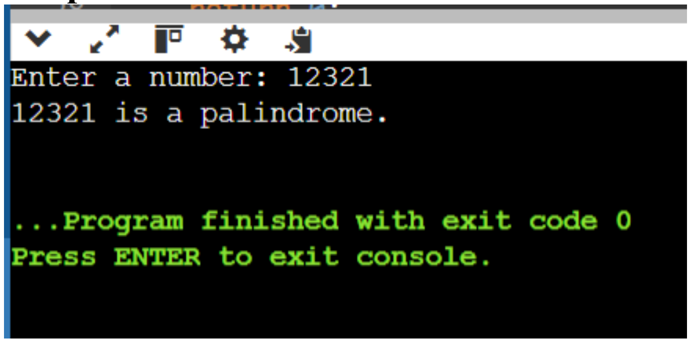
Output:



```
Enter a number: 467558
The largest digit is: 8


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

Problem 9: Check if a Number is a Palindrome
Check whether a given number is a palindrome or not. A number is called a palindrome if it reads the same backward as forward. For example, 121 is a palindrome because reading it from left to right is the same as reading it from right to left. Similarly, 12321 is also a palindrome, but 12345 is not.

Solution:
```
#include <iostream>
using namespace std;
bool isPalindrome(int n) {
    int original = n;
    int reversed = 0;
    while (n != 0) {
        int digit = n % 10;
        reversed = reversed * 10 + digit;
        n = n / 10;
    }
    return original == reversed;
}
int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    if (isPalindrome(n)) {
        cout << n << " is a palindrome." << endl;
    } else {
        cout << n << " is not a palindrome." << endl;
    }
    return 0;
}
```

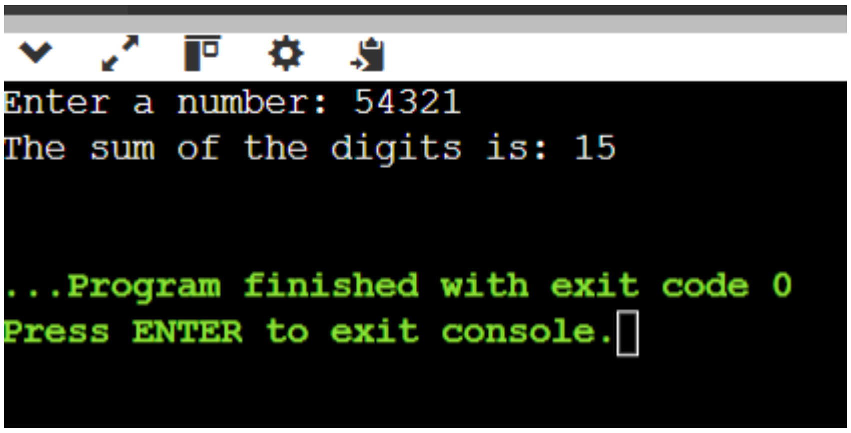
Output:

```
Enter a number: 12321
12321 is a palindrome.


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

Problem 10: Find the Sum of Digits of a Number
Calculate the sum of the digits of a given number n. For example, for the number 12345, the sum of the digits is 1+2+3+4+5=15. To solve this, you will need to extract each digit from the number and calculate the total sum.

Solution:

```
#include <iostream>
using namespace std;
int sumOfDigits(int n) {
    int sum = 0;
    while (n != 0) {
        sum += n % 10;
        n = n / 10;
    }
    return sum;
}
int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    int sum = sumOfDigits(n);
    cout << "The sum of the digits is: " << sum << endl;
    return 0;
}
```

Output:



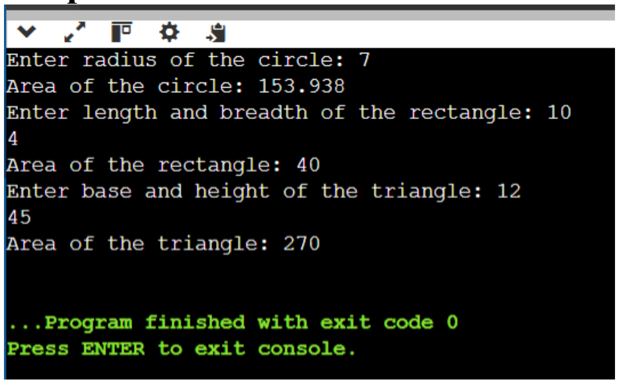
```
Enter a number: 54321
The sum of the digits is: 15


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

Problem 11: Function Overloading for Calculating Area.
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.

Solution:

```
#include <iostream>
#include <cmath>
using namespace std;
const double PI = 3.14159;
double area(double radius) {
    return PI * radius * radius;
}

double area(double length, double breadth) {
    return length * breadth;
}

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

int main() {
    double radius, length, breadth, base, height;

    cout << "Enter radius of the circle: ";
    cin >> radius;
    cout << "Area of the circle: " << area(radius) << endl;

    cout << "Enter length and breadth of the rectangle: ";
    cin >> length >> breadth;
    cout << "Area of the rectangle: " << area(length, breadth) << endl;

    cout << "Enter base and height of the triangle: ";
    cin >> base >> height;
    double n = 0.5;
    cout << "Area of the triangle: " << area(base, height, n) << endl;

    return 0;
}
```

Output:



```
Enter radius of the circle: 7
Area of the circle: 153.938
Enter length and breadth of the rectangle: 10
4
Area of the rectangle: 40
Enter base and height of the triangle: 12
45
Area of the triangle: 270


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

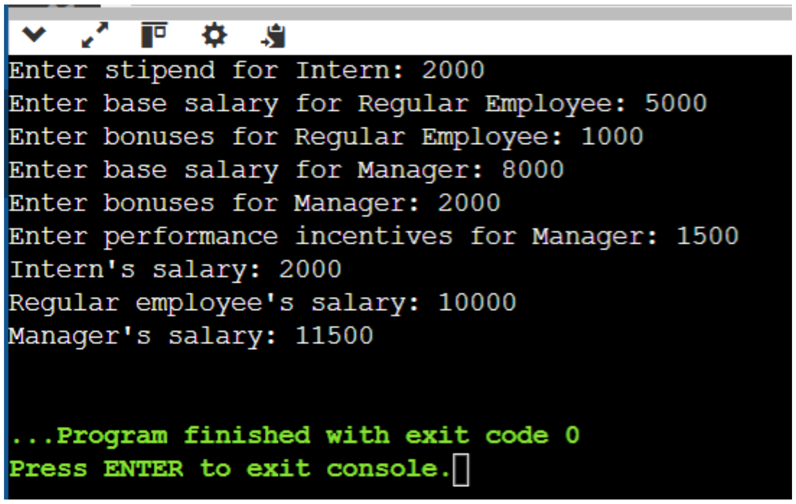
Problem 12: Function Overloading with Hierarchical Structure.
Write a program that demonstrates function overloading to calculate the salary of employees at different levels in a company hierarchy. Implement overloaded functions to compute salary for:

- Intern (basic stipend).
- Regular employee (base salary + bonuses).
- Manager (base salary + bonuses + performance incentives).

Solution:

```
#include <iostream>
using namespace std;
class Employee {
public:
    double calculateSalary(double stipend) {
        return stipend;
    }
    double calculateSalary(double baseSalary, double bonuses) {
        return baseSalary + bonuses;
    }
    double calculateSalary(double baseSalary, double bonuses, double
performanceIncentives) {
        return baseSalary + bonuses + performanceIncentives;
    }
};
int main() {
    Employee emp;
    double stipend, baseSalary, bonuses, performanceIncentives;
    cout << "Enter stipend for Intern: ";
    cin >> stipend;
    cout << "Enter base salary for Regular Employee: ";
    cin >> baseSalary;
    cout << "Enter bonuses for Regular Employee: ";
    cin >> bonuses;
    cout << "Enter base salary for Manager: ";
    cin >> baseSalary;
    cout << "Enter bonuses for Manager: ";
    cin >> bonuses;
    cout << "Enter performance incentives for Manager: ";
    cin >> performanceIncentives;
    cout << "Intern's salary: " << emp.calculateSalary(stipend) << endl;
    cout << "Regular employee's salary: " << emp.calculateSalary(baseSalary,
bonuses) << endl;
    cout << "Manager's salary: " << emp.calculateSalary(baseSalary, bonuses,
performanceIncentives) << endl;
    return 0;
}
```

Output:



```
Enter stipend for Intern: 2000
Enter base salary for Regular Employee: 5000
Enter bonuses for Regular Employee: 1000
Enter base salary for Manager: 8000
Enter bonuses for Manager: 2000
Enter performance incentives for Manager: 1500
Intern's salary: 2000
Regular employee's salary: 10000
Manager's salary: 11500


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

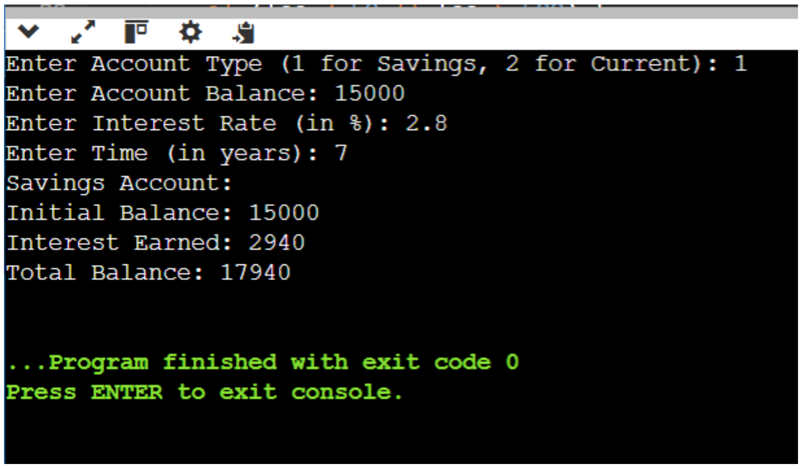
Problem 20: Implement Polymorphism for Banking Transactions
Design a C++ program to simulate a banking system using polymorphism. Create a base class Account with a virtual method calculateInterest(). Use the derived classes SavingsAccount and CurrentAccount to implement specific interest calculation logic:
SavingsAccount: Interest = Balance × Rate × Time.
CurrentAccount: No interest, but includes a maintenance fee deduction.

Solution:

```
#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 rate, int time) : Account(bal), rate(rate),
time(time) {}

    void calculateInterest() override {
        double interest = balance * (rate / 100) * time;
        cout << "Savings Account:\n";
        cout << "Initial Balance: " << balance << endl;
        cout << "Interest Earned: " << interest << endl;
        cout << "Total Balance: " << (balance + interest) << endl;
    }
};

class CurrentAccount : public Account {
    double maintenanceFee;
```

```
public:
    CurrentAccount(double bal, double fee) : Account(bal), maintenanceFee(fee)
{}

    void calculateInterest() override {
        double finalBalance = balance - maintenanceFee;
        cout << "Current Account:\n";
        cout << "Initial Balance: " << balance << endl;
        cout << "Maintenance Fee Deducted: " << maintenanceFee << endl;
        cout << "Total Balance: " << finalBalance << endl;
    }
};

int main() {
    int accountType;
    double balance;

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

    if (accountType < 1 || accountType > 2) {
        cout << "Invalid account type!" << endl;
        return 1;
    }

    cout << "Enter Account Balance: ";
    cin >> balance;

    if (balance < 1000 || balance > 1000000) {
        cout << "Invalid balance!" << endl;
        return 1;
    }

    Account* account = nullptr;

    if (accountType == 1) {
        double rate;
        int time;

        cout << "Enter Interest Rate (in %): ";
        cin >> rate;
        cout << "Enter Time (in years): ";
        cin >> time;

        if (rate < 1 || rate > 15 || time < 1 || time > 10) {
            cout << "Invalid interest rate or time!" << endl;
            return 1;
        }

        account = new SavingsAccount(balance, rate, time);

    } else if (accountType == 2) {
        double fee;

        cout << "Enter Monthly Maintenance Fee: ";
        cin >> fee;

        if (fee < 50 || fee > 500) {
            cout << "Invalid maintenance fee!" << endl;
            return 1;
        }

        account = new CurrentAccount(balance, fee);
    }

    if (account) {
        account->calculateInterest();
        delete account;
    }

    return 0;
}
```

Output:



```
Enter Account Type (1 for Savings, 2 for Current): 1
Enter Account Balance: 15000
Enter Interest Rate (in %): 2.8
Enter Time (in years): 7
Savings Account:
Initial Balance: 15000
Interest Earned: 2940
Total Balance: 17940


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