

Daily Assignment – 19th Dec 2024

Day 1

Student Name: Ishan Jain

UID: 22BCS13653

Branch: CSE

Section/Group: IOT_615-B

Subject Name: Winter Winning Camp

Date of Performance: 19-Dec-2024

Problem: 1

1. **Aim:** Sum of Natural Numbers up to N.
2. **Objective:** 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.
3. **Code:**

```
#include <iostream> using
namespace std; int main(){ int n;
cout << "Enter a number: "; cin
    >> n; cout << n* (n + 1) / 2;
    return 0;
}
```

4. **Output:**



```
< 5
15
...Program finished with exit code 0
Press ENTER to exit console.
```

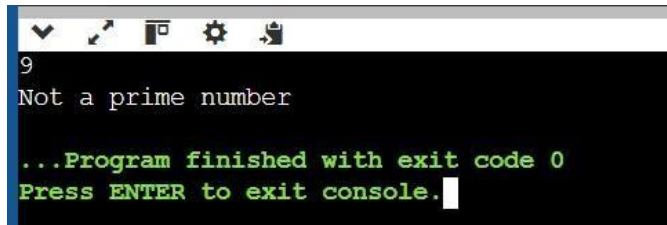
Problem: 2

1. **Aim:** Check if a Number is Prime.
2. **Objective:** 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. To determine if a number is prime, iterate from 2 to \sqrt{n} and check if n is divisible by any number in this range. If it is divisible, it is not a prime number; otherwise, it is a prime.
3. **Code:**

```
#include <iostream> using
namespace std; int main()
{ int n; cin >> n;
  int
  count =
  0; for (int i = 1; i <= n;
  i++)
  {

    if (n % i == 0)
    { cout <<
      +;
    } } if
(count == 2)
{ cout << "Prime" << endl;
} else if (count >
2)
{ cout << "Not Prime";
} else if (n ==
1)
{ cout << "Neither Prime nor even";
} else
{ cout << "Cannot determine";
}
}
```

4. Output:



```
9
Not a prime number

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

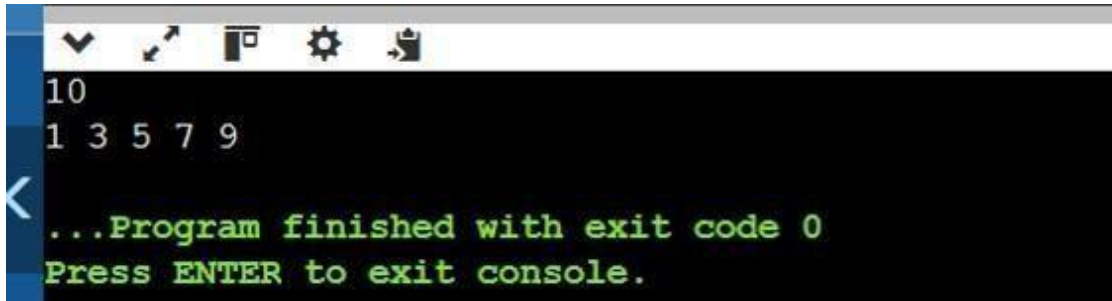
Problem: 3

1. **Aim:** Print Odd Numbers up to N.
2. **Objective:** 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. This problem is a simple introduction to loops and conditional checks. The goal is to use a loop to iterate over the numbers and check if they are odd using the condition $i \% 2 \neq 0$.

3. **Code:**

```
#include <iostream> using
namespace std; int main()
{ int n;
  cin
  >>
  n;
  for (int i = 1; i < n; i++)
  { if (i % 2
    != 0)
    { cout << i <<
      " ";
    }
  }
}
```

4. **Output:**



```
10
1 3 5 7 9
...Program finished with exit code 0
Press ENTER to exit console.
```

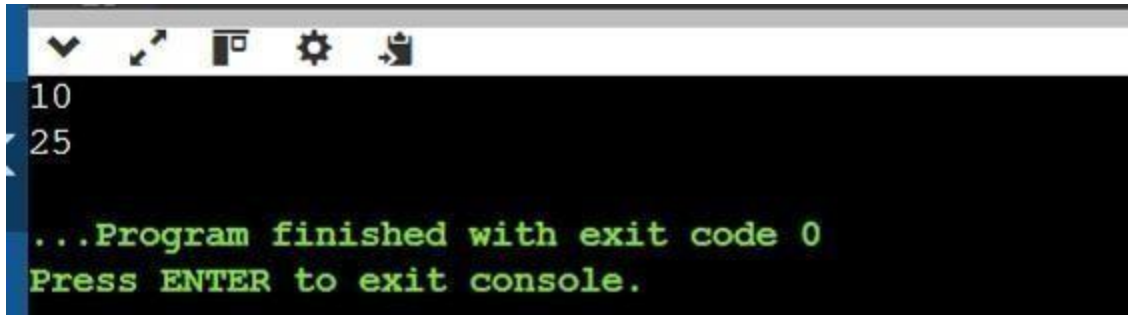
Problem: 4

1. **Aim:** Sum of Odd Numbers up to N.
2. **Objective:** 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.

3. **Code:**

```
#include <iostream> using
    namespace
std;
int main()
{ int n; cin
>> n; int
sum
= 0; for (int i = 1; i <= n; i++)
{ if (i % 2
    != 0)
    { sum =
      sum + i;
    }
} cout << "Sum of odd numbers: " <<
sum;
}
```

4. **Output:**



```
10
25

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

Problem: 5

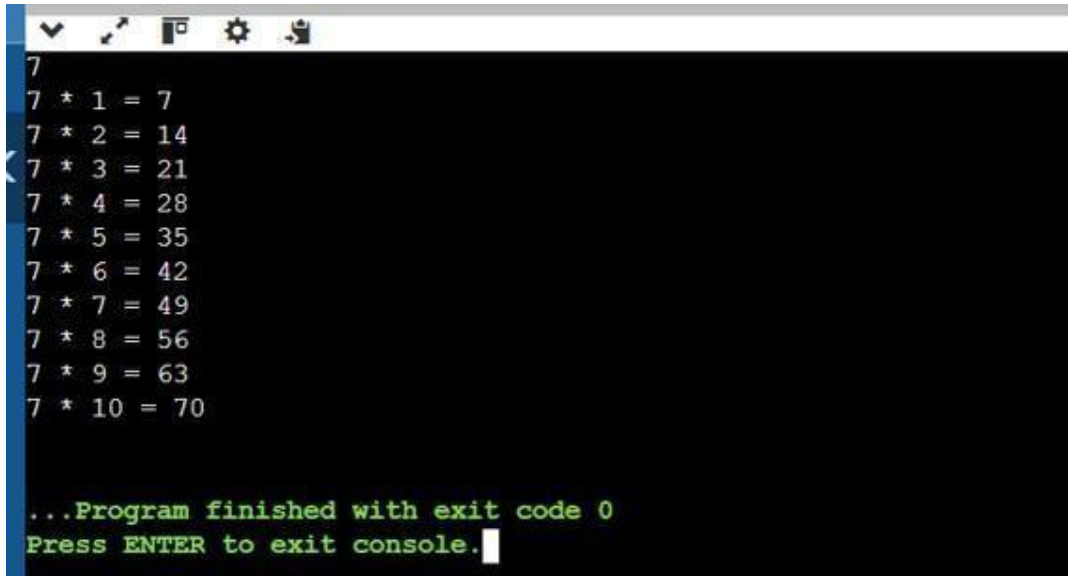
1. **Aim:** Print Multiplication Table of a Number.
2. **Objective:** Print the multiplication table of a given number n. A multiplication table for a number n is a list of products of n with integers from 1 to 10. For example, the multiplication table for 3 is:

$3 \times 1 = 3, 3 \times 2 = 6, \dots, 3 \times 10 = 30.$

3. **Code:**

```
#include <iostream> using
namespace std; int main()
{ int n;
  cin
  >>
  n; for (int i = 1; i <= 10; i++)
  { cout << n << " * " << i << " = " << n * i << endl;
    }
}
```

4. **Output:**



```
7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70

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

Problem: 6

1. **Aim:** Count Digits in a Number.
2. **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.

3. **Code:**

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

4. **Output:**



```
< 12345
5
...Program finished with exit code 0
Press ENTER to exit console.
```

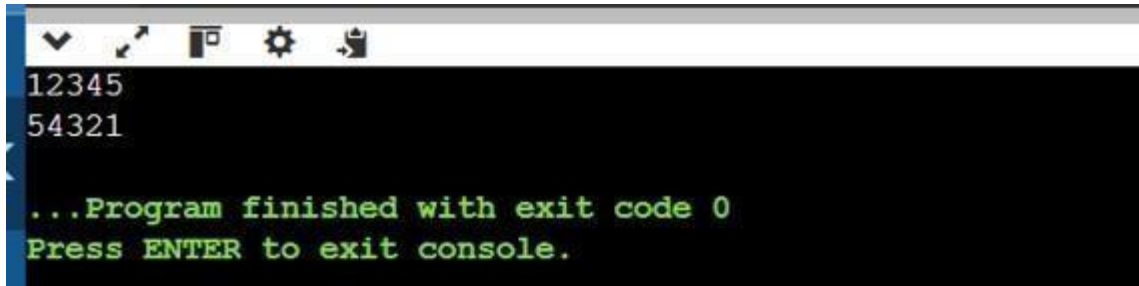
Problem: 7

1. **Aim:** Reverse a Number.
2. **Objective:** 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.

3. **Code:**

```
#include <iostream> using
namespace std; int main()
{ int n; cin >>
  n; int rev =
  0; while (n >
  0)
  { int rem = n % 10; rev = rev
    * 10 + rem; n =
    n / 10;
  } cout << "Reverse of number: " << rev;
}
```

4. **Output:**



```
12345
54321

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

Problem: 8

1. **Aim:** Find the Largest Digit in a Number.
2. **Objective:** 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.

3. **Code:**

```
#include <iostream> using namespace
std; int main()
{ int n; cin >> n; while
(n >
0)
{ int max = n % 10; n
= n / 10; while
(n > 0)
{ int rem = n % 10; if
(rem > max)
{
max = rem;
}
n = n / 10;
}
cout << "Maximum Digit: " << max << " ";
}
```




DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING



}

4. Output:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



```
13764
7

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

Problem: 9

1. **Aim:** Check if a Number is a Palindrome.
2. **Objective:** 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.

3. **Code:**

```
#include <iostream> using namespace
std; int main()
{ int n; cin >> n; int
  rev = 0; int temp
  = n; while
  (temp > 0)
  {

    int rem = temp % 10; rev
    = rev * 10 + rem; temp
    = temp / 10;
  } if (rev ==
n)
```

```
{ cout << "Pallindrome";  
} else  
{ cout << "Not Pallindrome"; } }
```

4. Output:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



```
14541  
Palindrome  
...Program finished with exit code 0  
Press ENTER to exit console.
```

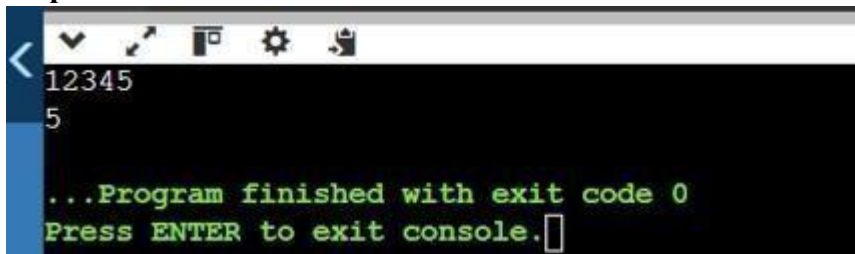
Problem: 10

1. **Aim:** Find the Sum of Digits of a Number.
2. **Objective:** 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.
3. **Code:**

```
#include <iostream>  
using namespace std;  
int main()
```

```
{ int n; cin >> n; int temp
= n; int
sum = 0; while
(temp > 0)
{ int rem = temp % 10; temp
= temp / 10; sum
= sum + rem;
} cout << "The sum of the digits is: " << sum; }
```

4. Output:



```
< 12345
5
...Program finished with exit code 0
Press ENTER to exit console.
```

Problem: 11

1. **Aim:** Function Overloading for Calculating Area.
2. **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.
3. **Code:** #include <iostream> using namespace std;

```
double calculateArea1(double r) {
    double area1; area1 = r * r *
    3.14159; return area1;
}
```

```
double calculateArea2(double l, double b) { double
area2; area2
= l * b;
return area2;
```

```
}  
  
double calculateArea3(double base, double h) {  
    double area3; area3 = 0.5 * base * h; return  
    area3;  
}  
  
int main() { double r, l,  
    b, base, h;  
  
    // Input for circle  
    cout << "Radius:  
    "; cin >> r; cout  
    << endl;  
  
    // Input for  
    rectangle cout <<  
    "Length: "; cin >> l;  
    cout << "Breadth: ";  
    cin >> b; cout <<  
    endl;  
  
    // Input for  
    triangle cout <<  
    "Base: "; cin >>  
    base; cout <<  
    "Height: "; cin >>  
    h;
```

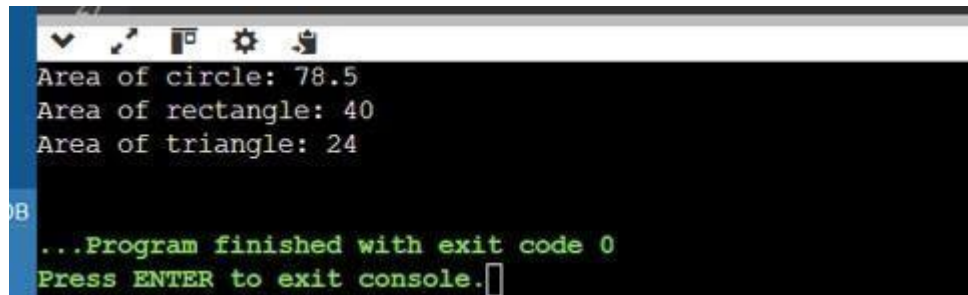
```
cout << endl;

// Calculate areas double area1 =
calculateArea1(r); double area2 =
calculateArea2(l, b); double area3 =
calculateArea3(base, h);

// Output the areas cout << "Area of Circle: "
<< area1 << endl; cout << "Area of Rectangle:
" << area2 << endl; cout << "Area of Triangle:
" << area3 << endl;

return 0;
}
```

4. Output:



```
Area of circle: 78.5
Area of rectangle: 40
Area of triangle: 24
...Program finished with exit code 0
Press ENTER to exit console.
```

Problem: 12

1. **Aim:** Function Overloading with Hierarchical Structure.
2. **Objective:** 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:
 - a. Intern (basic stipend).
 - b. Regular employee (base salary + bonuses).
 - c. Manager (base salary + bonuses + performance incentives).

3. Code:

```
#include <iostream> using namespace
std;    class
Company
{ public: int calculatesalary()
{
}
};

class Intern : public Company
{ public: int stipend;
int
input()
{ cout << "Stipend: "; cin
>> stipend;
}    int calculatesalary()
{ cout << "Intern Salary: " << stipend << endl; }
};

class RegularEmployee : public Company
{ public: int
BaseSalary; int
bonuses;
int input()
{ cout << "Base Salary: ";
cin >> BaseSalary; cout
<< "Bonuses: "; cin >>
bonuses;    }    int
calculatesalary()
{ int empsalary = BaseSalary + bonuses; cout <<
"Employee Salary: " << empsalary << endl; }
};

class Man : public Company
{ public: int
BaseSalary; int
bonuses; int
incentives;    int input()
```

```
{ cout << "Base Salary: "; cin >>
  BaseSalary; cout << "Bonuses: ";
  cin >> bonuses; cout <<
  "Performan Incentives: "; cin >>
  incentives;
}
int calculatesalary()
{ int mansalary = BaseSalary + bonuses + incentives; cout
  << "Mnager Salary: " << mansalary << endl; }
};
```

```
int main()
{ int stipend; int
  BaseSalary; int
  Bonuses; int
  incentives;
  Intern    i1;
  i1.input();
```

```
RegularEmployee r1; r1.input();
```

```
Man m1; m1.input();
```

```
i1.calculatesalary(); r1.calculatesalary();
m1.calculatesalary(
)
; }
```

4. Output:


```
input
Enter Employee Type (1 for Manager, 2 for Developer): 2
Enter Developer Details:
Name: Saloni
ID: 16659
Salary: 45000
Extra Hours Worked: 6
Hourly Rate: 10
Name: Saloni
ID: 16659
Base Salary: 45000
Overtime Compensation: 60
Total Earnings (Salary + Overtime Compensation): 45060

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

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
input
Enter Employee Type (1 for Manager, 2 for Developer): 1
Enter Manager Details:
Name: Saloni
ID: 16659
Salary: 60000
Performance Rating (percentage): 9.5
Name: Saloni
ID: 16659
Base Salary: 60000
Bonus: 5700
Total Earnings (Salary + Bonus): 65700
...Program finished with exit code 0
Press ENTER to exit console.
```

Problem: 13

1. **Aim:** Encapsulation with Employee Details.
2. **Objective:** Write a program that demonstrates encapsulation by creating a class Employee.
The class should have private attributes to store:
Employee ID.
Employee Name.
Employee Salary.
Provide public methods to set and get these attributes, and a method to display all details of the employee.

3. **Code:**

```
#include<iostream>
#include<iomanip>
using namespace std;
class Employee{ int ID;
    string  name; double
    salary; public:
    Employee(int ID,string name,double salary){ this->ID
        = ID; this->name = name; this->salary
        = salary;

    }

    void display(){ cout<<"Employee ID: " <<ID<<endl; cout<<"Employee
        Name: " <<name<<endl;
        cout<<"Employee          Salary:          "
        <<fixed<<setprecision(2)<<salary; } }; int main(){ int ID; string
        name; double salary; cout<<"ID: "; cin>>ID; cout<<"Name: ";
        cin.ignore(); // to clear the input buffer before reading the name
        getline(cin,name); cout<<"Salary: "; cin>>salary;
        Employee    e1(ID,name,salary); e1.display();

    }
```

4. Output:



```
input
Enter Employee Type (1 for Manager, 2 for Developer): 1
Enter Manager Details:
Name: Saloni
ID: 16659
Salary: 60000
Performance Rating (percentage): 9.5
Name: Saloni
ID: 16659
Base Salary: 60000
Bonus: 5700
Total Earnings (Salary + Bonus): 65700
...Program finished with exit code 0
Press ENTER to exit console.
```

Problem: 14

1. **Aim:** Inheritance with Student and Result Classes.
2. **Objective:** Create a program that demonstrates inheritance by defining:
 - a. A base class Student to store details like Roll Number and Name.
 - b. A derived class Result to store marks for three subjects and calculate the total and percentage.
3. **Code:**

```
#include<iostream>
using namespace
std; class Student{
public: int rollno; string
name;
};
```

```
class Results:public Student{ public: int
    marks1,marks2,marks3; void display(){
    cout<<"Roll No:
    "<<rollno<<endl; cout<<"Name:
        "<<name<<endl;                                cout<<"Marks:
        "<<marks1<<","<<marks2<<","<<marks3<<endl;
    }    int total(){
    int total =
    marks1+
    marks2+
    marks3;
    return total;
    }    int percent(){    float    percentage    =
    (marks1+marks2+marks3)/3.0;
    return percentage;
    }    };
```

```
int main(){ int rollno; string name; int marks1,marks2,marks3;
    cout<<"Roll No: "; cin>>rollno; cout<<"Name:
        ";
        cin.ignore(); getline(cin,name);
    cout<<"Marks: ";
    cin>>marks1>>marks2>>marks3
    ;
    Results r1; r1.rollno =
    rollno;    r1.name    =
    name;    r1.marks1    =
    marks1;    r1.marks2    =
    marks2;    r1.marks3    =
    marks3; r1.display();
    cout<<"Total marks: "<<r1.total()<<endl; cout<<"Percentage:
    "<<r1.percent(); }
```

4. Output:

```
input
Enter Account Type (1 for Savings, 2 for Current): 2
Enter Balance: 50000
Enter Monthly Maintenance Fee: 500
Balance after fee deduction: 49500

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

Problem: 15

1. **Aim:** Polymorphism with Shape Area Calculation.
2. **Objective:** Create a program that demonstrates polymorphism by calculating the area of different shapes using a base class Shape and derived classes for Circle, Rectangle, and Triangle. Each derived class should override a virtual function to compute the area of the respective shape.
3. **Code:**

```
#include <iostream>
#include <iomanip>
using namespace std;
double calculatearea1(double r)
{ double area1; area1 = r *
  r * 3.14159; return
  area1;
}

double calculatearea2(double l, double b)
{ double area2;
  area2 = l * b;
  return area2;
}

double calculatearea3(double base, double h)
```

```
{ double area3; area3 =  
    0.5 * base * h; return  
    area3;  
}
```

```
int main()  
{ double r, l, b, base, h;  
    cout << "Radius: ";  
    cin >> r; cout <<  
    endl;    cout <<  
    "Length: "; cin >> l;  
    cout << "Breadth: ";  
    cin >> b; cout <<  
    endl;  
  
    cout << "Base: "; cin >> base; cout << "Height: ";  
    cin >> h; cout << endl; double area1 =  
    calculatearea1(r);    double area2 =  
    calculatearea2(l,    b);    double area3 =  
    calculatearea3(base, h); cout << fixed <<  
    setprecision(2) << area1 << endl; cout << area2 <<  
    endl; cout << area3;  
}
```

4. Output:

```
▼ ↗ 📄 ⚙️ 📋
Select a shape to calculate the area:
1. Circle
2. Rectangle
3. Triangle
Enter your choice (1-3): 3
Enter the base and height of the Triangle: 10 7.5
Shape: Triangle
Base: 10.000
Height: 7.500
Area: 37.500

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

Problem: 16

1. **Aim:** Implementing Polymorphism for Shape Hierarchies.

2. **Objective:** Write a program to demonstrate runtime polymorphism in C++ using a base class Shape and derived classes Circle, Rectangle, and Triangle. The program should use virtual functions to calculate and print the area of each shape based on user input.

3. **Code:**

```
#include <iostream> using
namespace std; class Shape
{ public: virtual double
area() = 0;
}; class Rectangle : public Shape { double
length, breadth; public: void
setDimensions(double l,
double b) { length = l; breadth = b; }
```



```
double area() override { return length
* breadth; } }; class Circle : public
Shape { double radius; public: void
setDimensions(double r)
{ radius = r; }

double area() override { return
3.14159 * radius * radius;
}
}; class Triangle : public
Shape { double base, height;

public: void setDimensions(double    b,
double h) { base = b;
height = h; } double area()
override { return 0.5 * base
* height;
} };

int main()
{ double l, b, r, base, h;
```

```
cout << "Radius of Circle: "; cin
>> r;
```

```
cout << "Length of Rectangle: ";
cin >> l;
```

```
cout << "Breadth of Rectangle: "; cin
```

```
>> b; cout << "Base of Triangle:
"; cin
>> base;

cout << "Height of Triangle: "; cin >> h;

Circle c1; c1.setDimensions(r);

Rectangle r1; r1.setDimensions(l,
b);

Triangle t1; t1.setDimensions(base, h);

cout << "Area of Circle: " << c1.area() << endl;
cout << "Area of Rectangle: " << r1.area() << endl;
cout << "Area of Triangle: " << t1.area() << endl;
return 0;
}
```

4. Output:

```
▼ ↗ 📄 ⚙️ 📋
Select a shape to calculate the area:
1. Circle
2. Rectangle
3. Triangle
Enter your choice (1-3): 3
Enter the base and height of the Triangle: 10 7.5
Shape: Triangle
Base: 10.000
Height: 7.500
Area: 37.500

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

Problem: 17

1. **Aim:** Matrix Multiplication Using Function Overloading.
2. **Objective:** Implement matrix operations in C++ using function overloading. Write a function `operate()` that can perform:
 - a. Matrix Addition for matrices of the same dimensions.
 - b. Matrix Multiplication where the number of columns of the first matrix equals the number of rows of the second matrix.
3. **Code:** `#include`
`<iostream>` using
namespace `std`;

```
void operate(int A[10][10], int B[10][10], int result[10][10], int m, int n, int p, int operation)
{ if (operation == 1) {
```

```

if (m == p) { for (int i = 0; i
    < m; i++) { for (int j =
    0; j < n; j++) {
    result[i][j] = A[i][j] +
    B[i][j];
    } } for (int i
    = 0; i < m; i++)
    { for (int j = 0; j
    < n; j++) { cout
    << result[i][j] << "
    ";
    } cout <<
    endl;
    }
} else { cout << "Invalid dimensions for operation." << endl;
}
}
}

```

```

void operate(int A[10][10], int B[10][10], int result[10][10], int m, int n, int p, int q, int
operation) { if (operation == 2) { if (n == p) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < q; j++)
            { result[i][j] = 0; for (int k
            = 0; k < n; k++)
                { result[i][j] += A[i][k] *
                B[k][j]; }
            } }
    for (int i = 0; i < m; i++)
        { for (int j = 0;
        j < q; j++) { cout

```

```
        <<
        result[i][j] << " ";
    }
    cout << endl;
}
} else { cout << "Invalid dimensions for operation." << endl;
}
}
}

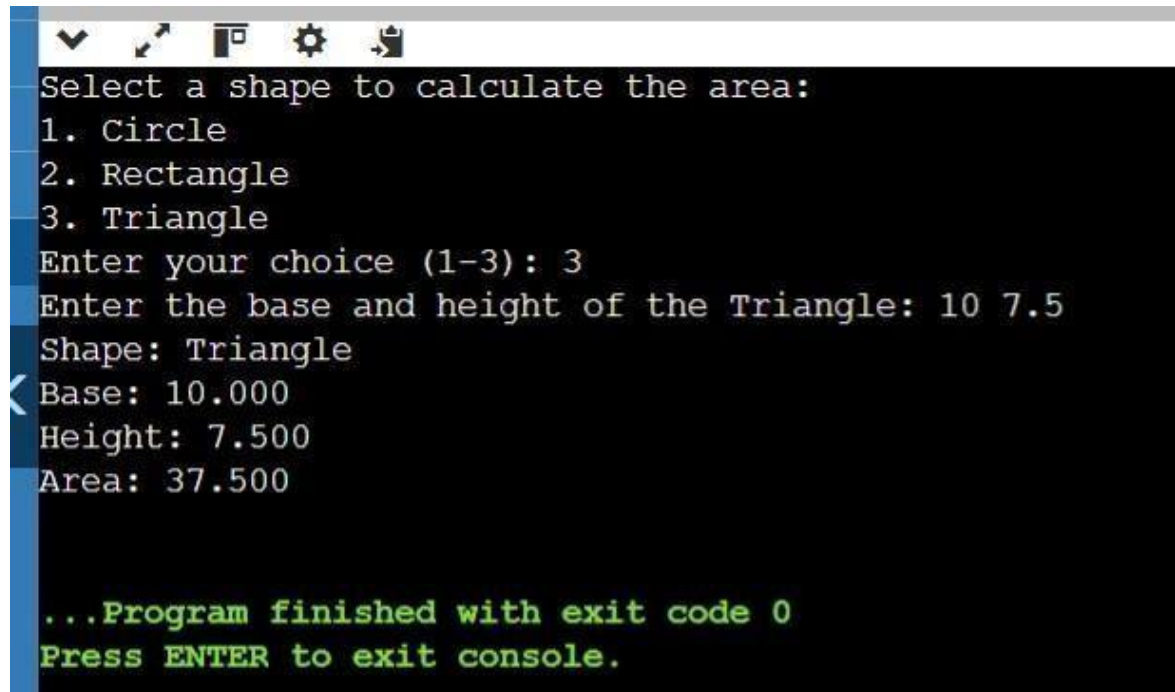
int main() { int m,
n, p, q,
operation;

cout << "Enter dimensions of Matrix A (m x n):
"; cin >> m >> n; int A[10][10]; cout << "Enter
elements of Matrix A:" << endl; for (int i = 0;
i < m; i++)
    { for (int j =
0; j < n; j++) { cin
>> A[i][j];
    }
}

cout << "Enter dimensions of Matrix B (p x q):
"; cin >> p >> q; int B[10][10]; cout << "Enter
elements of Matrix B:" << endl;
for (int i = 0; i < p; i++)
    { for (int j =
0; j < q; j++) { cin
>> B[i][j];
    }
}
```

```
cout << "Enter operation type (1 for Addition, 2 for Multiplication): "; cin  
>> operation; int result[10][10];  
  
operate(A, B, result, m, n, p, operation); operate(A,  
B, result, m, n, p, q, operation); return 0;  
}
```

4. Output:



```
Select a shape to calculate the area:  
1. Circle  
2. Rectangle  
3. Triangle  
Enter your choice (1-3): 3  
Enter the base and height of the Triangle: 10 7.5  
Shape: Triangle  
Base: 10.000  
Height: 7.500  
Area: 37.500  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Problem: 18

1. **Aim:** Polymorphism in Shape Classes.
2. **Objective:** Design a C++ program using polymorphism to calculate the area of different shapes:
 - a. A Rectangle (Area = Length \times Breadth).

- b. A Circle ($\text{Area} = \pi \times \text{Radius}^2$).
- c. A Triangle ($\text{Area} = \frac{1}{2} \times \text{Base} \times \text{Height}$).
- d. Create a base class Shape with a pure virtual function `getArea()`. Use derived classes Rectangle, Circle, and Triangle to override this function

3. **Code:**

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

```
class Shape { public:  
    // Pure virtual function virtual void getarea() = 0; // No arguments here, as the derived  
    classes will handle it  
    differently };
```

```
class Rectangle : public Shape { public: int  
    l, b;
```

```
    Rectangle(int length, int breadth) : l(length), b(breadth) {}
```

```
    // Override the pure virtual function void getarea()  
    override { cout << "Area of Rectangle is: " << l * b  
    << endl; }  
};
```

```
class Circle : public Shape { public: int  
    r;
```

```
    Circle(int radius) : r(radius) {}
```

```
    // Override the pure virtual function void getarea() override  
    { cout << "Area of Circle is: " << 3.14159 * r * r << endl;  
    }  
};
```

```
class Triangle : public Shape { public:  
    int b, h;
```

```
Triangle(int base, int height) : b(base), h(height) {}
```

```
// Override the pure virtual function void getarea()
override { cout << "Area of Triangle is: " << 0.5 * b * h <<
endl; } };
```

```
int main() { int type;
```

```
cout << "Enter 1 for Rectangle, 2 for Circle, 3 for Triangle: " << endl; cin >>
type;
```

```
if (type == 1) { int l,
b;
```

```
cout << "Enter length and breadth: " <<
endl; cin >> l >> b; Rectangle r1(l, b);
```

```
r1.getarea(); // Call the getarea method for Rectangle
```

```
}
```

```
else if (type == 2) { int r;
```

```
cout << "Enter radius: " << endl;
cin >> r; Circle
```

```
c1(r);
```

```
c1.getarea(); // Call the getarea method for Circle
```

```
}
```

```
else if (type == 3) { int b,
```

```
h;
```

```
cout << "Enter base and height: " << endl; cin
>> b >> h; Triangle
```

```
t1(b, h);
```

```
t1.getarea(); // Call the getarea method for Triangle
```

```
} else
```

```
{
```

```
cout << "Invalid Shape Type" << endl;
```

```
}
```

```
return 0;
```

```
}
```




DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING



4. Output:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
▼ ↗ 📄 ⚙️ 📌  
Select a shape to calculate the area:  
1. Circle  
2. Rectangle  
3. Triangle  
Enter your choice (1-3): 3  
Enter the base and height of the Triangle: 10 7.5  
Shape: Triangle  
Base: 10.000  
Height: 7.500  
Area: 37.500  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Problem: 19

1. **Aim:** Implement Multiple Inheritance to Simulate a Library System.
2. **Objective:** Create a C++ program using multiple inheritance to simulate a library system.
Design two base classes:
 - a. Book to store book details (title, author, and ISBN).
 - b. Borrower to store borrower details (name, ID, and borrowed book).
 - c. Create a derived class Library that inherits from both Book and Borrower. Use this class to track the borrowing and returning of books.
3. **Code:**

```
#include <iostream>
```

```
#include
```

```
<string> using
```

```
namespace std;
```

```
class Book { public:
```

```
    string    title;
```

```
    string
```

```
    author;    int
```

```
    ISBN;
```

```
    Book(string t, string a, int i) : title(t), author(a), ISBN(i) {}  
};
```

```
class Borrower
```

```
{ public: string
```

```
name; int    ID;
```

```
bool
```

```
hasBorrowed;
```

```
    Borrower(string n, int id) : name(n), ID(id), hasBorrowed(false) {} };
```

```
class Library : public Book, public Borrower { public:
```

```
    Library(string t, string a, int i, string n, int id) : Book(t, a, i), Borrower(n, id) {}
```

```
    void borrowBook() { if (hasBorrowed) { cout << "Borrower " << name << " (ID: "
```

```
<< ID << ") has already borrowed a book."
<< endl;

    } else { hasBorrowed = true; cout << "Borrower " << name << " (ID: " << ID <<
    ") has borrowed \"" << title << "\" by " << author << " (ISBN: " << ISBN << ")." <<
    endl;
    }
}

void returnBook() { if (hasBorrowed) { cout << "Borrower " << name << " (ID: " << ID
    << ") has returned \"" << title <<
    "\" by " << author << " (ISBN: " << ISBN << ")." << endl; hasBorrowed
    = false;
    } else {
```

COMPUTER SCIENCE & ENGINEERING

```
    cout << "Borrower " << name << " (ID: " << ID << ") has not borrowed any
    book." << endl;
    }
}

void performAction(int action) { if
    (action == 1)
    {
        borrowBook();
    } else if (action == 2) { returnBook();
    } else { cout << "Invalid action type"
    << endl; }
}
```

```
};

int main() { string title,
              author, name; int ISBN,
              ID, action;

    // Book details input
    cout << "Book Details: " << endl;
    cout << "Title: ";
    cin.ignore();
    getline(cin, title);
    cout << "Author: ";
    getline(cin, author);
    cout << "ISBN: ";
    cin >> ISBN;

    // Borrower details input
    cout << "Borrower Details: " << endl;
    cout << "Name: ";
    cin.ignore();
    getline(cin, name);
    cout << "ID: ";
    cin >> ID;

    // Create library object and perform action
    Library library(title, author, ISBN, name, ID);
    cout << "Action (1 to borrow, 2 to return): ";
    cin >> action;
    library.performAction(action);

    return 0;
}
```

4. Output:

```
input
Enter Employee Type (1 for Manager, 2 for Developer): 2
Enter Developer Details:
Name: Saloni
ID: 16659
Salary: 45000
Extra Hours Worked: 6
Hourly Rate: 10
Name: Saloni
ID: 16659
Base Salary: 45000
Overtime Compensation: 60
Total Earnings (Salary + Overtime Compensation): 45060

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

Problem: 20

1. **Aim:** Implement Polymorphism for Banking Transactions.
2. **Objective:** 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:
 - a. SavingsAccount: Interest = Balance × Rate × Time.
 - b. CurrentAccount: No interest, but includes a maintenance fee deduction.

3. Code:

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

COMPUTER SCIENCE & ENGINEERING

```
class Account { public: virtual double
calculateInterest() = 0; virtual
~Account() {}
};
```

```
class SavingsAccount : public Account {
private: double balance; double
interestRate; int time;
```

```
public:

    SavingsAccount(double b, double r, int t) : balance(b), interestRate(r), time(t) {}

    double calculateInterest() override { return
    balance * interestRate * time / 100; } };

class CurrentAccount : public Account { private:
    double balance; double maintenanceFee;

public:

    CurrentAccount(double b, double fee) : balance(b), maintenanceFee(fee) {}

    double    calculateInterest()    override    {
        balance -= maintenanceFee;
        return balance;
    }
};

int main() { int accountType;
    double balance, interestRate, maintenanceFee; int
    time;

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

    if (accountType == 1) { cout <<
        "Enter Balance: "; cin
        >> balance;  cout << "Enter Interest Rate
        (percentage): "; cin
        >> interestRate;  cout << "Enter
        Time (in years): "; cin
        >> time;

        SavingsAccount savings(balance, interestRate, time);
        cout << "Savings Account Interest: " << savings.calculateInterest() << endl;
    } else if (accountType == 2)
    { cout << "Enter Balance:
```

```
    "; cin >> balance;  
    cout << "Enter Monthly Maintenance Fee: "; cin >>  
    maintenanceFee;
```

```
    CurrentAccount current(balance, maintenanceFee); cout << "Balance after fee  
    deduction: " << current.calculateInterest() << endl;  
    } else { cout << "Invalid account type." << endl;  
    }
```

```
return 0;
```

```
}
```

4. Output:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



```
Enter Account Type (1 for Savings, 2 for Current): 1
Enter Balance: 50000
Enter Interest Rate (percentage): 10
Enter Time (in years): 5
Savings Account Interest: 25000

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

Problem: 21

1. **Aim:** Hierarchical Inheritance for Employee Management System.
2. **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:
 - a. Manager: Add and calculate bonuses based on performance ratings.
 - b. Developer: Add and calculate overtime compensation based on extra hours worked.
 - c. The program should allow input for both types of employees and display their total earnings.

3. **Code:**

```
#include <iostream> 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

    calculateEarnings() = 0;

    void display() { cout << "Employee: " << name << " (ID: "
        << id << ")\n"; cout << "Base Salary: " << salary << endl; }

};
```

COMPUTER SCIENCE & ENGINEERING

```
class Manager : public Employee { private:
    int rating;

public:
    Manager(string n, int i, int s, int r) : Employee(n, i, s), rating(r) {}

    void calculateEarnings() override {
        double bonus = salary * (rating * 0.10); double
        totalEarnings = salary + bonus; cout << "Role:
        Manager\n"; cout << "Bonus: " << bonus << endl;
        cout << "Total Earnings: " << totalEarnings << endl;
    }
};
```

```
class Developer : public Employee { private:
    int extraHours;

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

    void calculateEarnings() override { double
        overtimeCompensation = extraHours * 500; double
        totalEarnings = salary + overtimeCompensation; cout
        << "Role: Developer\n";
        cout << "Overtime Compensation: " << overtimeCompensation << endl;
        cout << "Total Earnings: " << totalEarnings << endl; }
};
```

```
int main() { int
    type; string
    name;
    int id, salary, rating, extraHours;

    cout << "Enter Employee Type (1 for Manager, 2 for Developer): "; cin >>
    type;

    if (type == 1) { cout <<
        "Enter Name: ";
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
    cin.ignore(); getline(cin, name); cout
    << "Enter ID: "; cin >> id; cout <<
    "Enter Salary: "; cin >> salary; cout
    << "Enter Rating (1 to 5): "; cin >>
    rating;
```

```
    Manager m(name, id, salary, rating); m.display();
    m.calculateEarnings(); }
else if (type == 2) {
    cout << "Enter Name: "; cin.ignore();
    getline(cin, name); cout << "Enter ID: ";
    cin >> id; cout << "Enter Salary: "; cin >>
    salary;    cout << "Enter Extra Hours
    Worked: "; cin
    >> extraHours;

    Developer d(name, id, salary, extraHours); d.display(); d.calculateEarnings()
; } else { cout << "Invalid employee
type.\n"; }

return 0;
}
```

4. Output:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



```
Enter Employee Type (1 for Manager, 2 for Developer): 2
Enter Developer Details:
Name: Saloni
ID: 16659
Salary: 45000
Extra Hours Worked: 6
Hourly Rate: 10
Name: Saloni
ID: 16659
Base Salary: 45000
Overtime Compensation: 60
Total Earnings (Salary + Overtime Compensation): 45060

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

Problem: 22

1. **Aim:** Multi-Level Inheritance for Vehicle Simulation
2. **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:
 - a. Fuel Efficiency: Miles per gallon (for Car).
 - b. Range: Total distance the electric car can travel with a full charge.
3. **Code:**

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

class Vehicle { protected:
```

```
string  brand;  
string  model;  
double  
mileage;
```

public:

```
Vehicle(string b, string m, double mil) : brand(b), model(m), mileage(mil) {}
```

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

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
virtual void calculate() = 0; // Pure virtual function };
```

```
class Car : public Vehicle { protected:  
    double      fuel;  double  
    distanceCovered;
```

public:

```
Car(string b, string m, double mil, double f, double d)  
    : Vehicle(b, m, mil), fuel(f), distanceCovered(d) {}
```

```
void calculate() override { double fuelEfficiency  
    = distanceCovered / fuel; cout << "Fuel Efficiency: " << fuelEfficiency <<  
    " miles/gallon" << endl;  
}  
};
```

```
class ElectricCar : public Car { private:  
    double      batteryCapacity;  
    double efficiency;
```

public:

```
ElectricCar(string b, string m, double mil, double f, double d, double bc, double e)  
    : Car(b, m, mil, f, d), batteryCapacity(bc), efficiency(e) {}
```

```
void calculate() override { double range =  
    batteryCapacity * efficiency; cout << "Range: "  
    << range << " miles" << endl; }  
};  
  
int main() {  
    int  
    vehicleType; string  
    brand, model;  
    double mileage;  
  
    cout << "Enter Vehicle Type (1 for Car, 2 for Electric Car): "; cin >>  
    vehicleType;
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
    cout << "Enter Brand: "; cin.ignore();  
    getline(cin, brand);  
  
    cout << "Enter Model: ";  
    getline(cin, model);  
  
    cout << "Enter Mileage: ";  
    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);  
        car.display(); car.calculate();
```

```
} else if (vehicleType == 2) { double fuel = 0, distanceCovered = 0, batteryCapacity,
    efficiency; cout << "Enter Battery Capacity
    (kWh): "; cin >> batteryCapacity; cout <<
    "Enter Efficiency (miles per kWh): "; cin >>
    efficiency;
```

```
    ElectricCar eCar(brand, model, mileage, fuel, distanceCovered, batteryCapacity,
    efficiency); eCar.display(); eCar.calculate();
    } else { cout << "Invalid vehicle type." <<
    endl; }
```

```
return 0;
```

```
}
```

4. Output:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
Enter Vehicle Type (1 for Car, 2 for Electric Car): 1
Enter Brand: Tesla
Enter Model: Model s
Enter Mileage: 5000
Enter Fuel (gallons): 5
Enter Distance Covered (miles): 400
Vehicle: Tesla Model s
Mileage: 5000 miles
Fuel Efficiency: 80 miles/gallon

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

Problem: 23

1. **Aim:** Function Overloading for Complex Number Operations.
2. **Objective:** Design a C++ program using function overloading to perform arithmetic operations on complex numbers. Define a Complex class with real and imaginary parts.
Overload functions to handle the following operations:
 - a. Addition: Sum of two complex numbers.
 - b. Multiplication: Product of two complex numbers.
 - c. Magnitude: Calculate the magnitude of a single complex number.
 - d. The program should allow the user to select an operation, input complex numbers, and display results in the format $a + bi$ or $a - bi$ (where b is the imaginary part).
3. **Code:**
`#include <iostream>`

```
#include <cmath> // For calculating the magnitude using namespace  
std;
```

```
class Complex { private:  
    int real;  
    int imaginary;
```

public:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
    // Constructor with default values  
    Complex(int r = 0, int i = 0) : real(r), imaginary(i) {}  
  
    // Function to display the complex number void  
    display() { if (imaginary >= 0) cout << real << " + " <<  
        imaginary << "i" << endl;  
        else cout << real << " - " << -imaginary << "i" << endl;  
    }  
  
    // Overloading the + operator for complex number addition  
    Complex operator+(const Complex &c) { return  
        Complex(real + c.real, imaginary + c.imaginary); }  
  
    // Overloading the * operator for complex number multiplication  
    Complex operator*(const Complex &c) { int newReal = real  
        * c.real - imaginary * c.imaginary; int newImaginary = real  
        * c.imaginary + imaginary * c.real; return  
        Complex(newReal, newImaginary);  
    }  
  
    // Function to calculate the magnitude of the complex number  
    double magnitude() { return sqrt(real * real + imaginary *  
        imaginary); }  
};
```

```
int main() { int operation;
    cout << "Enter operation type (1 for Addition, 2 for Multiplication, 3 for Magnitude): "; cin >>
    operation;

    if (operation == 1 || operation == 2) { int real1,
        imaginary1, real2, imaginary2;
        cout << "Enter complex number 1 (real and imaginary): "; cin
        >> real1 >> imaginary1; Complex c1(real1,
        imaginary1);

        cout << "Enter complex number 2 (real and imaginary): "; cin >>
        real2 >> imaginary2;
```

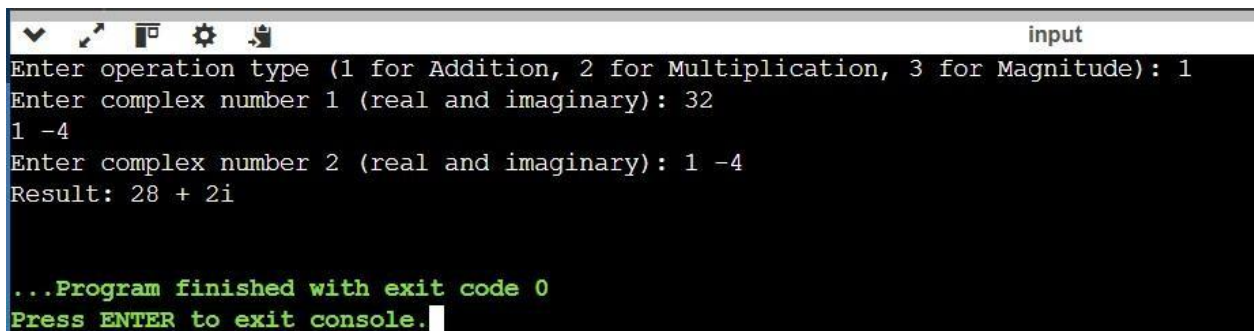
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
    Complex c2(real2, imaginary2);

    if (operation == 1) {
        Complex result = c1 + c2;
        cout << "Result:
        "; result.display();
    }
    else if (operation == 2) {
        Complex result = c1 * c2;
        cout << "Result:
        "; result.display();
    }
}
else if (operation == 3) { int
    real, imaginary;
    cout << "Enter complex number (real and imaginary): "; cin
    >> real >> imaginary; Complex
    c(real, imaginary);
    cout << "Result: Magnitude = " << c.magnitude() << endl;
```

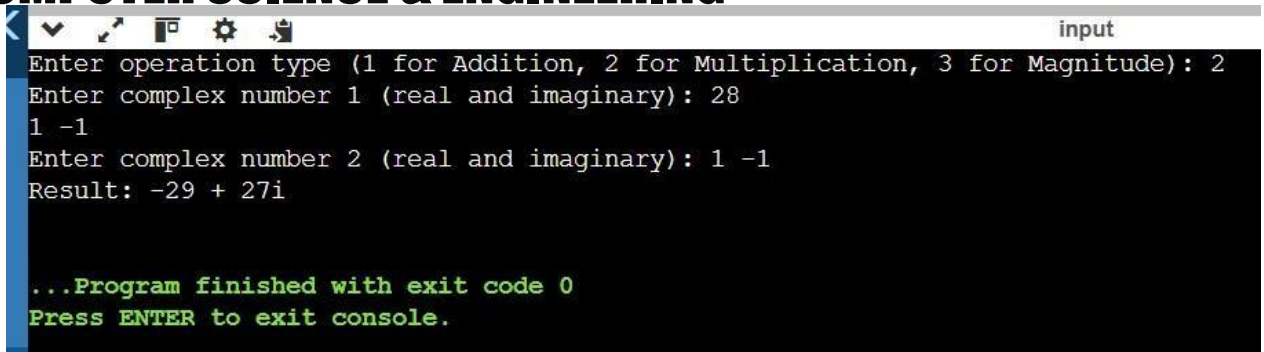
```
        } else  
        { cout << "Invalid operation type." << endl;  
        }  
  
return 0;  
  
}
```

4. Output:



```
input  
Enter operation type (1 for Addition, 2 for Multiplication, 3 for Magnitude): 1  
Enter complex number 1 (real and imaginary): 32  
1 -4  
Enter complex number 2 (real and imaginary): 1 -4  
Result: 28 + 2i  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



```
input  
Enter operation type (1 for Addition, 2 for Multiplication, 3 for Magnitude): 2  
Enter complex number 1 (real and imaginary): 28  
1 -1  
Enter complex number 2 (real and imaginary): 1 -1  
Result: -29 + 27i  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Problem: 24

1. **Aim:** Polymorphism for Shape Area Calculations.
2. **Objective:** Create a C++ program that uses polymorphism to calculate the area of various shapes. Define a base class Shape with a virtual method calculateArea(). Extend this base class into the following derived classes:

- a. Rectangle: Calculates the area based on length and width.
- b. Circle: Calculates the area based on the radius.
- c. Triangle: Calculates the area using base and height.
- d. The program should use dynamic polymorphism to handle these shapes and display the area of each.

3. Code:

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

```
class Shape
{ public: virtual float calculateArea()
= 0; };
```

```
class Rectangle : public Shape
{ private:
float length, width;
```

public:

```
Rectangle(float l, float w) : length(l), width(w) {} float calculateArea() override
{ return length * width;
}
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
};
```

```
class Circle : public Shape
{ private: float radius;
public:
```

```
Circle(float r) : radius(r) {} float calculateArea() override
{ return M_PI * radius * radius; }
};
```

```
class Triangle : public Shape
{ private:
    float base, height;

public:
    Triangle(float b, float h) : base(b), height(h) {} float calculateArea() override
    { return 0.5 * base * height;
    }
};

int main()
{ int shapeType; cout << "Enter Shape Type (1 for Rectangle, 2 for Circle, 3 for
  Triangle): ";
  cin >> shapeType; Shape *shape = nullptr; if
  (shapeType == 1)
  { float length, width; cout << "Enter Length and Width for
    Rectangle: "; cin
    >> length >> width; shape = new Rectangle(length,
    width);
  } else if (shapeType ==
  2)
  { float radius; cout << "Enter Radius
    for Circle: "; cin >> radius; shape =
    new
    Circle(radius);
  }
}
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

else if (shapeType == 3)

```
{ float base, height; cout << "Enter Base and
  Height for Triangle: "; cin >> base >> height;
  shape = new Triangle(base, height);
} else
{ cout << "Invalid shape type." << endl;
  return 0; }

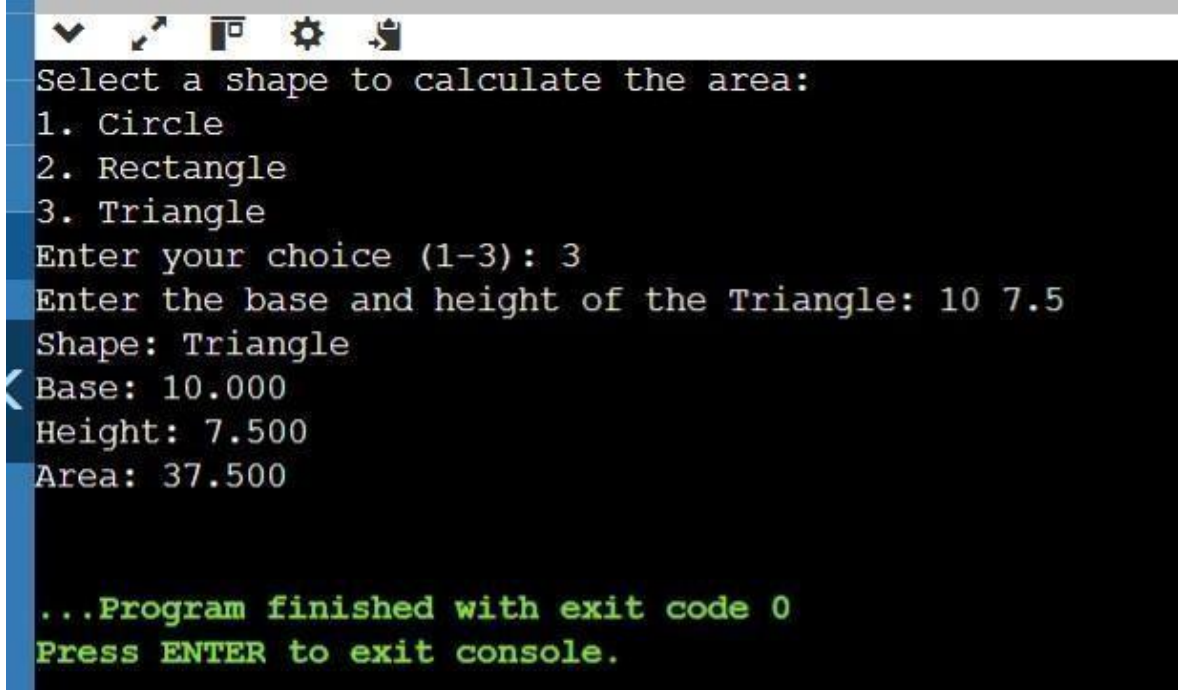
cout << fixed << setprecision(2); if
(shapeType == 1) cout << "Shape:
  Rectangle" << endl; else if
(shapeType == 2) cout <<
  "Shape: Circle" << endl; else if
(shapeType == 3)
  cout << "Shape: Triangle" << endl; cout << "Area:

" << shape->calculateArea() << endl;

delete shape; return
0; }
```

4. Output:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



```
Select a shape to calculate the area:
1. Circle
2. Rectangle
3. Triangle
Enter your choice (1-3): 3
Enter the base and height of the Triangle: 10 7.5
Shape: Triangle
Base: 10.000
Height: 7.500
Area: 37.500

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

Problem: 25

1. **Aim:** Advanced Function Overloading for Geometric Shapes.
2. **Objective:** Create a C++ program that demonstrates function overloading to calculate the area of different geometric shapes. Implement three overloaded functions named `calculateArea` that compute the area for the following shapes:
 - a. Circle: Accepts the radius.
 - b. Rectangle: Accepts the length and breadth.
 - c. Triangle: Accepts the base and height.
 - d. Additionally, use a menu-driven program to let the user choose the type of shape and input the respective parameters. Perform necessary validations on the input values.

3. **Code:**

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



```
float calculateArea(float radius) { return  
M_PI * radius * radius; }
```

```
float calculateRectangleArea(float length, float breadth) { return length  
* breadth;
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
}
```

```
float calculateTriangleArea(float base, float height) { return  
0.5 * base * height;  
}
```

```
int main() { int choice;  
cout << "Select a shape to calculate the area:" << endl;  
cout << "1. Circle" << endl; cout << "2. Rectangle" <<  
endl; cout << "3. Triangle" << endl; cout << "Enter  
your choice (1-3): "; cin >> choice;
```

```
// Validate choice if (choice < 1 || choice > 3) { cout << "Invalid choice. Please  
select a valid shape type." << endl; return 0;  
}
```

```
float area;
```

```
if (choice == 1) { float  
radius;  
cout << "Enter the radius of the Circle: "; cin >>  
radius;
```

```
if (radius <= 0) { cout << "Invalid input. Radius must be a positive number." <<  
endl; return 0;  
}
```

```
        area = calculateArea(radius); cout <<
        fixed << setprecision(3); cout <<
        "Shape: Circle" << endl; cout <<
        "Radius: " << radius << endl; cout <<
        "Area: " << area << endl;
    }
    else if (choice == 2) { float length, breadth; cout << "Enter
        the length and breadth of the Rectangle: ";
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
        cin >> length >> breadth;
    if (length <= 0 || breadth <= 0) { cout << "Invalid input. Length and breadth must be positive numbers." <<
        endl; return 0;
    }

    area = calculateRectangleArea(length, breadth);
    cout << fixed << setprecision(3); cout <<
    "Shape: Rectangle" << endl; cout << "Length: "
    << length << endl; cout << "Breadth: " <<
    breadth << endl; cout << "Area: " << area <<
    endl;
}
else if (choice == 3) { float
    base, height;
    cout << "Enter the base and height of the Triangle: "; cin >>
    base >> height;
    if (base <= 0 || height <= 0) { cout << "Invalid input. Base and height must be positive numbers." <<
        endl; return 0;
    }

    area = calculateTriangleArea(base,
    height); cout << fixed << setprecision(3); cout
    << "Shape: Triangle" << endl; cout << "Base: "
```

```
<< base << endl; cout << "Height: " << height  
<< endl; cout << "Area: " << area  
<< endl; }
```

```
return 0;
```

```
}
```

4. Output:



Discover. Learn. Empower.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
▼ ↗ 📄 ⚙️ 🖨️
Select a shape to calculate the area:
1. Circle
2. Rectangle
3. Triangle
Enter your choice (1-3): 1
Enter the radius of the Circle: 7.5
Shape: Circle
Radius: 7.500
Area: 176.715

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

```
▼ ↗ 📄 ⚙️ 🖨️
Select a shape to calculate the area:
1. Circle
2. Rectangle
3. Triangle
Enter your choice (1-3): 2
Enter the length and breadth of the Rectangle: 10 5
Shape: Rectangle
Length: 10.000
Breadth: 5.000
Area: 50.000

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

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
✓ ↗ 📄 ⚙️ 📋  
Select a shape to calculate the area:  
1. Circle  
2. Rectangle  
3. Triangle  
Enter your choice (1-3): 3  
Enter the base and height of the Triangle: 10 7.5  
Shape: Triangle  
Base: 10.000  
Height: 7.500  
Area: 37.500  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```