

Name: Yash Kumar

UID: 22BCS15424

Section: 22BCS_IOT-620

1) Sum of Natural Numbers up to N

Code:

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cout<<"enter"<<endl;
    cin>>n;
    int ans=0;
    for(int i=0;i<=n;i++){
        ans+=i;
    }
    cout<<ans;
    return 0;
}
```

Output:

main.cpp

```
1 // Online C++ compiler to run C++ program online
2 #include <iostream>
3 using namespace std;
4 int main() {
5     int n;
6     cout<<"enter"<<endl;
7     cin>>n;
8     int ans=0;
9     for(int i=0;i<=n;i++){
10         ans+=i;
11     }
12     cout<<ans;
13     return 0;
14 }
```

2) Check if a Number is Prime

Code:

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

```

else{
    cout<<"not prime"<<endl;
}

return 0;
}

```

Output:

```

// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
int main() {
    int n;
    int flag=0;
    cout<<"enter"<<endl;
    cin>>n;
    for(int i=2;i<=n/2;i++){
        if(n%i==0){
            flag=1;
        }
    }
    if(flag==0){
        cout<<" prime"<<endl;
    }
    else{
        cout<<"not prime"<<endl;
    }

    return 0;
}

```

3) Check if a Number is a Palindrome

Code:

```

#include <iostream>
using namespace std;

```

```
int main() {  
    int n;  
    cout<<"enter"<<endl;  
    cin>>n;  
    int temp=n;  
    int reverse=0;  
    while(temp!=0){  
        reverse=(reverse*10)+(temp%10);  
        temp=temp/10;  
    }  
    if(reverse==n){  
        cout<<"yes";  
    }  
    else{  
        cout<<"no";  
    }  
    return 0;  
}
```

Output:

main.cpp



Share

```
1 // Online C++ compiler to run C++ program online
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     int n;
7     cout<<"enter"<<endl;
8     cin>>n;
9     int temp=n;
10    int reverse=0;
11    while(temp!=0){
12        reverse=(reverse*10)+(temp%10);
13        temp=temp/10;
14    }
15    if(reverse==n){
16        cout<<"yes";
17    }
18    else{
19        cout<<"no";
20    }
21    return 0;
22 }
```

4)Function Overloading for Calculating Area.

Code:

```
#include <iostream>
using namespace std;
void sum(int a,int b){
    int c=a*b;
    cout<<c<<endl;
}
void sum(double a,double b){
```

```
double c=a*b;
cout<<c<<endl;

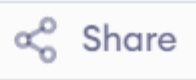
}
void sum(int a,double b){
    double c=a*b;
    cout<<c<<endl;

}
```

```
int main() {
    sum(5,4);
    sum(5.5,6.5);
    sum(4,8.8);
    return 0;
}
```

Output:

main.cpp



```
1 // Online C++ compiler to run C++ program online
2 #include <iostream>
3 using namespace std;
4 void sum(int a,int b){
5     int c=a*b;
6     cout<<c<<endl;
7 }
8 void sum(double a,double b){
9     double c=a*b;
10    cout<<c<<endl;
11
12 }
13 void sum(int a,double b){
14     double c=a*b;
15     cout<<c<<endl;
16
17 }
18
19
20 int main() {
21     sum(5,4);
22     sum(5.5,6.5);
23     sum(4,8.8);
24     return 0;
25 }
```

5) Function Overloading with Hierarchical Structure.

Objective

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

```
class Employee {
public:
```

```

    virtual void calculateSalary() {
        cout << "Calculating salary for an employee." << endl;
    }
};

class Intern : public Employee {
private:
    double stipend;
public:
    Intern(double stipend) : stipend(stipend) {}

    void calculateSalary() override {
        cout << "Intern's salary (stipend): $" << stipend << endl;
    }
};

class RegularEmployee : public Employee {
protected:
    double baseSalary;
    double bonuses;
public:
    RegularEmployee(double baseSalary, double bonuses)
        : baseSalary(baseSalary), bonuses(bonuses) {}

    void calculateSalary() override {
        cout << "Regular Employee's salary: $" << baseSalary + bonuses << endl;
    }
};

class Manager : public RegularEmployee {
private:
    double performanceIncentives;
public:
    Manager(double baseSalary, double bonuses, double performanceIncentives)
        : RegularEmployee(baseSalary, bonuses), performanceIncentives(performanceIncentives) {}

    void calculateSalary() override {
        cout << "Manager's salary: $" << (baseSalary + bonuses + performanceIncentives) << endl;
    }
};

int main() {
    Intern intern(1000);
    RegularEmployee regularEmployee(3000, 500);
    Manager manager(5000, 1000, 2000);



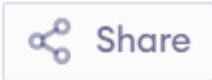
    intern.calculateSalary();
    regularEmployee.calculateSalary();
    manager.calculateSalary();

    return 0;
}

```


}

Output:

```
main.cpp    Share
1  #include <iostream>
2  using namespace std;
3
4  class Employee {
5  public:
6      virtual void calculateSalary() {
7          cout << "Calculating salary for an employee." << endl;
8      }
9  };
10
11 class Intern : public Employee {
12 private:
13     double stipend;
14 public:
15     Intern(double stipend) : stipend(stipend) {}
16
17     void calculateSalary() override {
18         cout << "Intern's salary (stipend): $" << stipend << endl;
19     }
20 };
21
22 class RegularEmployee : public Employee {
23 protected:
24     double baseSalary;
25     double bonuses;
26 public:
```

6) Polymorphism with Shape Area Calculation.

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

```

class Shape {
public:
    virtual double calculateArea() = 0;
    virtual ~Shape() {}
};

class Circle : public Shape {
private:
    double radius;
public:
    Circle(double r) : radius(r) {}

    double calculateArea() override {
        return M_PI * radius * radius;
    }
};

class Rectangle : public Shape {
private:
    double length, breadth;
public:
    Rectangle(double l, double b) : length(l), breadth(b) {}

    double calculateArea() override {
        return length * breadth;
    }
};

class Triangle : public Shape {
private:
    double base, height;
public:
    Triangle(double b, double h) : base(b), height(h) {}

    double calculateArea() override {
        return 0.5 * base * height;
    }
};

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

    cout << "Enter the radius of the circle: ";
    cin >> radius;

    cout << "Enter the length and breadth of the rectangle: ";
    cin >> length >> breadth;

    cout << "Enter the base and height of the triangle: ";
    cin >> base >> height;
}

```

```
Shape* circle = new Circle(radius);
Shape* rectangle = new Rectangle(length, breadth);
Shape* triangle = new Triangle(base, height);

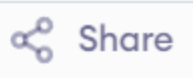
cout << "Area of Circle: " << circle->calculateArea() << endl;
cout << "Area of Rectangle: " << rectangle->calculateArea() << endl;
cout << "Area of Triangle: " << triangle->calculateArea() << endl;

delete circle;
delete rectangle;
delete triangle;

return 0;
}
```

Output:

main.cpp



```
46
47     cout << "Enter the radius of the circle: ";
48     cin >> radius;
49
50     cout << "Enter the length and breadth of the rectangle: ";
51     cin >> length >> breadth;
52
53     cout << "Enter the base and height of the triangle: ";
54     cin >> base >> height;
55
56     Shape* circle = new Circle(radius);
57     Shape* rectangle = new Rectangle(length, breadth);
58     Shape* triangle = new Triangle(base, height);
59
60     cout << "Area of Circle: " << circle->calculateArea() <<
61     cout << "Area of Rectangle: " << rectangle->calculateArea()
        endl;
62     cout << "Area of Triangle: " << triangle->calculateArea()
        endl;
63
64     delete circle;
65     delete rectangle;
```

7) Inheritance with Student and Result Classes.

Code:

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

```
class Student {
protected:
    int rollNumber;
    string name;
public:
    void getStudentDetails() {
```

```

        cout << "Enter Roll Number: ";
        cin >> rollNumber;
        cin.ignore();
        cout << "Enter Name: ";
        getline(cin, name);
    }

    void displayStudentDetails() {
        cout << "Roll Number: " << rollNumber << endl;
        cout << "Name: " << name << endl;
    }
};

class Result : public Student {
private:
    float marks1, marks2, marks3;
    float totalMarks;
    float percentage;
public:
    void getMarks() {
        cout << "Enter marks for Subject 1: ";
        cin >> marks1;
        cout << "Enter marks for Subject 2: ";
        cin >> marks2;
        cout << "Enter marks for Subject 3: ";
        cin >> marks3;
    }

    void calculateResult() {
        totalMarks = marks1 + marks2 + marks3;
        percentage = (totalMarks / 300) * 100;
    }

    void displayResult() {
        displayStudentDetails();
        cout << "Total Marks: " << totalMarks << "/300" << endl;
        cout << "Percentage: " << percentage << "%" << endl;
    }
};

int main() {
    Result studentResult;

    studentResult.getStudentDetails();
    studentResult.getMarks();
    studentResult.calculateResult();
    studentResult.displayResult();

    return 0;
}

```

Output:

main.cpp



```
36     cin >> marks3;
37 }
38
39 void calculateResult() {
40     totalMarks = marks1 + marks2 + marks3;
41     percentage = (totalMarks / 300) * 100;
42 }
43
44 void displayResult() {
45     displayStudentDetails();
46     cout << "Total Marks: " << totalMarks << "/300" << endl;
47     cout << "Percentage: " << percentage << "%" << endl;
48 }
49 };
50
51 int main() {
52     Result studentResult;
53
54     studentResult.getStudentDetails();
55     studentResult.getMarks();
56     studentResult.calculateResult();
57     studentResult.displayResult();
58
59     return 0;
```

8) Function Overloading for Complex Number Operations.

Code:

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

class Complex {
```

```

private:
    double real;
    double imag;

public:
    Complex() : real(0), imag(0) {}
    Complex(double r, double i) : real(r), imag(i) {}

    void input() {
        cout << "Enter real part: ";
        cin >> real;
        cout << "Enter imaginary part: ";
        cin >> imag;
    }

    void display() const {
        if (imag >= 0)
            cout << real << " + " << imag << "i" << endl;
        else
            cout << real << " - " << -imag << "i" << endl;
    }

    Complex operator+(const Complex& c) {
        return Complex(real + c.real, imag + c.imag);
    }

    Complex operator*(const Complex& c) {
        return Complex(real * c.real - imag * c.imag, real * c.imag + imag * c.real);
    }

    double magnitude() const {
        return sqrt(real * real + imag * imag);
    }
};

int main() {
    int choice;
    Complex c1, c2, result;

    cout << "Complex Number Operations\n";
    cout << "1. Addition\n";
    cout << "2. Multiplication\n";
    cout << "3. Magnitude\n";
    cout << "Enter your choice (1/2/3): ";
    cin >> choice;

    if (choice == 1 || choice == 2) {
        cout << "Enter first complex number:\n";
        c1.input();
        cout << "Enter second complex number:\n";
        c2.input();
    }
}

```

```
} else if (choice == 3) {
    cout << "Enter a complex number:\n";
    c1.input();
}

switch (choice) {
    case 1:
        result = c1 + c2;
        cout << "Sum: ";
        result.display();
        break;
    case 2:
        result = c1 * c2;
        cout << "Product: ";
        result.display();
        break;
    case 3:
        cout << "Magnitude of the complex number: " << fixed << setprecision(2) << c1.magnitude()
        << endl;
        break;
    default:
        cout << "Invalid choice\n";
}

return 0;
}
```

Output:

main.cpp

Share

```
59         cout << "Enter a complex number:\n";
60         c1.input();
61     }
62
63     switch (choice) {
64     case 1:
65         result = c1 + c2;
66         cout << "Sum: ";
67         result.display();
68         break;
69     case 2:
70         result = c1 * c2;
71         cout << "Product: ";
72         result.display();
73         break;
74     case 3:
75         cout << "Magnitude of the complex number: " << fi
76             setprecision(2) << c1.magnitude() << endl;
77         break;
78     default:
79         cout << "Invalid choice\n";
80     }
81     return 0;
82 }
```

9)Print Multiplication Table of a Number

Code:

```
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
int main() {
    int n;
```

```

cout<<"enter"<<endl;
cin>>n;
int ans=0;
for(int i=1;i<=10;i++){
    ans=n*i;
    cout<<n<<"*"<<i<<"= " <<ans<<endl;
}

return 0;
}

```

Output:

```

#include <iostream>
using namespace std;
int main() {
    int n;
    cout<<"enter"<<endl;
    cin>>n;
    int ans=0;
    for(int i=1;i<=10;i++){
        ans=n*i;
        cout<<n<<"*"<<i<<"= " <<ans<<endl;
    }

    return 0;
}

```

10)) Count Digits in a Number

Code:

```

#include <iostream>
using namespace std;

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

```

}

Output:

```
1 // Online C++ compiler to run C++ program online
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     int n;
7     cout<<"enter"<<endl;
8     cin>>n;
9     int count=0;
10    while(n>0){
11        n=n/10;
12        count++;
13    }
14    cout<<count;
15    return 0;
16 }
```